Enterprise Architect

User Guide Series

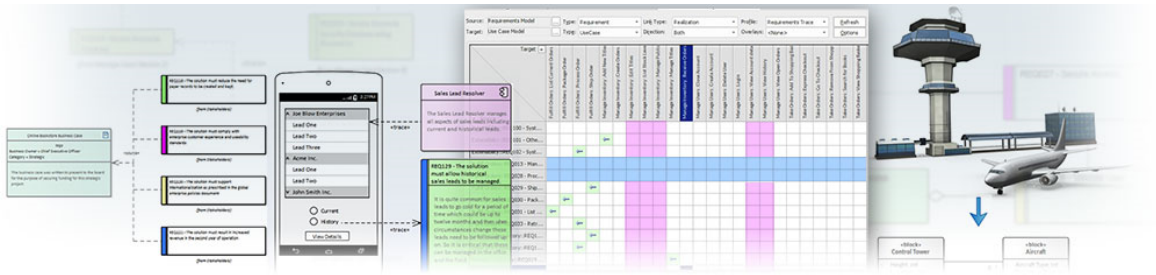
Modeling Domains

What support for modeling domains? Sparx Systems Enterprise Architect supports a range of modeling languages, technologies and methods that can be used in modeling and integrating different domains to reflect aspects of a complete business or process.

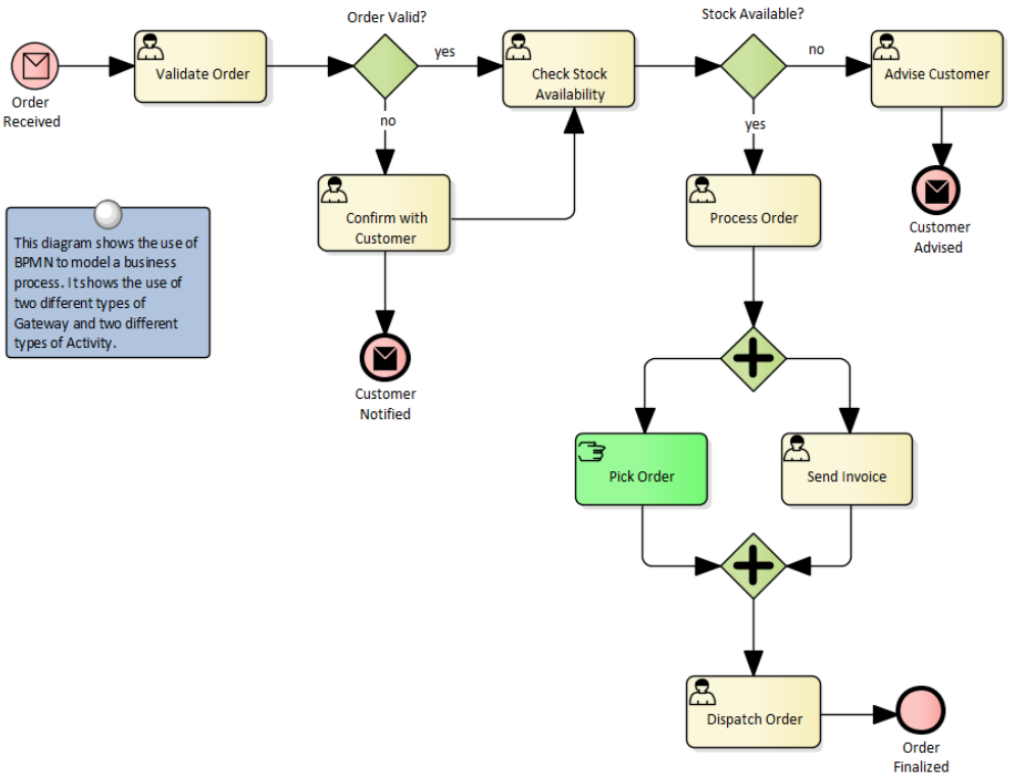
Author: Sparx Systems
Date: 2022-02-03
Version: 16.0

Table of Contents

Modeling Domains



The world and the systems that underpin almost every human activity, from education to aerospace and from mining to manufacturing, require collaboration between multi-disciplinary teams. There are teams of business strategists that steer the ship of the enterprise through often turbulent and uncharted waters. Enterprise Architects that design how the enterprise will operate and change over time. Business Analysts derive requirements from customer workshops and detail processes, services, and information to prepare for technical analysis. Geospatial engineers and informaticians reason about location and its importance to human activity. Technical teams develop applications, information, and database systems in cloud or hybrid architectures, and systems engineers develop robust cyber-mechanical systems based on detailed designs and architectures.



Enterprise Architect supports all these domains and roles and is a purpose-built collaboration platform where users can chat, message, review, discuss and unite their otherwise disparate models. The power of modeling is the ability to integrate the various system representations and stitch together models from multiple domains and disciplines. For example, the ability to integrate models that describe the geospatial aspects of a feature in the world, such as an airport, with regulatory and air traffic control models and baggage handling system models, provides clarity that has not been possible before. The ability to model these concepts in the language of multiple disciplines and then tie them together in a single modeling environment makes Enterprise Architect such a valuable and productive tool.

Modeling Domains

Domain	Description
ODM	Enterprise Architect enables you to develop large-scale ontologies within the fully-integrated modeling environment, for your project domain. ODM helps you to develop a formalized representation of business semantics and taxonomies, and a knowledge representation based on those formalizations.
Requirements	Enterprise Architect is one of the few UML tools that integrate Requirements Management with other software development disciplines in the core product, by defining requirements within the model.
Business Modeling	Modeling the business process is an essential part of any software development process, enabling the analyst to capture the broad outline and procedures that govern what it is a business does.

Business Rules	Business Rule modeling captures the rules that govern a business, and their relationships with the entities and specific tasks within the organization or system.
BPMN	The Business Process Model and Notation (BPMN) is specifically targeted at the business modeling community and has a direct mapping to UML through BPMN Profiles; these profiles enable you to develop BPMN diagrams quickly and simply.
BPEL	Business Process Execution Language is an executable language for specifying interactions with Web Services. Enterprise Architect uses the BPMN profile as a graphical front-end to capture BPEL Process descriptions.
SysML	SysML is a general-purpose graphical modeling language for specifying, analyzing, designing, and verifying complex systems that might include hardware, software, information, personnel, procedures and facilities.
Data Modeling	Enterprise Architect provides easy-to-use tools for building and maintaining all of

	the fundamental data models - Conceptual, Logical and Physical; because Enterprise Architect lets you visualize each type of data model in the same repository, you can easily manage dependencies between each level of abstraction.
XSD	Enterprise Architect supports rapid modeling, forward engineering and reverse engineering of W3C XML schemas (XSD), critical for the development of a complete Service Oriented Architecture (SOA).
WSDL	Enterprise Architect enables rapid modeling, forward engineering and reverse engineering of W3C XML Web Service Definition Language (WSDL), critical for the development of a complete Service Oriented Architecture (SOA).
SPEM	The Software and Systems Process Engineering Metamodel (SPEM) is a conceptual framework for modeling, documenting, presenting, managing, interchanging, and enacting development methods and processes. SPEM 2.0 focuses on providing the

	additional information structures that you require for processes modeled with UML 2 Activities or BPMN/BPDM.
ArchiMate	ArchiMate is an open-standard enterprise architecture language based on the IEEE 1471 standard, providing a common language for describing the construction and operation of business processes, organizational structures, information flows, IT systems and technical infrastructure. It enables Enterprise Architects to clearly describe, analyze and visualize the relationships amongst business domains.
ArcGIS	ArcGIS is a suite of Geographic Information Systems (GIS) software products developed by Esri.
AML	The Archetype Modeling Language (AML) defines a standard means for representing clinical information.
Data Flow Diagrams	A Data Flow diagram (DFD) is a graphical representation of the flow of data through an information system; it can also be used to visualize data processing (structured design).

	Developing a DFD helps in identifying the transaction data in the data model.
Entity Relationship Diagrams	Entity-relationship modeling is an abstract and conceptual database modeling method, used to produce a schema or semantic data model of, for example, a relational database and its requirements, visualized in Entity-Relationship diagrams (ERDs). ERDs in Enterprise Architect assist you in building conceptual data models through to generating Data Definition Language (DDL) for the target DBMS.
Eriksson-Penker Extensions	Eriksson-Penker extensions provide a framework for UML business processing model extensions, to which an Enterprise Architect can add stereotypes and properties appropriate to their business. In Enterprise Architect, the Eriksson-Penker profile provides, through a set of stereotypes, a unique and effective means of visualizing and communicating business processes and the necessary flow of information within an organization.
Gang of Four	Gang of Four (GoF) Patterns are 23

Patterns	classic software Design Patterns providing recurring solutions to common problems in software design. Enterprise Architect provides each Pattern through an icon in the Diagram Toolbox.
ICONIX	The ICONIX Process is a streamlined approach to Use Case driven UML modeling that uses a core subset of UML diagrams and techniques to provide thorough support of object-oriented analysis and design. Its main activity is robustness analysis, a method for bridging the gap between analysis and design.
Mind Mapping	A Mind Map is an image-centered diagram used to represent semantic or other connections between words, ideas, tasks or other items arranged radially around a central key word or idea. A Mind Map is used to generate, visualize, structure and classify ideas, and as an aid in study, organization, problem solving, decision making, and writing.
SoaML	Service Oriented Architecture (SOA) is an architectural paradigm for defining

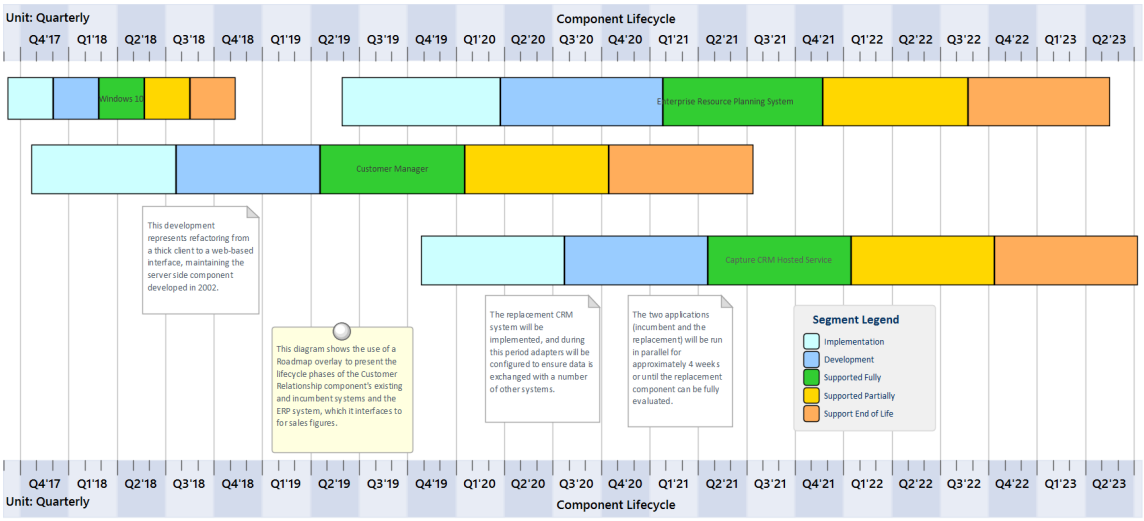
	how people, organizations and systems provide and use services to achieve results.
SOMF	The service-oriented modeling framework (SOMF) is a service-oriented development life cycle methodology, offering a number of modeling practices and disciplines that contribute to a successful service-oriented life cycle management and modeling.
Extended Diagrams	Enterprise Architect provides an additional set of diagram types that extend the core UML diagrams for domain-specific models. Also, the specialized modeling tools listed in the first part of this table each have their own specialized diagrams.
Inbuilt and Extension Stereotypes	Behavioral and Structural elements can be extended through the use of stereotypes; Enterprise Architect provides a number of inbuilt extensions.
Build Your Own Modeling Language	Enterprise Architect enables you to extend the scope both of your modeling and of the UML components you use, through the use of stereotypes, Profiles

	and Patterns to develop your own modeling applications.

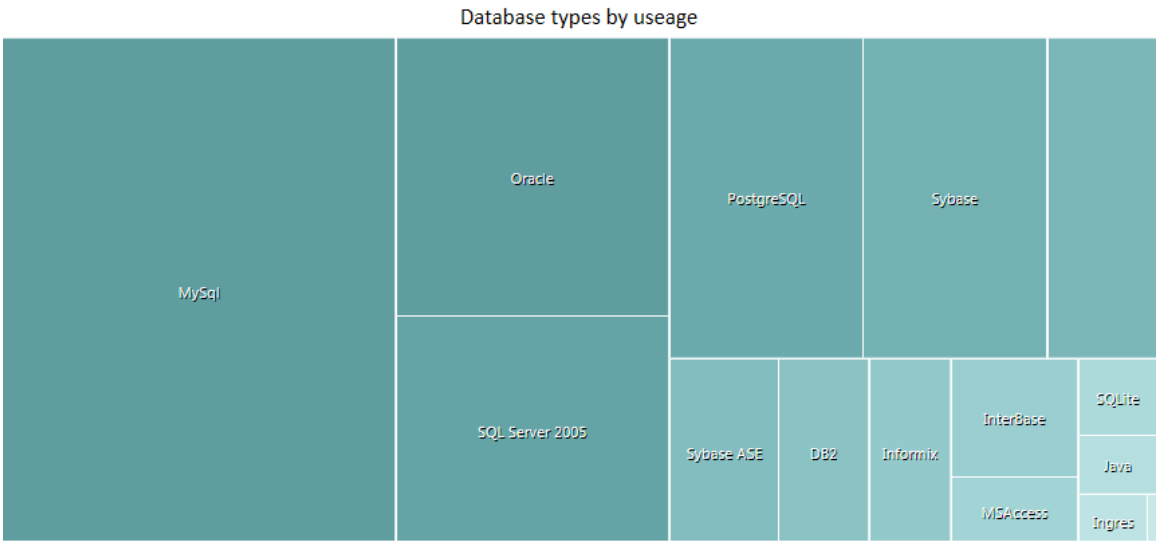
Guide to Enterprise Architecture

Enterprise Architecture has emerged as a discipline that can help steer the 'ship' of the enterprise through both quiet and turbulent waters, charting a course from its current location to a future location in a safe and streamlined way. The discipline has become more prevalent in recent years, but the precepts go back almost to the beginning of what is often termed the information age. An enterprise is one of the most complex man-made systems and is composed of human, political, social, software, hardware and technology components. In an enterprise of any appreciable size, it is impossible for a single person to understand the way the parts all work together, let alone understand its position in relation to the system of other organizations that form its environment, or to determine how it can evolve.

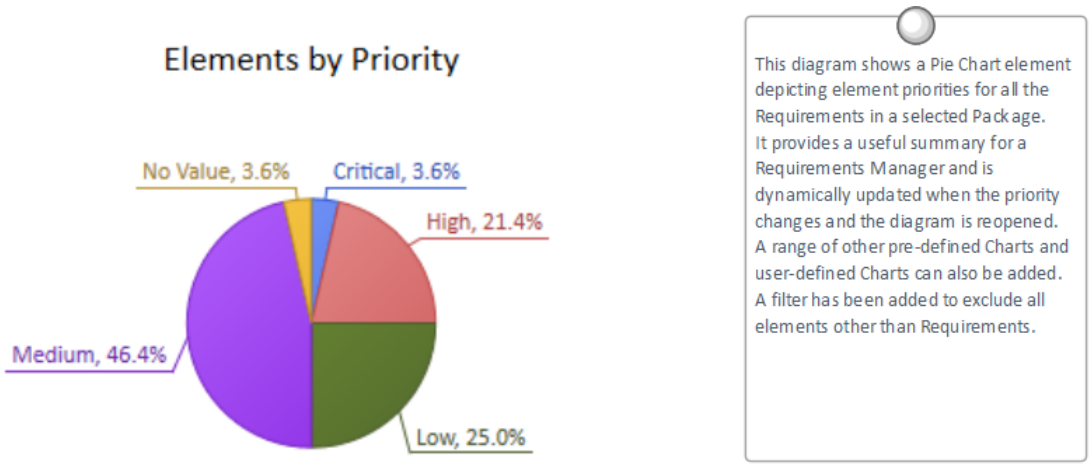
Enterprise Architecture can be used to create visualizations of the enterprise at different levels of abstractions and to create Roadmaps that show how the enterprise can be transitioned from its baseline (current) state to a target (future) state.



Enterprise Architect is a multi-featured platform that can be used to define the strategic context for an Enterprise Architecture, the Enterprise Architecture itself, and the Implementation initiatives that realize the designs and that finally deliver the business value. It can serve both as the architectural repository and as a tool for managing the process by which architectures are created and maintained, including an architectural requirements management platform. Rich visualization capabilities allow models to be transformed and presented in a variety of compelling ways that will delight stakeholders, from the executive level through to the implementation teams.



The tool can be used to define Strategic, Tactical and Solution Architectures and to provide compelling views for a wide range of stakeholders, from senior executives through to implementation partners. Business, Information, Application and Technology architectures can be created and managed, and baseline and target architectures defined, allowing transitions to be visualized.



Enterprise Architecture Overview

Enterprise Architecture has emerged as a critical discipline to ensure that an enterprise and the organizations it comprises have an understanding of the significant elements from which it is made, from strategic goals through to the business and information technology components that assist in achieving those goals. The discipline also allows enterprises to create architectures that will transition from where they are to where they need to be. Now more than ever in this age of digital disruption, when organizations can no longer rely on length of tenure in a field or being bigger than their competition as a safeguard against competitive forces, Enterprise Architecture as a discipline is vital.

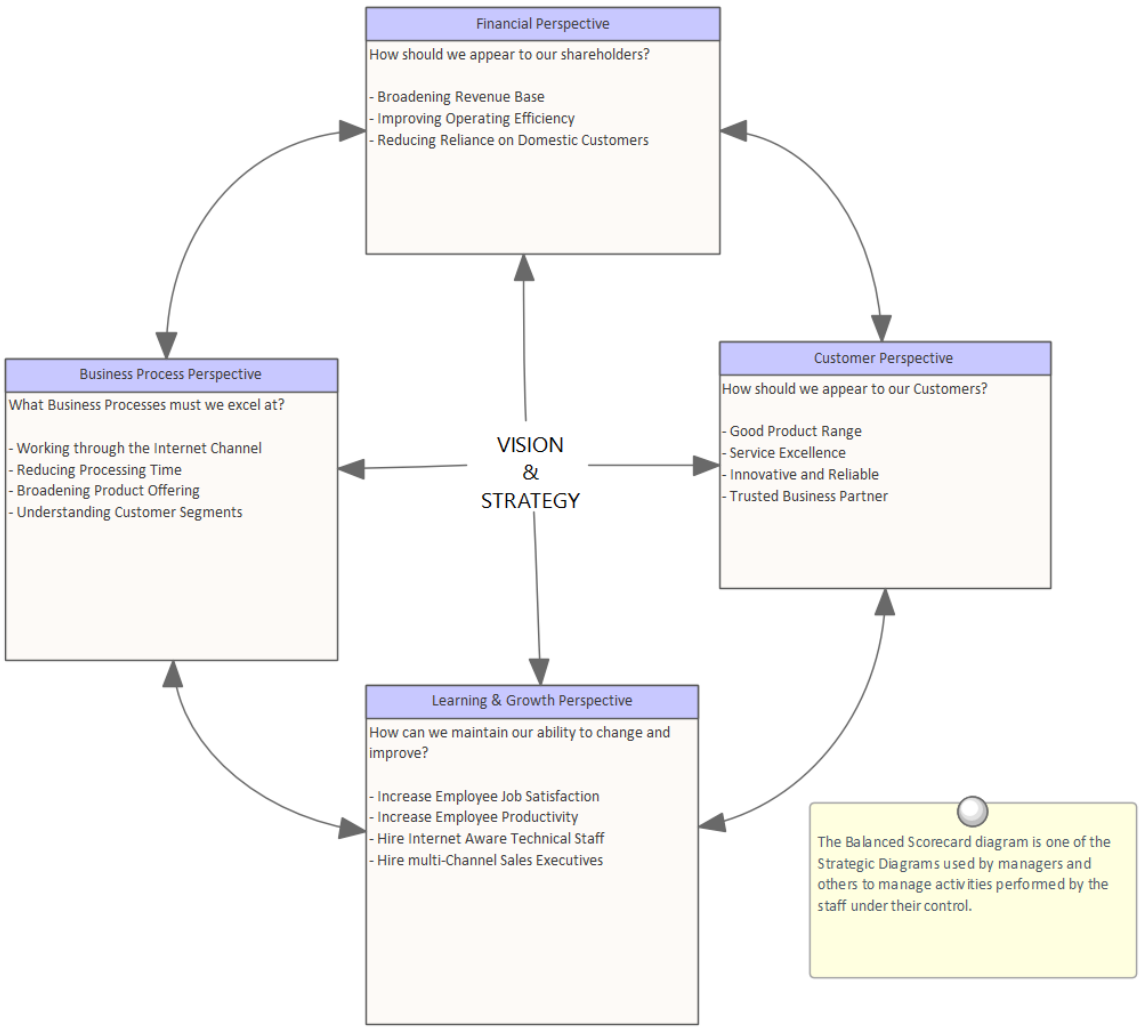
The profession of Enterprise Architect is rarely taught as a separate degree course at tertiary institutions, and it also suffers under the proliferation of large, conflicting and at times burdensome frameworks and a lack of tool support. This has led to architecture practitioners finding it difficult to articulate or demonstrate the value of their 'profession'.

This section addresses the questions:

- What is Enterprise Architecture?
- Where does it fit into the context of other disciplines?
- What are the characteristics of good architecture?

The section also discusses the levels, types and styles of architecture that exist, and describes the notational mechanisms that are at an architect's disposal.

Enterprise Architect's pragmatic approach to modeling, and the extensive set of facilities available to the architect and others, make it a versatile tool both as an architecture repository and as a platform for creating, managing and disseminating architectural work.



What is Enterprise Architecture

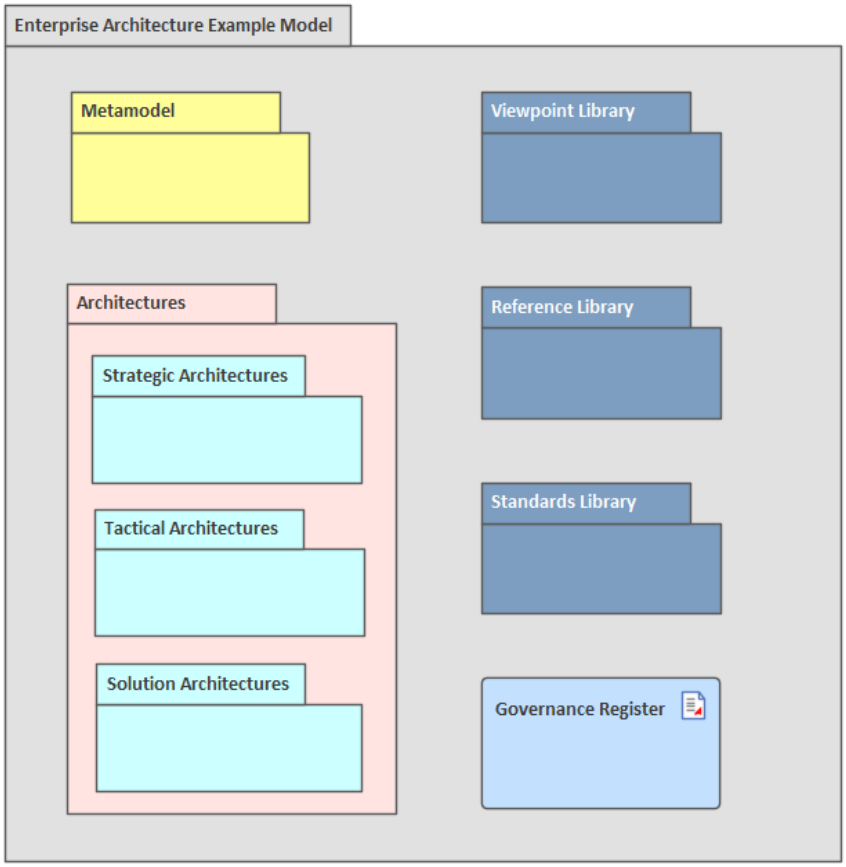
There is still robust and lasting debate about what architecture is in the context of an enterprise, which makes it difficult to provide a universally accepted definition of the term 'Enterprise Architecture'. This is not helped by the fact that there is not a standard curriculum for architecture or any of its flavors in our tertiary institutions, nor do many institutions teach architecture at a graduate or post graduate level. People come to the discipline from a wide range of backgrounds and produce an equally wide range of work products that come under the general heading of architecture.

The ISO/IEC 42010 standard defines architecture as:

'The fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution'.

In this context the system that is under discussion is the enterprise. Using this definition it is important to separate the fact that there is a description of the architecture (often called an architecture) and the architecture that has been built. Enterprise Architect is used to define or document the description of the architecture and therefore provides a representation or way of visualizing a planned or built architecture. The comprehensive facilities available in Enterprise Architect have been designed by practitioners and are continually improved to create a formidable tool that

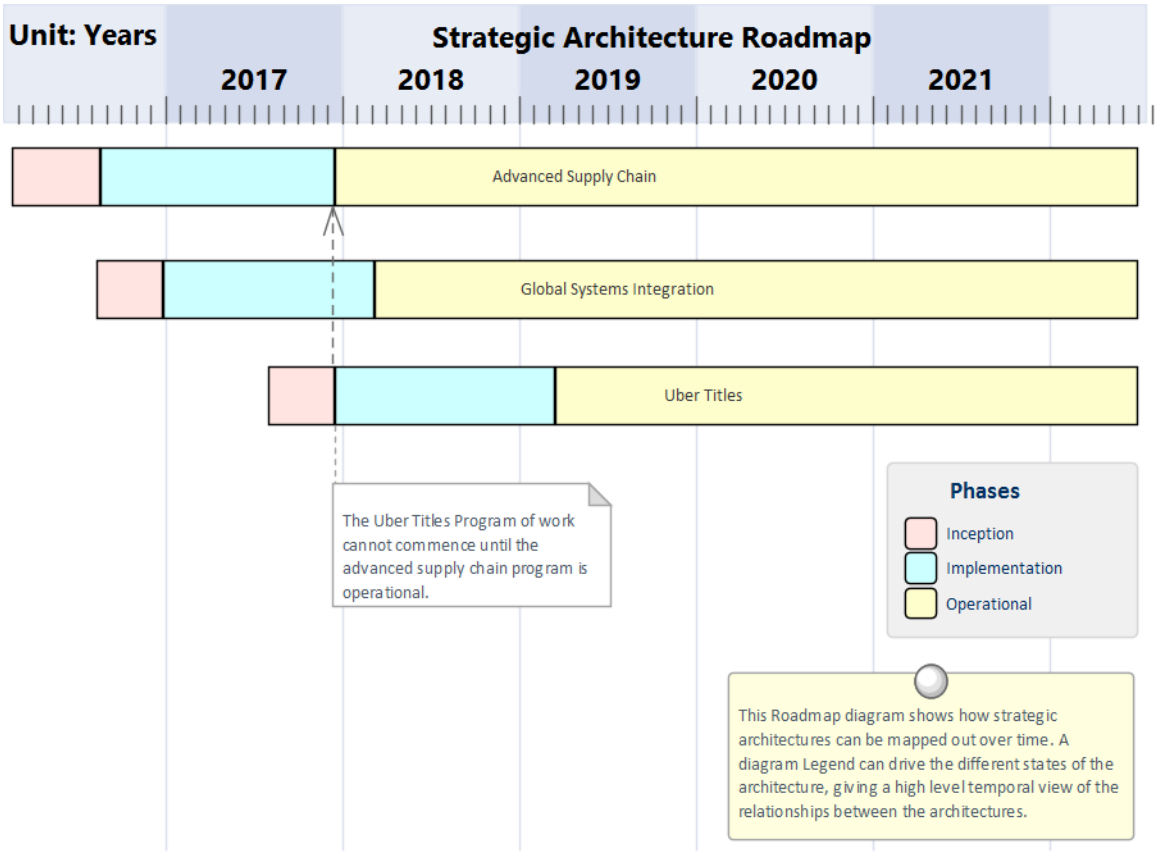
is both robust and flexible, and that makes creating and managing Enterprise Architectures easy, regardless of the definition that is applied.



Context for Enterprise Architecture

An Enterprise Architecture Program is best viewed as an operational unit of the business, and as such it has a context. The program must provide value to the business; it does this by ensuring that architectural effort is aligned with the strategic plans of the organization and that the implementation initiatives are carried out in such a way that honors the enterprise architecture.

Enterprise Architect has a number of convenient features to model and visualize the alignment of the architectures and the strategic plans. These visualizations can be used when assessing which Information Technology initiatives are contributing to the strategic business goals and objectives. Enterprise Architect has a number of useful facilities for guiding the implementation projects and assessing their level of compliance, including the Model Library facility and the definition of how principles are applied in the context of the solution architectures.

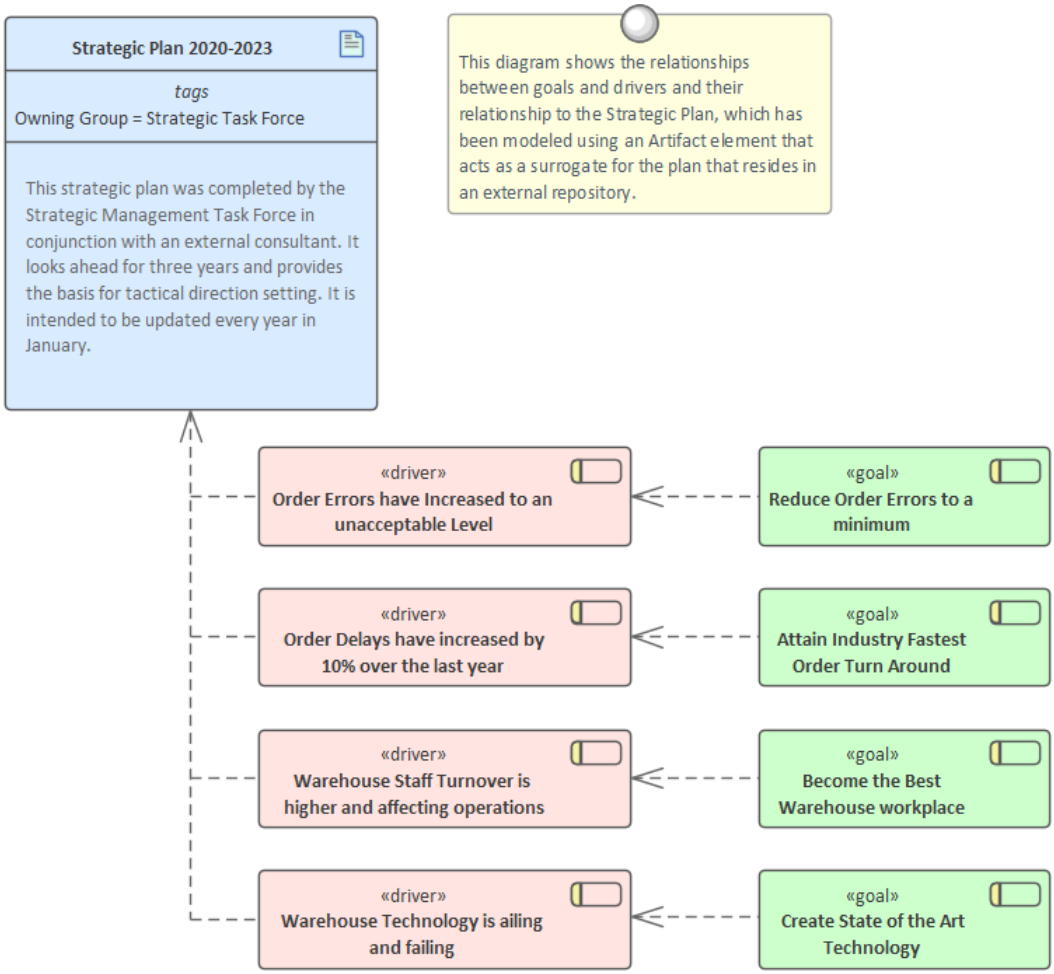


Strategic Context

The Business Architecture must be related back to the Strategic Plans to ensure that all the other architecture domains that describe the architecture in more detail are ultimately going to benefit the enterprise and deliver value. Business Architects typically gather information from the company strategists and should be privy to the high-level discussions and decisions about the future plans for the enterprise and its organizations.

Enterprise Architect has a number of tools that help to ensure that the architecture's alignment to the strategic plans can be visualized, and that the drivers forming part of the Business Architecture are related back to elements of the

plan. The business plan will typically reside in a corporate repository but a hyperlink can be created from an Artifact within Enterprise Architect that will launch the Strategic Plan or other documentation. Elements of the Business Architecture can be related to the Artifact that acts as a surrogate for the external plan.



Implementation Context

The purpose of the architecture is to ensure that initiatives and projects deliver the business value and benefits that have been described in the architectural models, so

monitoring the compliance of the implementation projects is critical to the success of the architectures and ultimately the architectural program. Implementation Governance is a key part of the architectural process and needs to be formally managed to ensure that the architecture acts as a guide for the implementation teams, but also to ensure that the architecture is clearly understood and followed. The Implementation initiatives are ultimately what transforms an organization from a baseline (current) state to a target (future) state, and ensuring that these initiatives comply with the principles and the designs is critical to the success of the program.

Enterprise Architect has a number of tools to assist with the governance of Implementation initiatives, including the Formal Review facility that can be used to conduct one-off or repeating reviews of projects, ensuring that their level of compliance can be determined. The fact that Strategic, Tactical and Solution Architectures and implementation projects can be managed in the same tool makes the governance process more streamlined. Even if the Enterprise Architecture and the implementation projects are located in separate repositories, content can be imported into the architecture repository for the purpose of making the assessment. Examples of how the principles are applied in the context of each initiative can be modeled using Instances of Principles, providing a useful guidance for implementation teams.

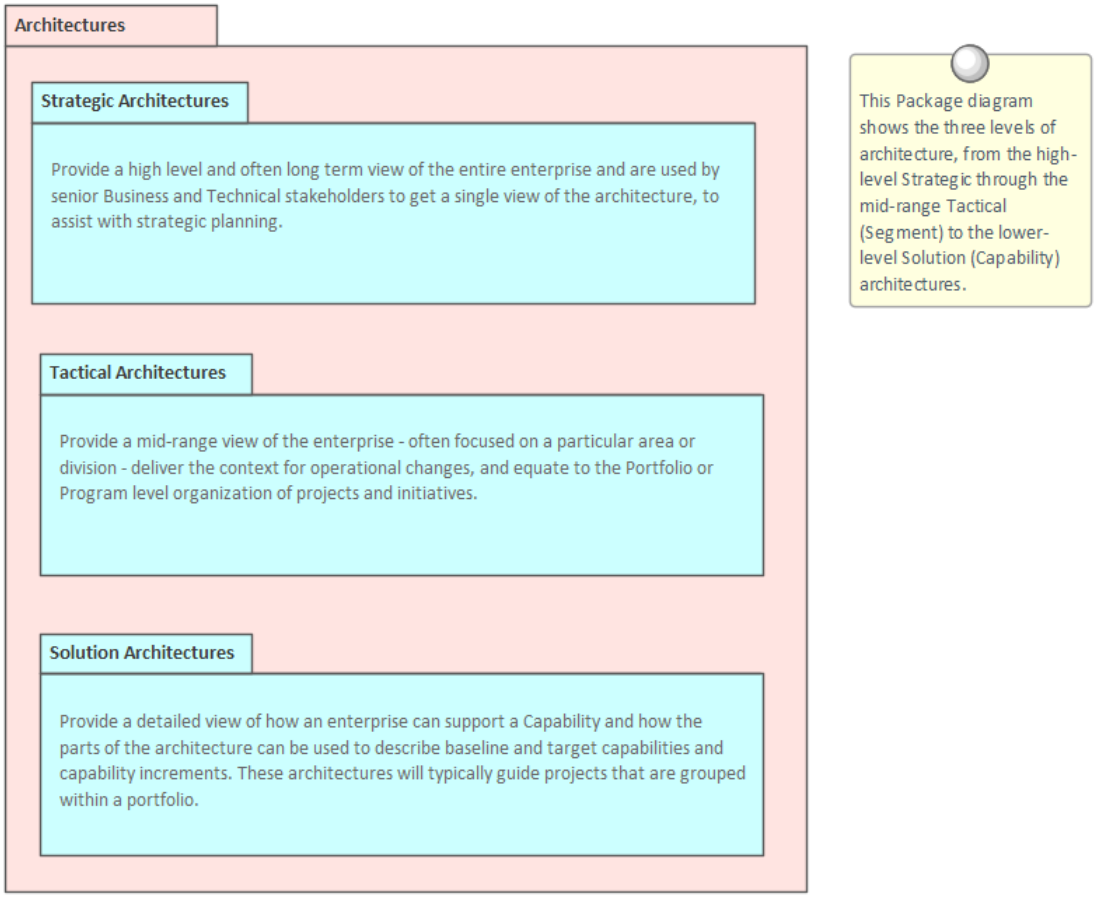
Levels of Architecture

An Enterprise has a complex and typically hierarchical structure, and you will need to create architectures at discrete levels of this structure. This hierarchy of architectures is analogous to the hierarchies of goals and capabilities, and intuitively aligns with Strategic, Program and Project level divisions. In a small organization it might be possible to create a single architecture that covers the Strategic level and the Project or Capability levels, but in an enterprise of any appreciable size at least three separate levels are typically needed. The naming of the levels has been influenced by The Open Group Architecture Framework (TOGAF).

- Strategic - Long term in the range of 3 - 5 years
- Tactical - Mid term in the range of 1 - 2 years
- Solution - Short term in the range of 6 - 12 months

The different levels of architecture will address different levels of concerns and have different audiences. The architecture framework and the repository must assist in ensuring these architectures are leveled and synchronized so that they form a cohesive and balanced view of the entire enterprise.

Enterprise Architect has a number of useful features that will assist the architecture program to partition and maintain these levels of the architecture and their inter-relationships.

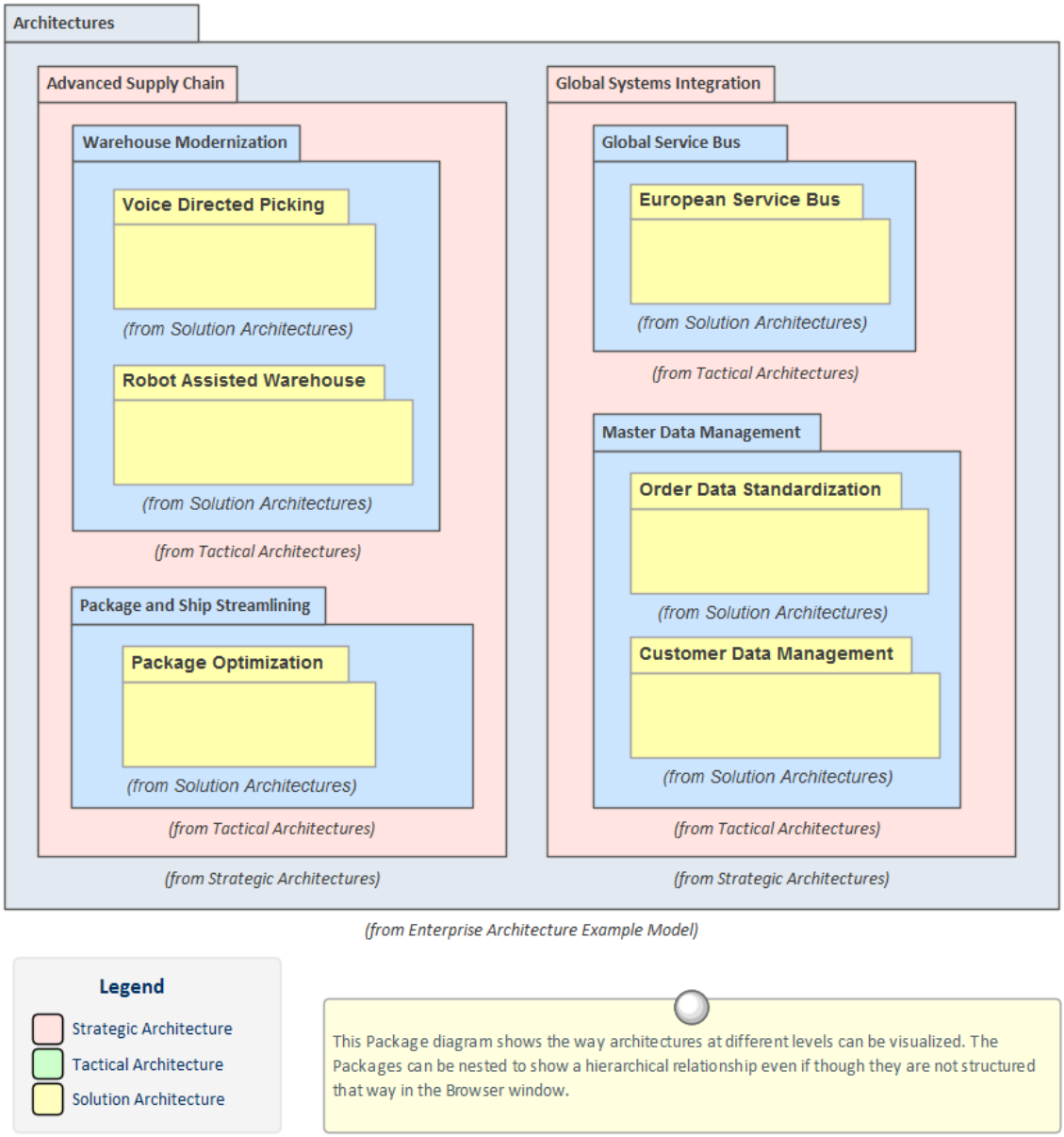


Strategic

Strategic architectures describe strategic plans and initiatives and typically run for years rather than months. A Strategic architecture will provide a long-range plan that is typically a view of the future over a three to five year period; this period can be longer for industries or enterprises that are not affected by dynamic and disruptive environments. The Strategic architectures must support (or align with) the enterprise's strategic goals, and Tactical and Solution architectures must help achieve the Strategic architecture or run the risk of not being funded.

Enterprise Architect can be used to define and manage

Strategic-level architectures, and can also help to ensure that Tactical and Solution architectures are aligned to support the strategic direction. The Strategic Modeling technology has a number of tools that can be utilized, such as the Balanced Scorecard diagram that can help to identify goals related to Information Technology. There are a number of tools, including the Relationship Matrix, diagrams and the Traceability window, that can be used to show the relationships between elements of the Business, Information, Application and Technology architectures and to ensure that they all demonstrably contribute to the achievement of the strategic goals.

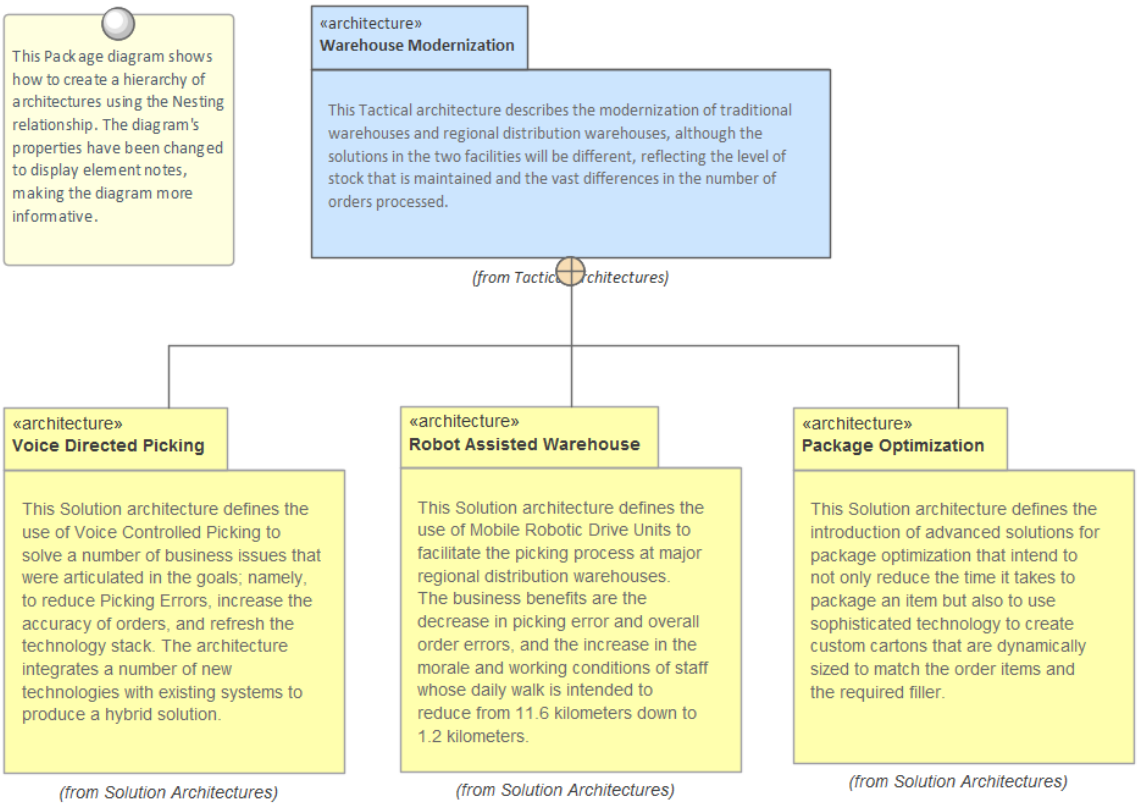


Tactical

Tactical architectures describe mid-range plans that help to partition the Strategic-level architectures into manageable groups. They typically could run for a number of years and represent a portfolio or program-level plan of how to achieve the goals expressed in the strategic architectures

they relate to. They act as a framework for organizing solution-level initiatives and ensuring that capabilities are developed that ultimately create business value.

Enterprise Architect has tools to support the Tactical-level architectures and to ensure that the partitioning of the Strategic-level architectures - and in turn the partitioning to Solution-level architectures - can be visualized. Roadmap diagrams can be used at all levels of the Tactical architecture, including Business, Information, Application and Technology architectures, showing the time sequencing of the initiatives at a portfolio or program level.



Solution

Solution architectures describe specific project- or capability-level initiatives that can typically be completed in months rather than years. From a business perspective they are usually focused on a particular problem or opportunity. Similarly, from a technical level they usually involve a slice through Information, Application and Technology areas, but might in some circumstances require a number of these to be addressed.

Enterprise Architect has a rich set of tools that can assist at the Solution architecture level, from defining the business goals and objectives and relating these to information and application components, to the technology devices that underpin the applications. Business Architecture can be defined and managed using stereotypes and profiles to create representations of the Business Drivers, Goals and Objectives, and these can be demonstrated to stakeholders using diagrams, matrices and documentation published automatically from the models. Tools such as the Schema Composer, Database Builder and UML Class diagram will assist with working with the Information architecture, and the elements created can be related to the Business architecture. Application Services, Applications and Interfaces can be modeled, and their relationships to each other and to elements of the Business and Technology architectures can be defined and presented in a variety of visually compelling representations such as diagrams, matrices and lists. The technology services and technology nodes and devices can be managed and, where applicable, they can be derived from the Technical Reference Model.

Enterprise Architect can also be used as the platform for Architectural Requirements Management, which can be related to elements that make up the Business, Information, Application, Technology and other specific architectures. The effective and flexible Kanban facilities can be used to manage these projects and ensure that the business value is delivered in a timely manner.

Types of Architecture

The overall architecture of an enterprise can be described by four integrated sub-architectures. These are:

- Business architecture
- Information architecture
- Application architecture
- Technology architecture

Most frameworks describe analogous or similar subsets of an Enterprise Architecture, as the division is based largely on organizational units performing work in these areas. There are a number of other architectures that could best be described as views, as they cross-cut and are contained by the other architectures, but because of their importance they are often raised to the level of an architecture. These include:

- Security architecture
- Geospatial architecture
- Social architecture

The seven types of architecture are described in this section.

Business Architecture

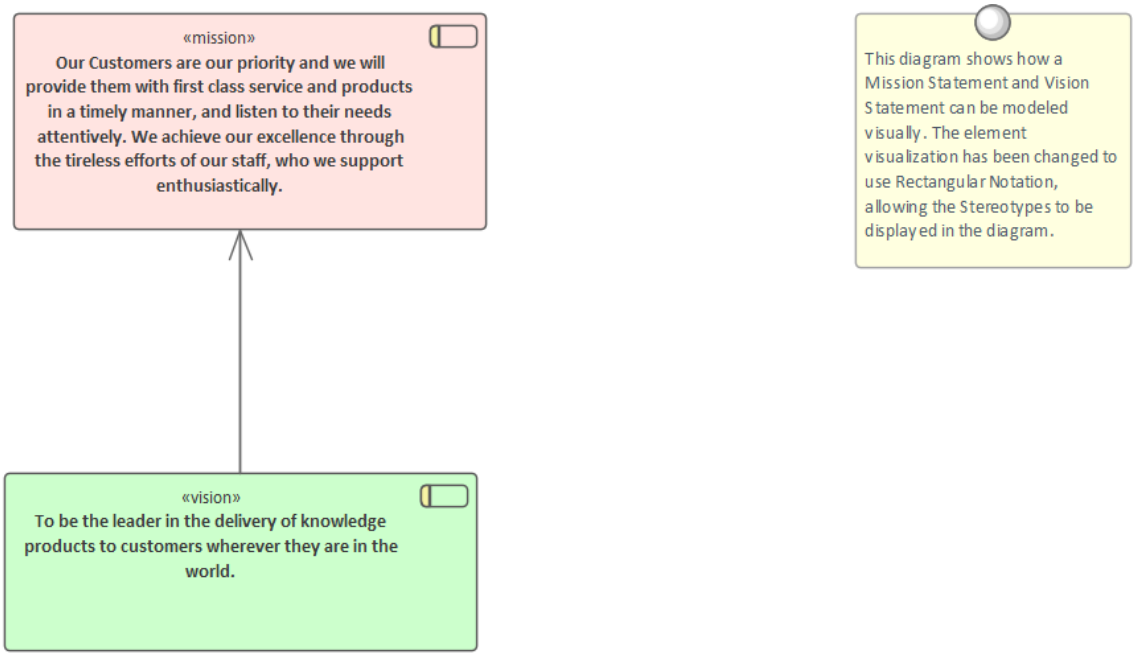
A well articulated Business architecture is the cornerstone of a successful outcome of the overall Enterprise Architecture. It defines the business drivers, the business strategy,

operational models, goals and objectives that the organization needs to achieve to transition in a potentially competitive and disruptive business environment. Architects working in the other architecture disciplines need to understand the Business architecture as the basis for their own architecture descriptions and as a guide to the business outcomes that have to be achieved.

The Business architecture will typically consist of a description of the baseline and target architectures, and definitions of a series of transitions that can be executed and that would be described on Roadmap diagrams.

Enterprise Architect has a wide range of tools that can be used for modeling Business architecture from the Strategic level down to the Operational level. Even when Strategic Plans reside in a corporate document repository, they can be represented and launched from inside Enterprise Architect. This allows other elements such as Drivers, Goals and Objectives to be traced back to their origins in the strategy documents. Mission and Vision statements can also be modeled alongside Business Capabilities, Processes and Functions. Stakeholders and Organizational Roles can be represented and used to show who owns elements of the architecture, such as requirements and applications. Business and Stakeholder Requirements can be created and managed at any level of detail, and the Document Generator can be used to create elegant and comprehensive publications that describe aspects of the Business architecture.

Learn More: Business Architecture



Information Architecture

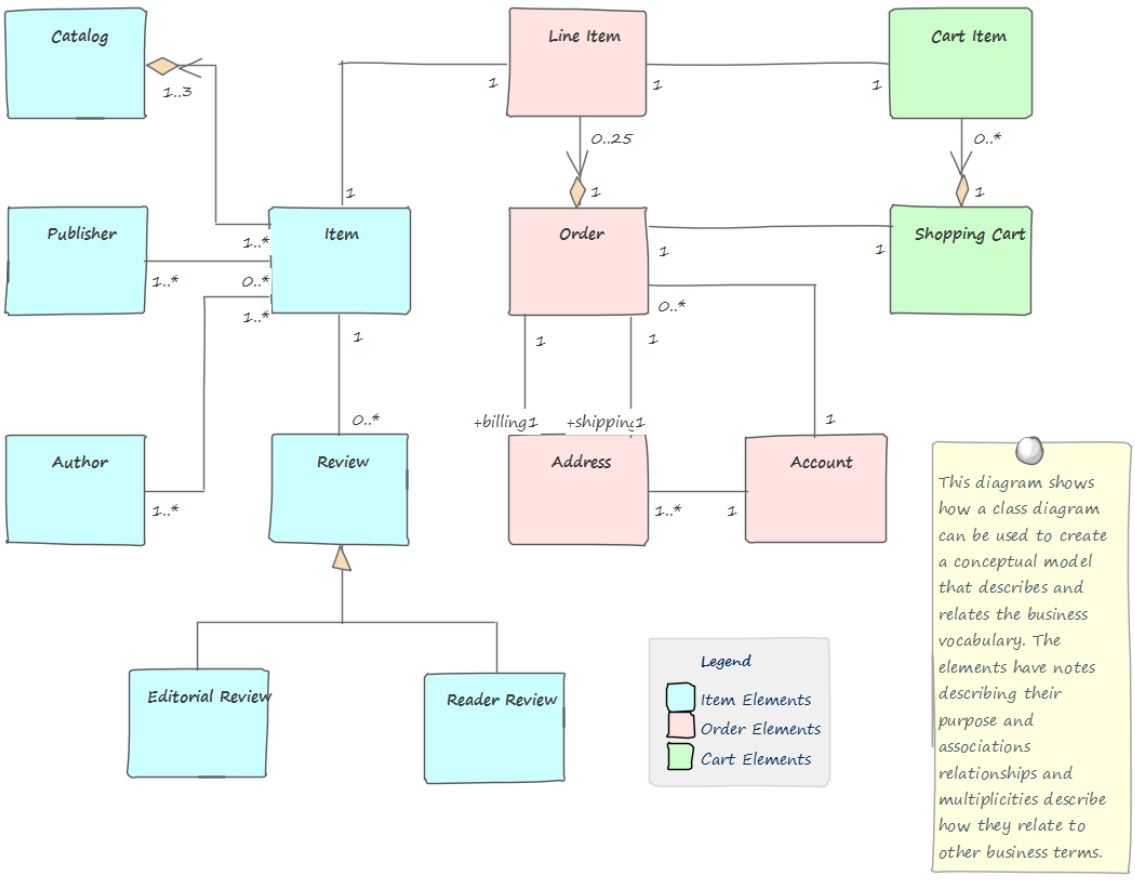
Information Architecture is key to the success of an Enterprise Architecture Program, as information is created, consumed and destroyed by the components that make up the other architectures. Understanding which business functions and processes utilize information, which applications serve as the master record, where information is created and destroyed, and which technology components store and manipulate the information, is critical to achieving the business outcomes.

The information architecture will typically consist of a description of the baseline and target architectures, with a series of transitions defined that can be executed and that would be described on Roadmap diagrams.

Enterprise Architect is a profoundly useful tool for creating

and maintaining information architectures, with its sophisticated and extensive support for standards and its wide range of tools to support information models, from high level classifications and concepts right down to the level of schemas and the elements and columns they are composed of. Tools such as the Schema Composer and the Database Builder, along with the Unified Modeling Language (UML) Class diagram and Glossary, and the Model Transformation facility, will be invaluable.

Learn More: Information Architecture

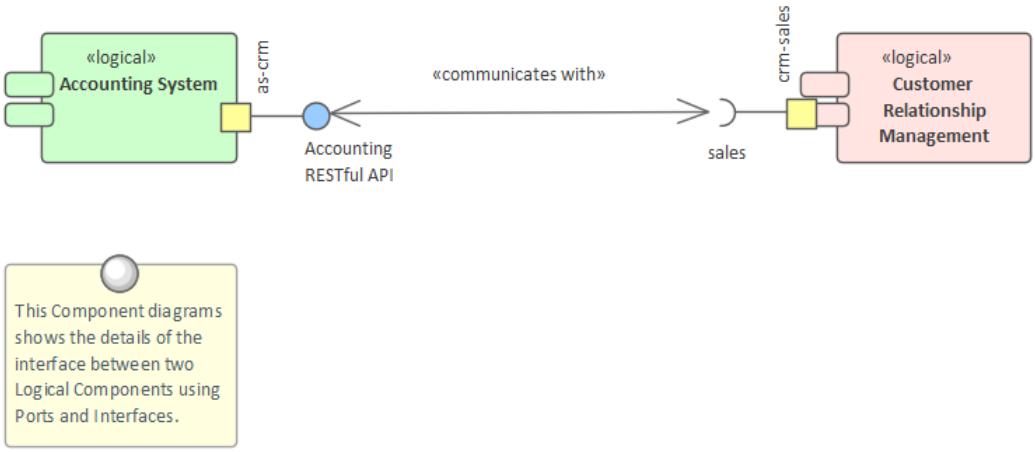


Application Architecture

The application architecture provides an important catalog of the applications in the enterprise, describing the work that they do to transform, transmit and store information. The architecture also describes the interfaces that are required or provided by the applications and the way the applications interact to carry out the activities described in the business models, such as the Business Process diagrams and Capability models. The catalog of applications, interfaces and the diagrams and matrices that describe their interaction only need to be defined once at the enterprise level. An application architect will be able to draw upon this inventory of existing artifacts to create new architectures, classifying them as part of the baseline and potentially the future state architecture. Where an architecture introduces new applications, these can be added to the description of the target state.

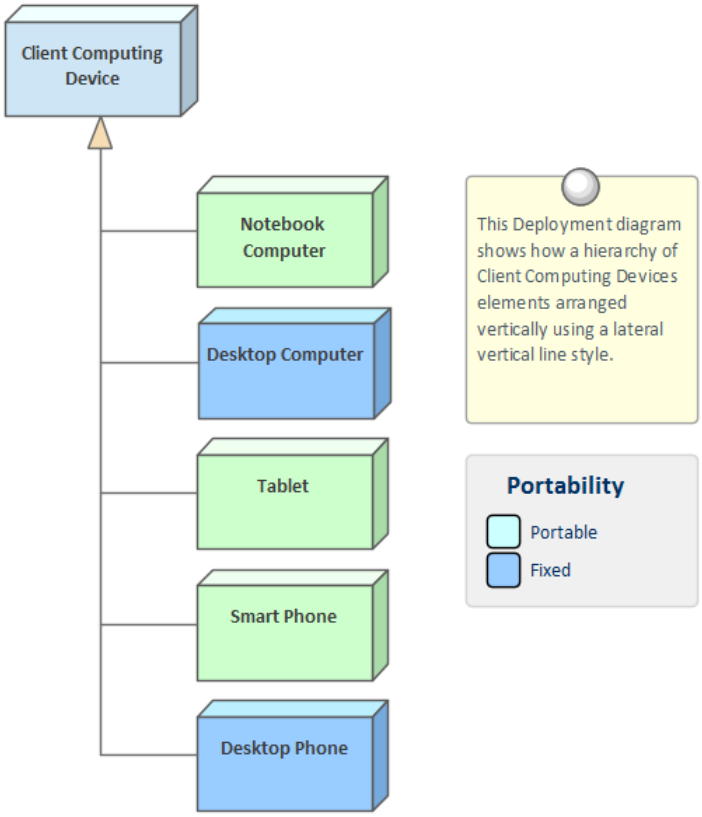
The application architecture will typically consist of a description of the baseline and target architectures, with a series of transitions defined that can be executed and that would be described on Roadmap diagrams.

Learn More: Application Architecture



Technology Architecture

The technology architecture underpins the other architectures, providing a description of the logical, physical and virtual infrastructure that supports the execution of application services that in turn support information and business functions and services.



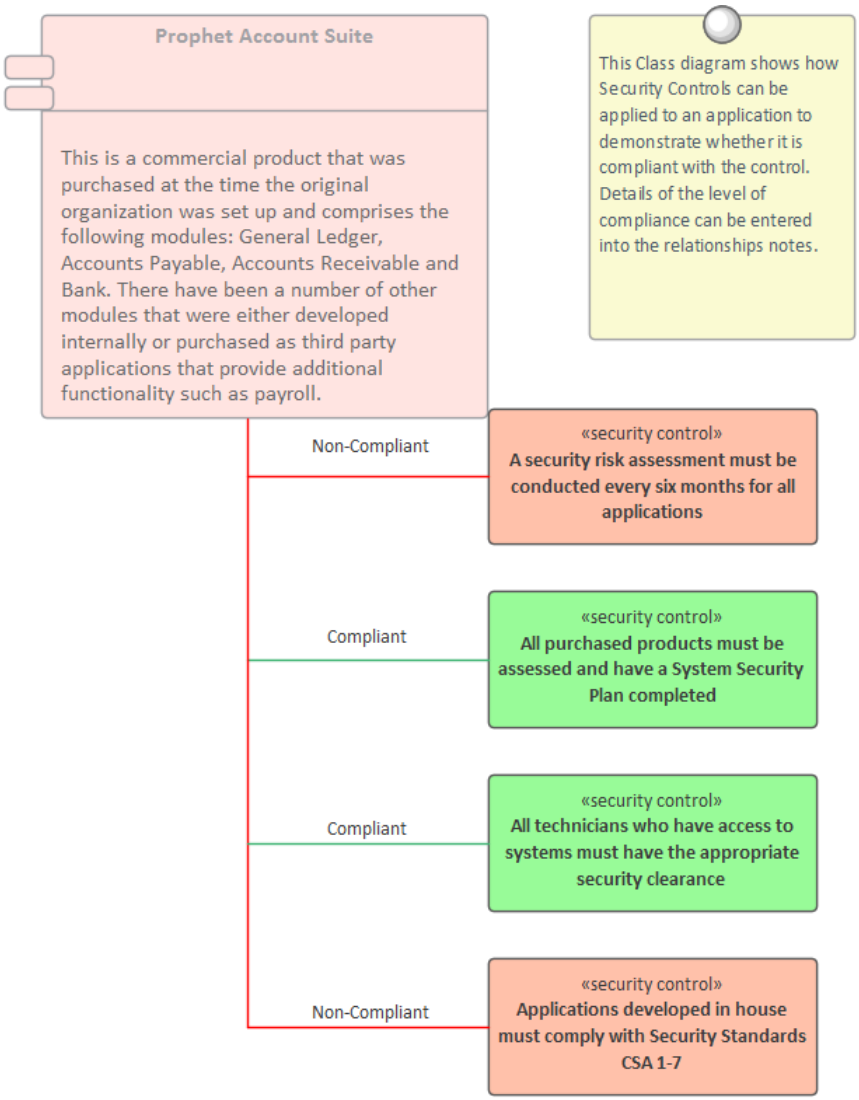
Learn More: Technology Architecture

Security Architecture

The security architecture is a slice through all of the other architectures from a security viewpoint. It is listed as a separate architecture because of its importance in ensuring that the enterprise security polices are implemented through the architecture. A breach of security could occur at any point from the business architecture through to the technology architecture. This could include demonstrating how the architectures comply with security controls published by the enterprise or available as part of an

industry compliance regulation.

Enterprise Architect can be used to model the security controls maintained as a list of elements in the repository and when new architectures are created the applicable controls could be applied to aspects of the architecture such as applications, technology devices, communication paths and more. Significant reuse can be achieved by the use of Patterns, and the use of existing building blocks that have already been certified as compliant with the controls.



Geospatial Architecture

The Geospatial Architecture is a slice through all of the other architectures from a geo-spatial or location-based viewpoint. It is listed as a separate architecture because of its increasing importance in a world dominated by location-based applications and business technology functions. Not every architecture program will have the need to develop separate geospatial architectures and in these cases it can be relegated to a view of the other architectures.

Enterprise Architect is well placed to be a repository for geo-spatial architectures, with its integration with some of the leading standards and tools in the market place, including support for the Geographic Markup Language (GML), ISO 1900 series standards and tools such as ArcGIS and other geodatabases.

Social Architecture

Social Architecture has been born in an era that depends on social media and social behavior to achieve a wide range of outcomes in built and digital environments. Increasingly in this period of digital and social disruption, understanding the social aspects of an organization's interaction is critical.

This can include the way it interacts with its customers, suppliers and observers and other individuals and communities through what is commonly described as social media.

Enterprise Architect can be used to model the social context of the organization and if significant information is published to social media sites or information is gleaned from these sites these can be modeled in the repository and an architect can create visualizations that show how this information relates to other parts of the business and information architectures.

Scope of Architecture

Architectures will only be successful if they are scoped correctly. The Enterprise Architecture Body of Knowledge (EABOK) describes three important aspects of scope, but another one can be added that addresses the importance of the stakeholders in the success of the architecture program and the architectures it creates and manages:

- Time Scope
- Organization Scope
- Detail Scope
- Stakeholder Scope

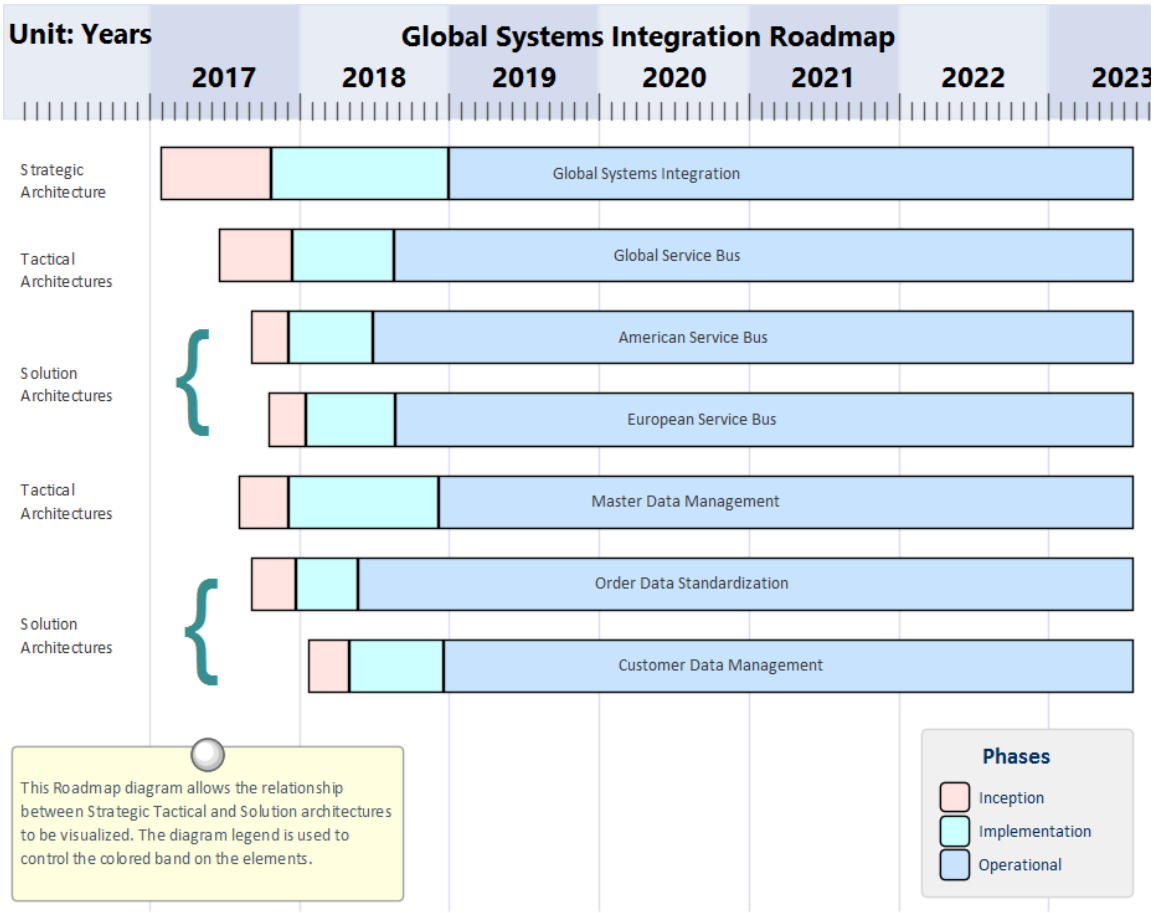
Architecture time frames, organizational context, levels of detail and appeal to stakeholders must all be appropriately set for the architecture to be relevant and successful. Enterprise Architect has tools that support all of these types of scope, from the Roadmap overlay feature for time modeling, the Organization chart to show which parts of the enterprise will be affected, and the wide range of diagrams and matrices that can be used to allow stakeholders to visualize the architectures at the appropriate level of detail.

Time Scope

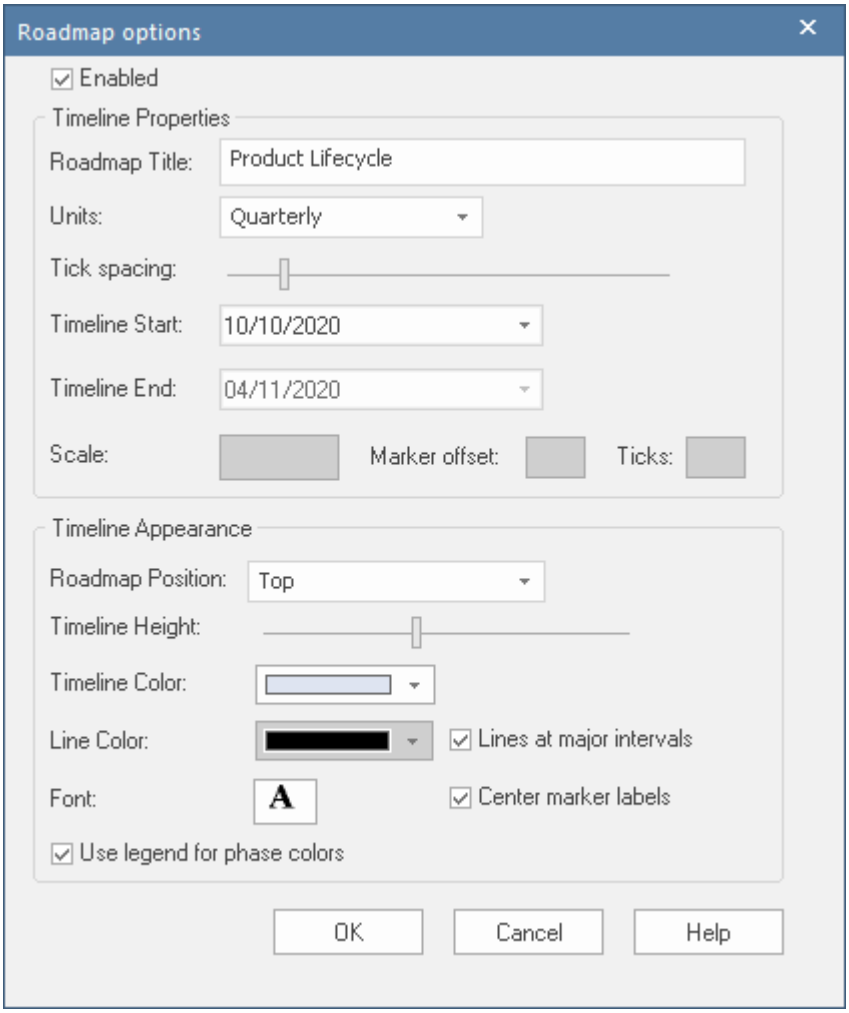
Time Scope is important because the business typically works in cycles, and it is critical that the architectures respect this time dimension of the organization's

management and operation. Strategic plans for medium to large enterprises typically span a period of between three and five years, so it is common for strategic architectures to describe a similar time period, while implementation projects typically run for a period of between three and twelve months. Tactical architectures that group a number of implementation projects can span a period of between one and two years.

Enterprise Architect has some useful features that can help with managing time, including the Roadmap overlay that allows a time scale and extent to be defined and that can indicate the phases any element passes through set against the backdrop of that timescale. The tick spacing can be set from days up to years, allowing any time extent to be represented. Any architectural elements can be represented on Roadmap diagrams, including architectures themselves, principles, capabilities, applications, information, technology devices and more.



The flexibility of the overlay allows any diagram to be converted into a Roadmap, and there is a wide range of settings that can be used to configure the visualization of the Roadmap, including the Diagram Legend, which can be used to define the segmentation of the elements into a series of phases. The Roadmap Options can be used to change the time scale from years, quarters, months, or days down to very fine graduations (used for engineering diagrams). The start and finish times can be set, and the scale can be changed to stretch or collapse the time scale. The position and height of the time ruler can be changed, and fonts and colors can all be configured to make the diagrams more appealing.

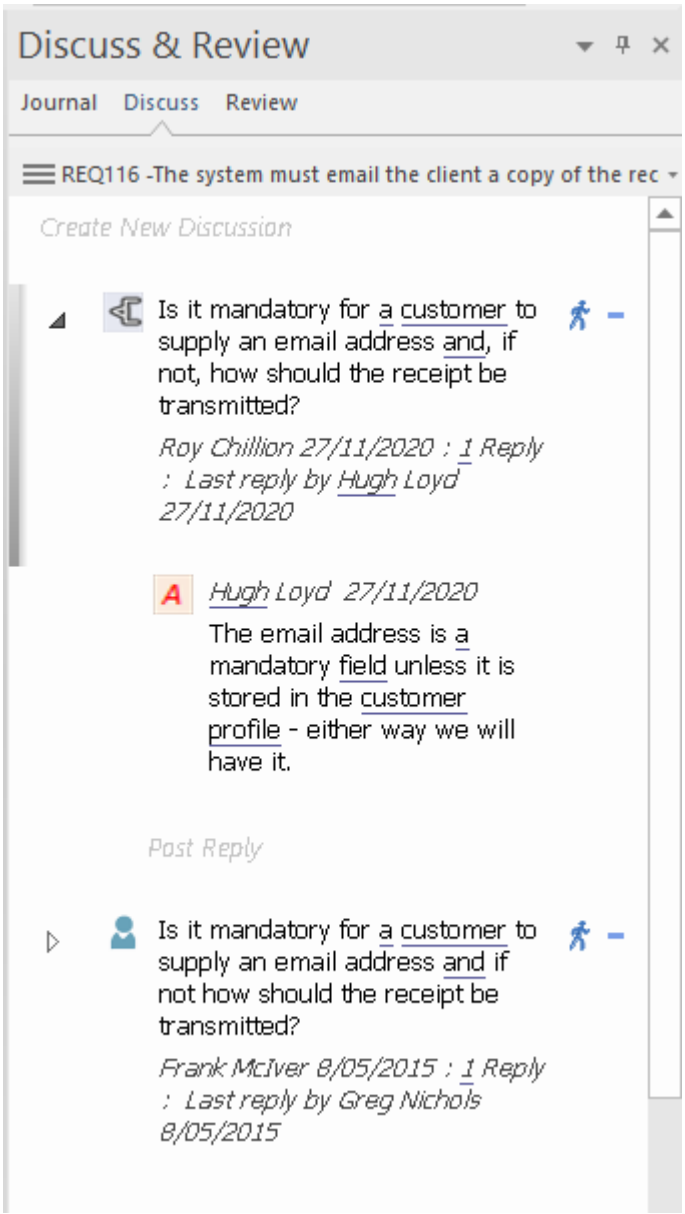


Detail Scope

Selecting the correct level of detail for an architecture is critical to its success; this is particularly true when it comes to the Implementation teams. Creating architectures that are too lofty or aspirational will result in Implementation teams making important design decisions themselves which, while they might be appropriate for their solution, might not be the best outcome for the entire enterprise. On the flip side of this argument, creating architectures that are too prescriptive and detailed can constrain an Implementation team and

result in the team not having the flexibility to select the best solution.

Enterprise Architect is a tool based on the concept of collaboration, and there are many facilities that will help the architecture team members work with each other and with all stakeholders, including implementation teams, to determine the most appropriate level of detail for the architectures. The Model Library facility allows in-model reviews to be created, where elements from the architectures such as Goals, Objectives, Applications, Technology nodes and more can be dragged in as references for the reviews. The Discuss & Review and Discuss & Review - History windows allow architects and stakeholders to deliberate about the architectures and the consequent implementations. The Diagram Filtering facility and a wide range of tools for changing the visualization of elements in diagrams allow the appropriate level of detail to be set for the architectures and the views that are created for stakeholders.

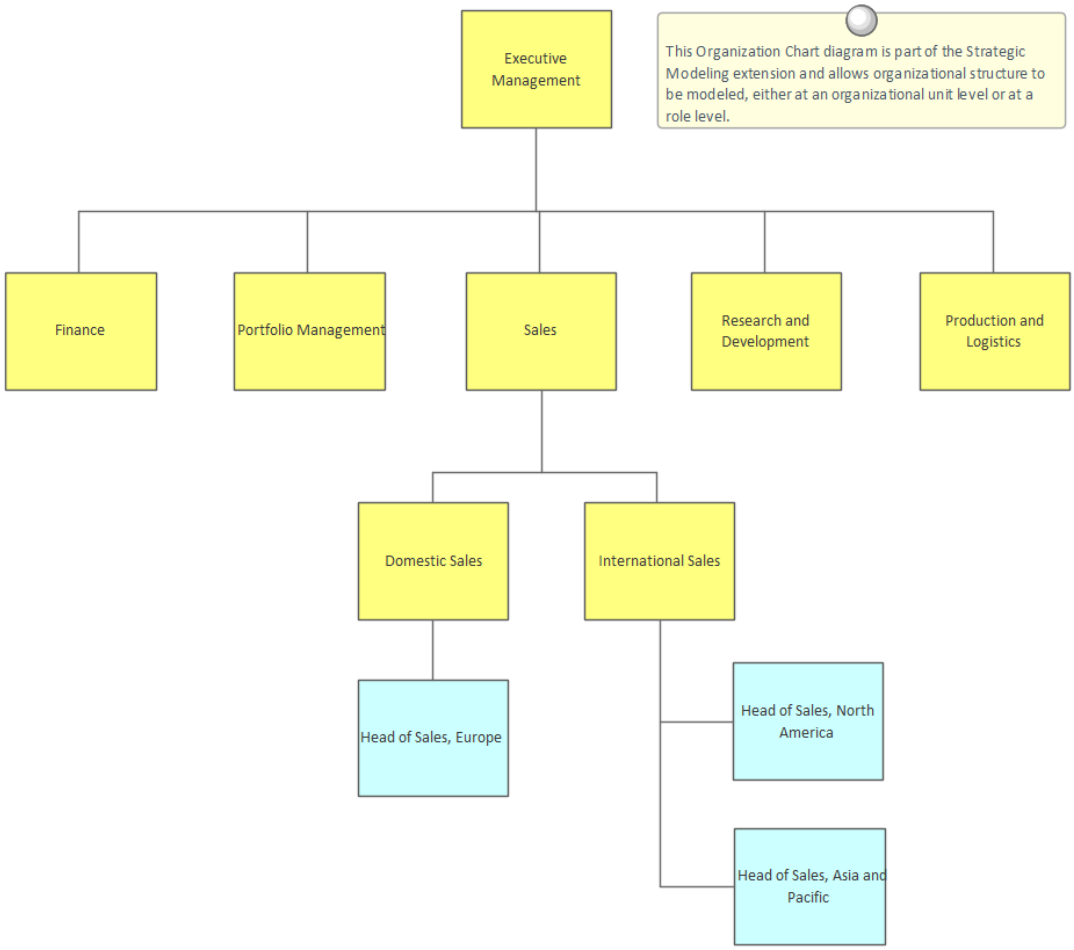


Organization Scope

Enterprise Architecture is a non-trivial and costly discipline, and it is critical that value is delivered to the business. The best outcomes will be achieved if the architecture touches all parts of the enterprise, but it is quite common for some parts of the enterprise to receive greater emphasis in the

architectural descriptions than others. Having a clear understanding of the structure of an enterprise and its organizations and how strategic plans relate to this structure is critical for the success of any enterprise architecture effort.

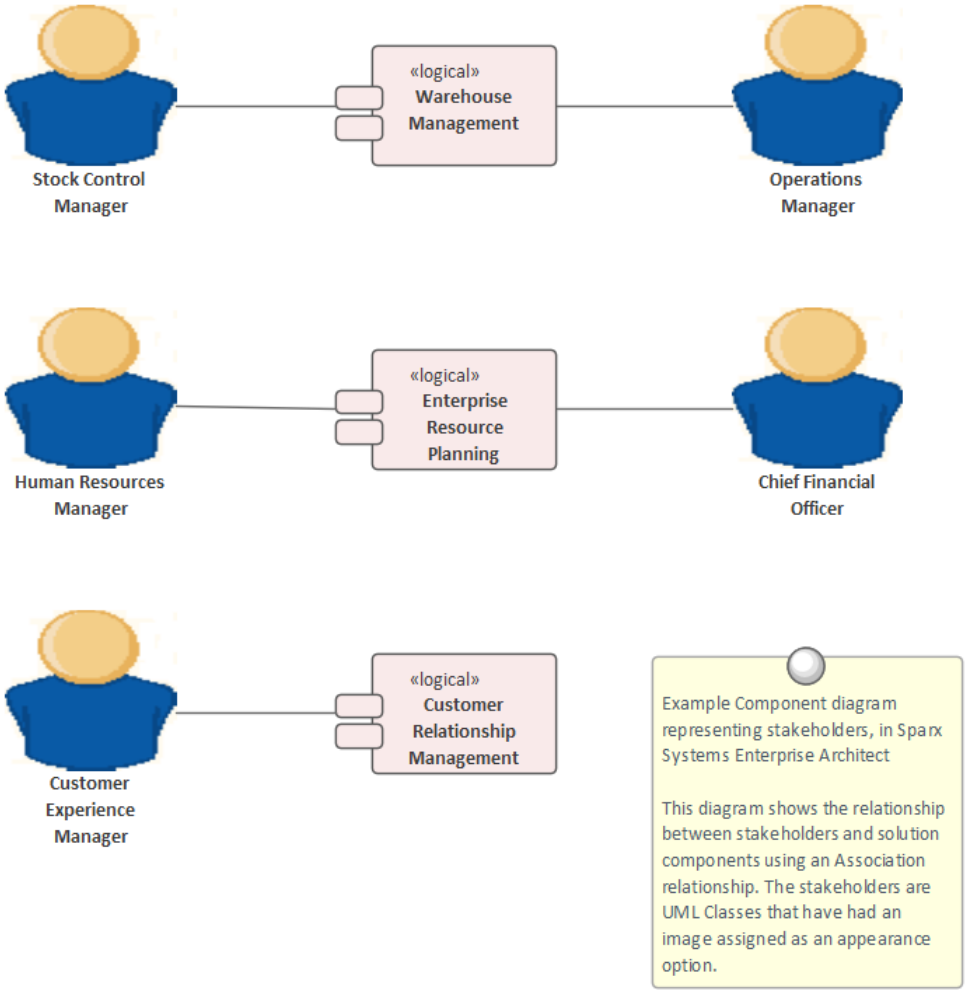
Enterprise Architect has an Organizational Chart within the Strategic Modeling Technology, which can be used to model the structure of an enterprise and its organizations. The architectures can be related to this structure, which allows the organizational scope to be visualized.



Stakeholder Scope

The stakeholders and the shareholders or organization owners that they represent are the ultimate beneficiaries of the enterprise architecture, and it is important that the right stakeholders are selected and communication is managed to ensure they are kept updated with the progress of the architectural work and the governance of the Implementation Initiatives.

Enterprise Architect has a number of facilities to ensure that the stakeholder scope is determined and that the architectures are created in accordance with what these individuals or groups require. The stakeholders themselves can be modeled inside the tool and their relationship to elements such as Drivers, Goals, Objectives, Applications and Architectural Requirements can be maintained. This allows impact analysis to be visualized, so that when changes occur that affect any of these elements, the stakeholders who have an interest in the change can be determined. The visualization can be through diagrams, matrices or lists of elements and can be viewed directly in the model, or publications can be generated in a variety of formats including PDF, DOCX and Web Pages.

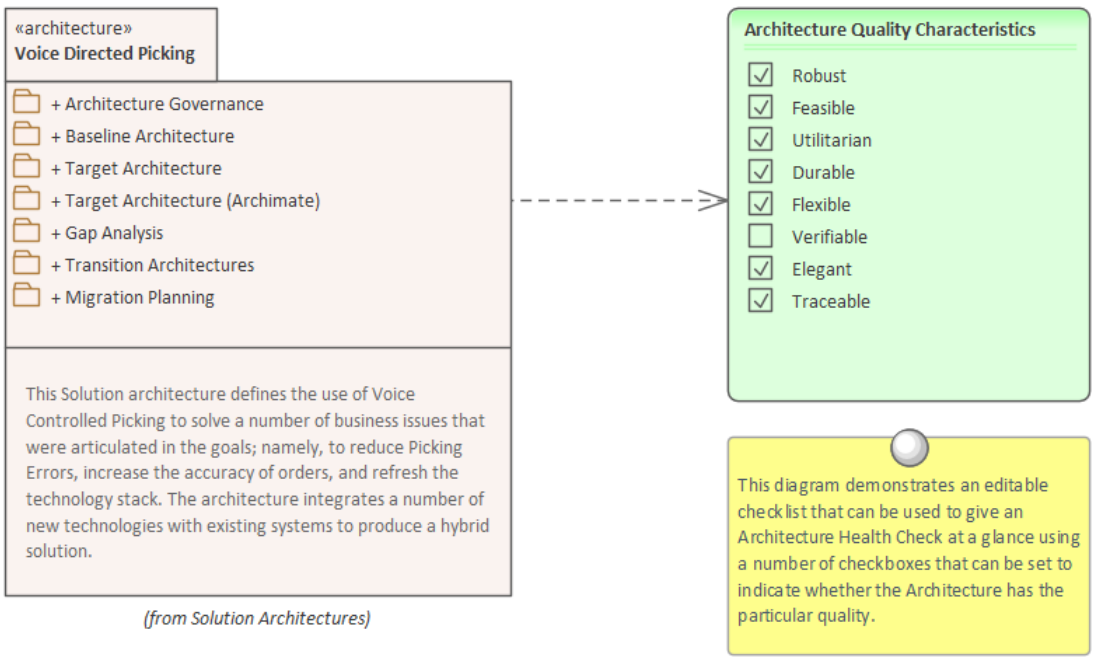

«logical»
Warehouse
Management
Stock Control
Manager
Operations
Manager
«logical»
Enterprise
Resource
Planning
Human Resources
Manager
Chief Financial
Officer
«logical»
Customer
Relationship
Management
Customer
Experience
Manager
Example Component diagram representing stakeholders, in Sparx Systems Enterprise Architect
This diagram shows the relationship between stakeholders and solution components using an Association relationship. The stakeholders are UML Classes that have had an image assigned as an appearance option.

Characteristics of Good Architecture

It is difficult to define what the characteristics of good architecture are when there is still vigorous and lasting debate about what the term 'architecture' actually means in the context of enterprise systems in the Twenty-First Century. The Roman architect Vitruvius defined three characteristics of good architecture in his treatise *De Architectura* more than 2,000 years ago. Interestingly it is the only surviving text from antiquity describing architecture. These principles are:

- Durability (Firmatis) – It should stand up robustly and remain in good condition
- Utility (Utilitas) – It should be useful and function well for the people using it
- Beauty (Venustatis) – It should delight people and raise their spirits

These ancient characteristics can be elaborated on and expanded to apply to the Enterprise Architectures that are developed in the Twenty-First Century.



Qualities of Good Architecture

To be effective, an architecture must have a number of qualities or characteristics. Enterprise Architect provides an extensive set of features and tools for helping the Architect produce architectures that are of high quality. This table contains some of the most important qualities, with a description of how Enterprise Architect can be used to ensure the qualities are built into Architecture created and maintained in the tool.

Quality	Description
Robust	An architecture should be strong and not be vulnerable to minor changes in the

	business, information, application and technology systems. Enterprise Architect can assist in ensuring that the architectures are well integrated and related to each other and provides a number of tools such as the Traceability window, the Relationship Matrix and the Insert Related Elements feature that can be used for this purpose
Feasible	An architecture that cannot be implemented will mean that the goals and objectives of the enterprise will not be met. It is best to identify these requirements as quickly as possible so as not to disappoint the party who requisitioned the architecture work. Enterprise Architect can assist by allowing architects, designers and developers to discuss the architecture and determine its feasibility using the Discuss & Review window and by mapping the Enterprise Architecture to Capability or Solution Architectures.
Utilitarian	An architecture must have utility which, in turn, when implemented will result in practical outcomes. Architectures that are elegant but do not provide demonstrable and measurable value to the stakeholders

	or the parties that requisitioned them will ultimately not be successful. Enterprise Architect has tools that allow an architecture to be visualized and understood by a diverse group of stakeholders, allowing any problems with utility to be discovered early in the architecture process.
Durable	An architecture is a living entity that describes a target state and - once implemented - will become the new baseline state. The architectures should prove to be durable with the passage of time and be resilient to changes in the business and technical environments that might occur over the lifetime of the architectures. This implies that they must - as much as possible - pre-empt the future conditions and environments.
Flexible	The architectures must be flexible and be able to adapt to changing conditions and also provide enough guidance for implementation teams that have the knowledge of their discipline to make the important and necessary decisions about technical problems and opportunities. Architectures that are created with too much detail will often result in brittle and

	inflexible designs and implementations resulting in systems that cannot adapt to changing circumstances and environments. Enterprise Architect has a wide range of features that can assist with change including the Change element, the Baseline facility and the Kanban diagrams that allow Requirements Features, User Stories and more to be visualized and prioritized.
Verifiable	It should be possible to verify that the architecture will perform as designed and that there would not be side effects that result from the architecture and the parts of the enterprise that it impacts. The ultimate test of this is whether it delivers the business value that was promised in the Vision Statement. Enterprise Architect can be used to model the measures that are defined to verify that the Business Objectives (and therefore the Goals) have been attained.
Elegant	Architectures must have both form and function and it is a good test of an architecture to measure its elegance. An architecture that is well designed will

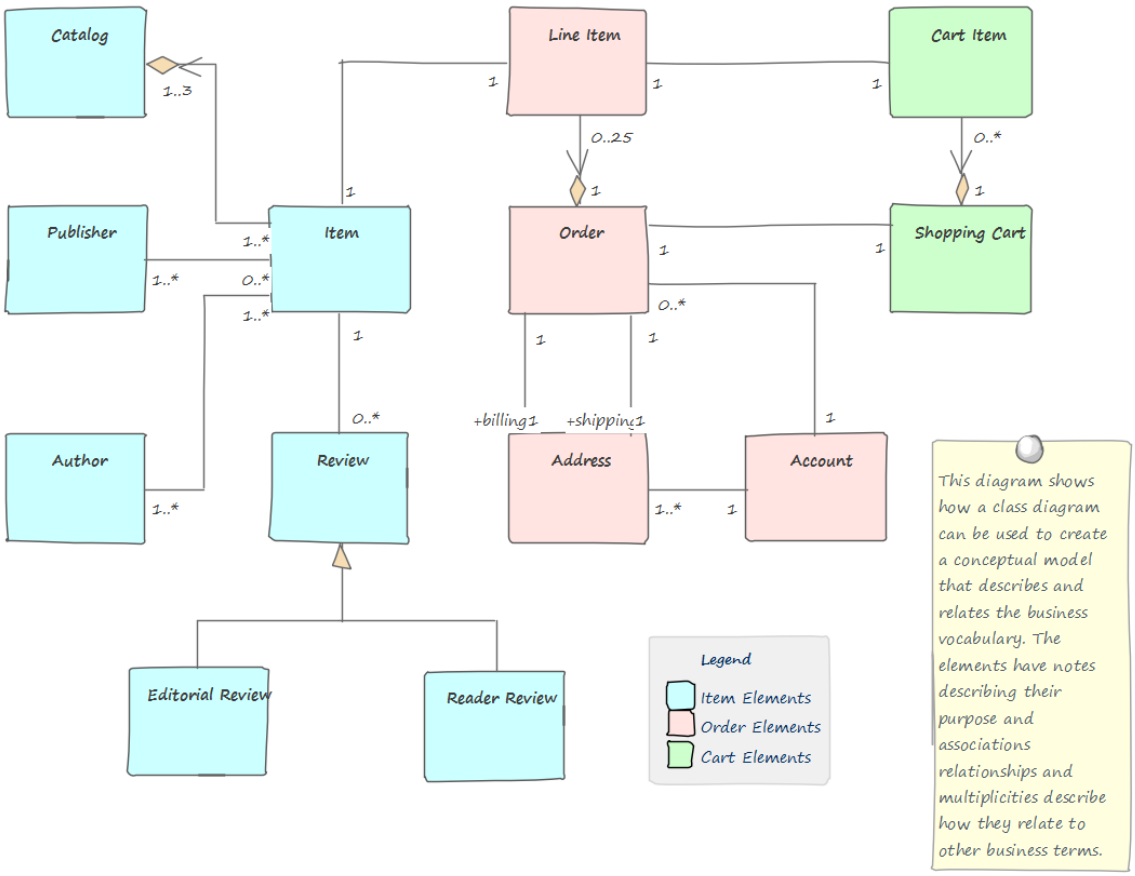
	tend to be elegant and have a simplicity of form that will be obvious to those that take the time study it. Enterprise Architect has extensive features that allow the elegance of an architecture to be visualized including the ability to create professional publications that can be generated automatically from the tool using a series of built-in or user defined templates.
Traceable	An architecture is a description of the an enterprise at a particular level of detail and does not exist in isolation but is typically related to business drivers and goals and other architectures at the same level or higher or lower and to implementation programs and projects. Enterprise Architect allows elements to be traced in any direction and provides a number of useful tools to visualize the traces including the Relationship Matrix, the Traceability Window and diagrams. The Insert Related Elements facility can be used to automatically construct a diagram of traces almost magically creating expressive and never before seen views of the repository.

Lists, Diagrams and Matrices

Lists, Diagrams and Matrices are the three main ways of presenting architectural information to stakeholders. These three representations can be used in isolation or together to provide a rich communication of the architectures; they can also be tailored to suit individuals or groups of stakeholders. They are often combined in repeatable sets called Views that provide consistent, coherent and relevant information to an audience.

Lists provide a simple catalog of items that can be displayed in a tabular format, where relevant properties and metadata can be viewed for each item and compared between items. Diagrams are a graphical projection of items connected as a graph and provide a compelling visual representation of the elements and their relationships. Matrices are grids that show the relationship between two sets of items from a particular point of view; they are an effective visual device because they allow gaps and overlaps to be identified easily.

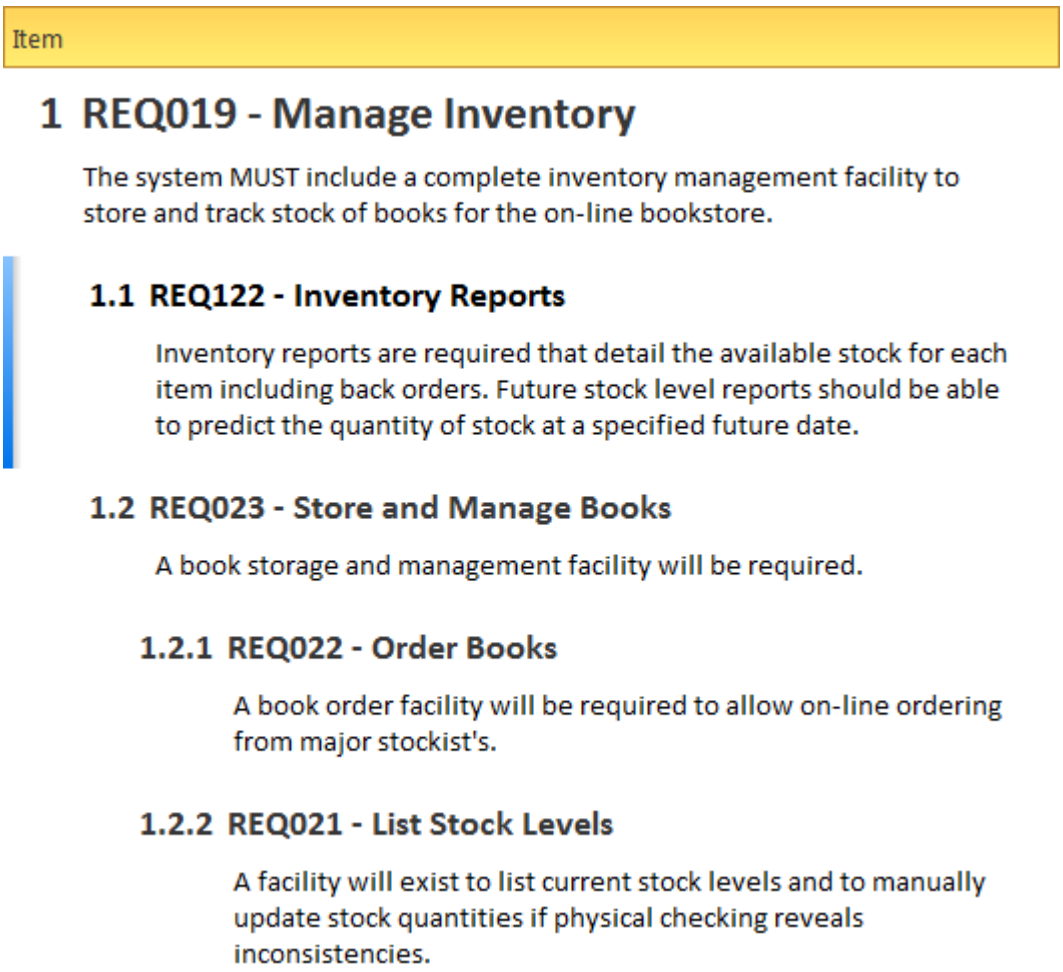
Enterprise Architect has a wide range of tools that support all three of these representations, with many extended features such as searching, sorting, layout, filtering, alternative images and simulations that allow an architect to create visually compelling representations of the architectural content.



Lists

Enterprise Architect has a number of tools for working with elements in a list. These tools can be applied to any type of element, including Principles, Business Drivers, Requirements, Applications, Interfaces, Technology Devices and more. The Specification Manager can be used to create and view any type of element in a visually appealing, word processor-type or spreadsheet-type format. The properties of the listed elements can be displayed and any number of properties can be added or removed from the display, including Tagged Values. The properties can be

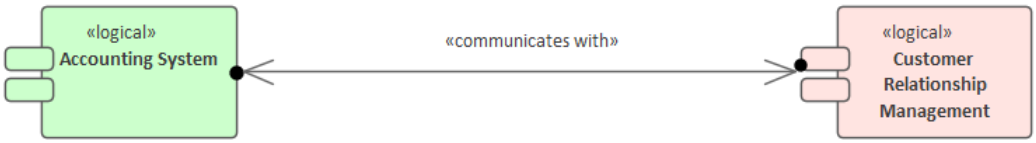
edited for each row of the list, including selecting values from drop-down lists such as selecting a status. Filters appear below the header of each column and can be applied to restrict the display to elements that meet a particular condition. Changing any of the details of an element, including its name and description, will change it in every other part of the repository, including the Browser window and all diagrams in which it appears.



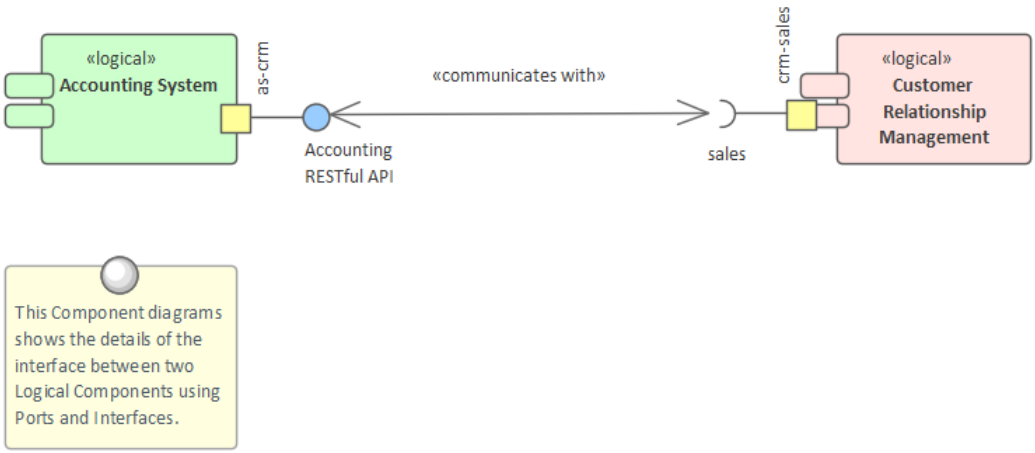
Diagrams

Diagrams are an expressive and visually compelling way of

presenting information and can be carefully constructed to deliver a particular message. Diagrams can be used to communicate with stakeholders to provide a view of the architecture from a particular viewpoint. The same elements can appear in multiple diagrams and color, styles, filters, layout, alternative images and more can be used to convey meaning and to create compelling diagrams that will help stakeholders engage with the architecture. The diagrams can be converted to a hand drawn style and a white board mode to create further appeal and to soften the edges of the formal modeling languages that can deter some stakeholders.

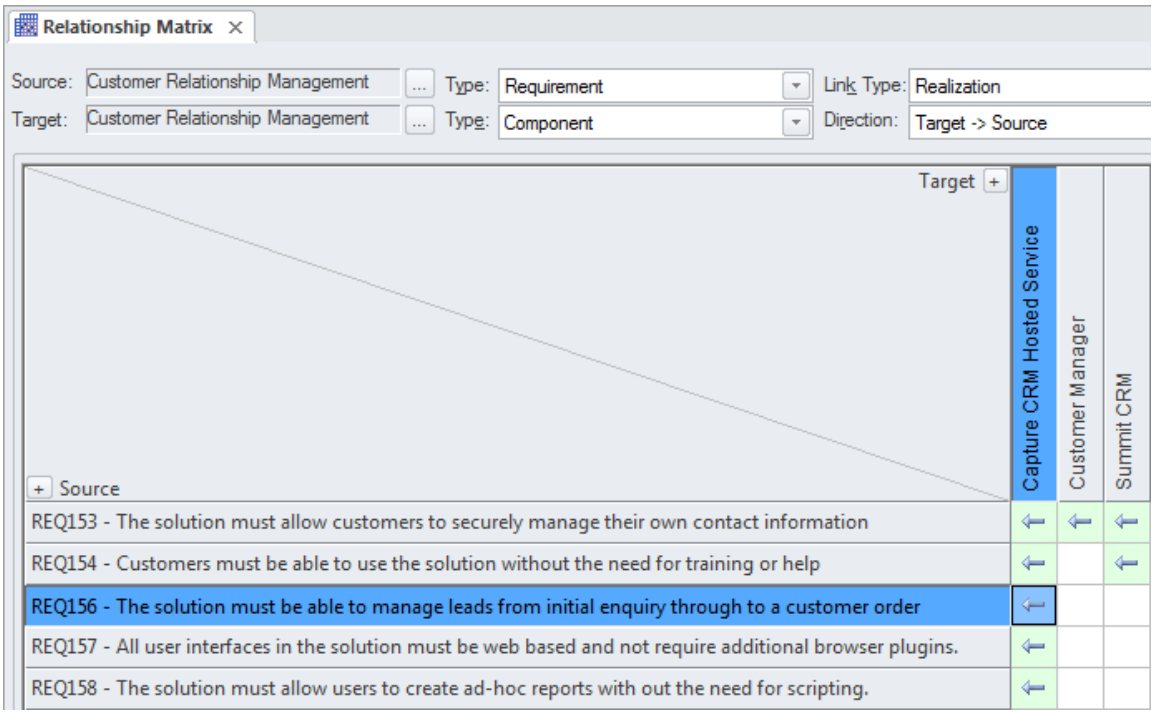


The same diagram can be altered to show the detail of the interface that has been rolled-up in the previous diagram. This facility that allows repository content to be automatically altered to create alternative views for different stakeholders.



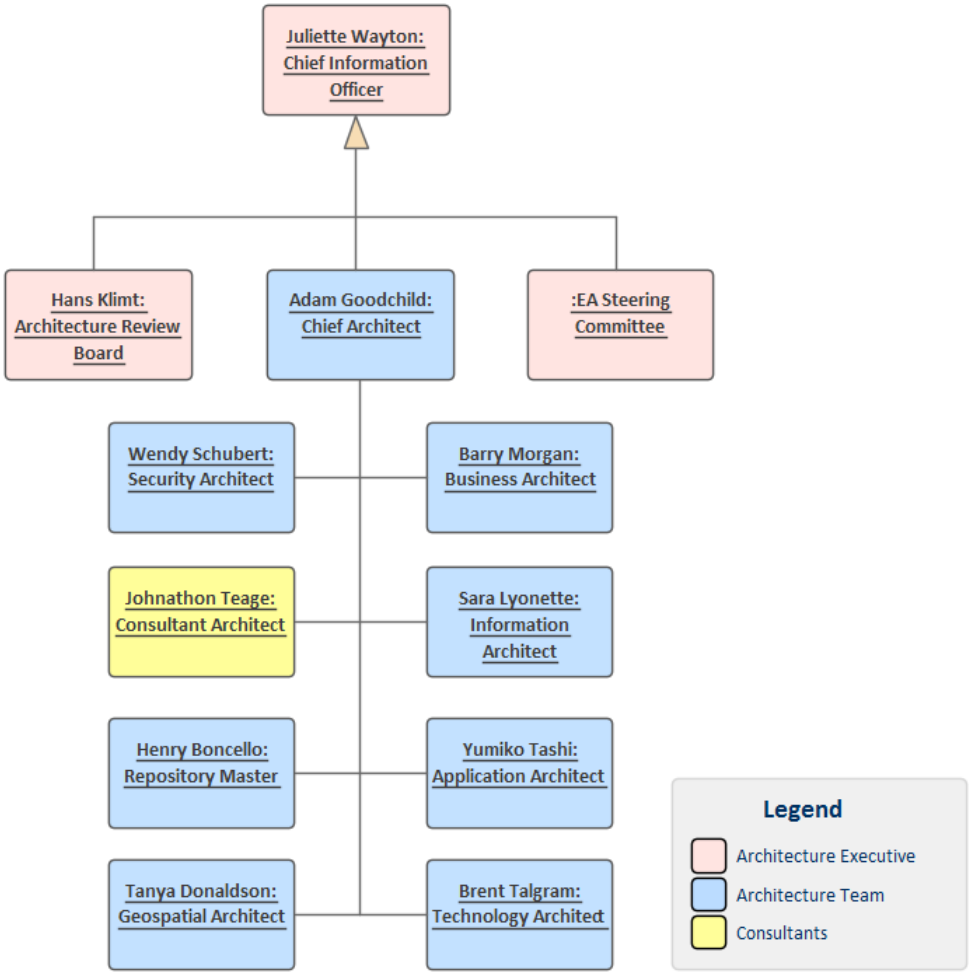
Matrices

The Relationship Matrix and Gap Matrix are grids that allow the relationships between two sets of element to be visualized in a matrix format with one set of elements on the horizontal axis and the other on the vertical axis. Markers at the intersection of a column and a row indicate information about the relationship between the two elements.



Developing an Enterprise Architecture

The development of an Enterprise Architecture is a complex and time consuming endeavor involving a multi-disciplined team of architects. Ultimately the architectures must deliver value to the business and this can only be achieved with the engagement of the stakeholders. Without a comprehensive stakeholder engagement and communication plan the architectures will inevitably fail to deliver the envisioned business benefit. The Enterprise Architecture should be a cohesive and well organized product that provides meaning for all stakeholders even though it comprises a number of interlocking and quite separately authored architectural contents. In this way the chief architect needs to act in an editorial role, ensuring that the Business, Information, Application, Technology, Security, Geospatial and Social architectures are merged into a coherent whole. The Architecture must act as a guide for the implementation teams, as source material for decision making and as a description of a system either before or after it is built. The Architecture Process must be well defined, repeatable and flexible to deal with the great variety of problems and opportunities that will be presented to the program. Fundamental to the execution of the process and the resulting architectures is the Architecture Team.


Juliette Wayton: Chief Information Officer
Hans Klimt: Architecture Review Board
Adam Goodchild: Chief Architect
:EA Steering Committee
Wendy Schubert: Security Architect
Barry Morgan: Business Architect
Johnathon Teage: Consultant Architect
Sara Lyonette: Information Architect
Henry Boncello: Repository Master
Yumiko Tashi: Application Architect
Tanya Donaldson: Geospatial Architect
Brent Talgram: Technology Architect
Legend
Architecture Executive
Architecture Team
Consultants

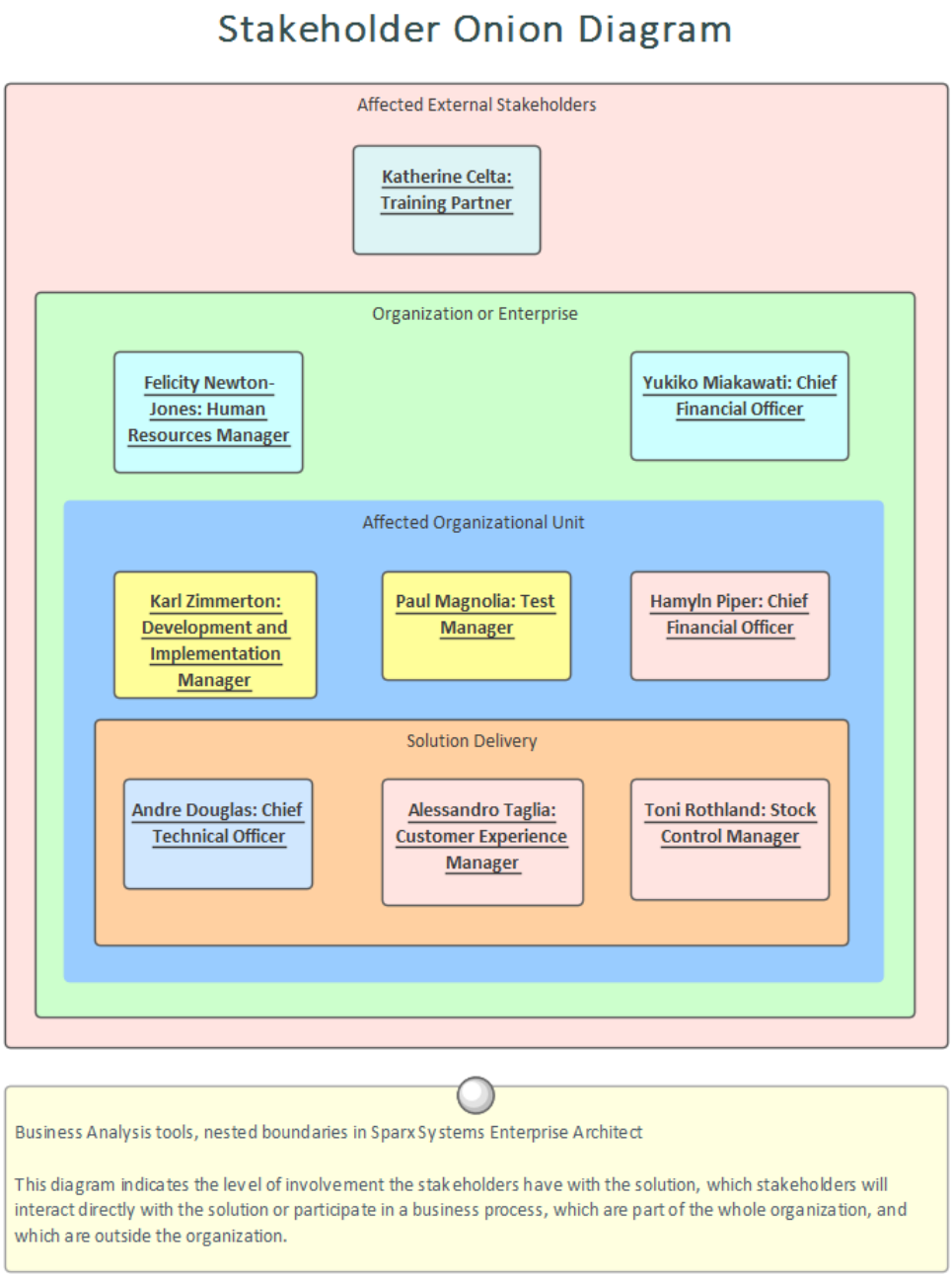
Stakeholder Modeling

Stakeholder Management is critical to successfully initiating and maintaining an architectural program. Architects will need to engage with a wide range of stakeholders, from senior executives through to implementation staff. The engagements will often require political sensitivity, diplomacy and flexibility to ensure that the stakeholders' needs and concerns are dealt with appropriately. Providing relevant and tailored views of the architectures will be critical to ensuring that the stakeholders are kept informed and that they spend the requisite time needed to understand the impact the architectures will have on their domain. Having a comprehensive communication plan is also critical to ensuring that the stakeholders receive the information they require and that they maintain an interest and have input into the architectures. Stakeholder analysis should be undertaken early in the set up of an architecture program and early in the development of an architecture, as an individual stakeholder's position with respect to a particular architectural initiative can vary. A matrix can be used to describe the positions key stakeholders or stakeholder groups hold with respect to aspects of the architecture program, such as how supportive they are, or what commitment they have to the architecture. The matrix can be updated and monitored over the lifetime of an architecture and remedial efforts made to manage the relationships with key stakeholders.

Source \ Target	Ability to Disrupt	Commitment to Architecture	Flexible	Supportive	Understanding of Domain
Chief Financial Officer	M	H	L	H	L
Chief Technical Officer	M	H	H	H	L
Customer Experience Manager	H	L	M	M	H
Development and Implementation Manager	H	H	L	M	M
Facilities Manager	L	L	H	M	M
Human Resources Manager	L	M	L	H	H
Operations Manager	H	H	L	M	H
Stock Control Manager	H	L	M	L	M
Test Manager	L	M	M	H	H
Training Partner	L	L	H	M	H

Enterprise Architect has a wide range of facilities and tools that can assist with Stakeholder Management. These include the ability to model the individual and groups of stakeholders, to classify them in a taxonomy and to show the extent of their influence using a series of nested Boundary elements.

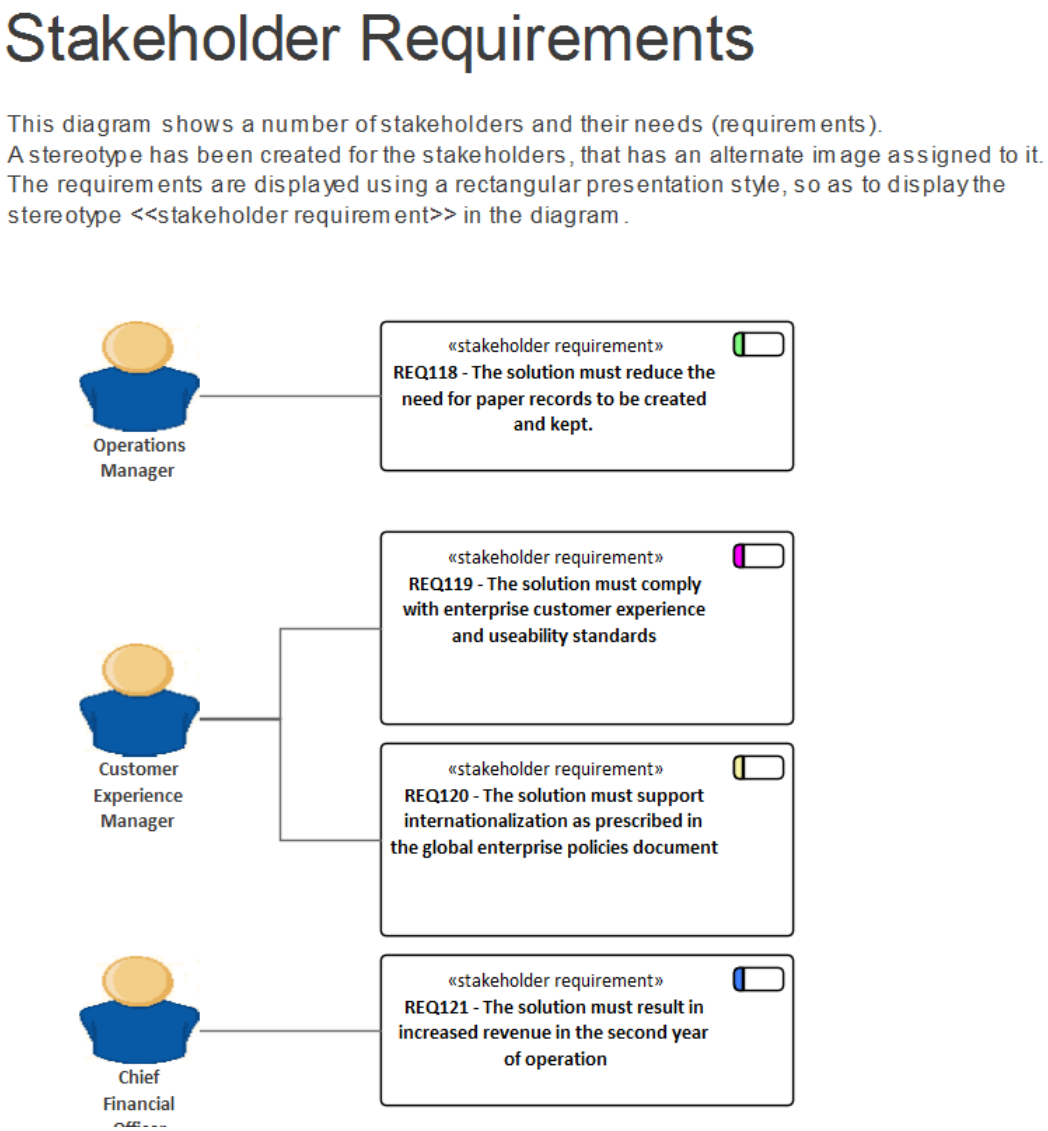
There is a wide range of lists, diagrams and matrices that will be relevant to certain stakeholders, including lists presented in the Specification Manager, Component diagrams describing Applications, Class diagrams used to present the Information Architectures, principles and a range of other ideas. The Calendar and Model Mail are useful tools for keeping stakeholders informed about things of interest and important events in the architecture program.



There is also a wide range of additional tools that assist when tailoring the information in the repository for individual or groups of stakeholders.

Requirements Modeling

Requirements engineering is one of the most important disciplines in the system lifecycle and when done well it will set the foundation for a successful architecture or program of work, ultimately ensuring that great value is delivered to the users and other stakeholders.



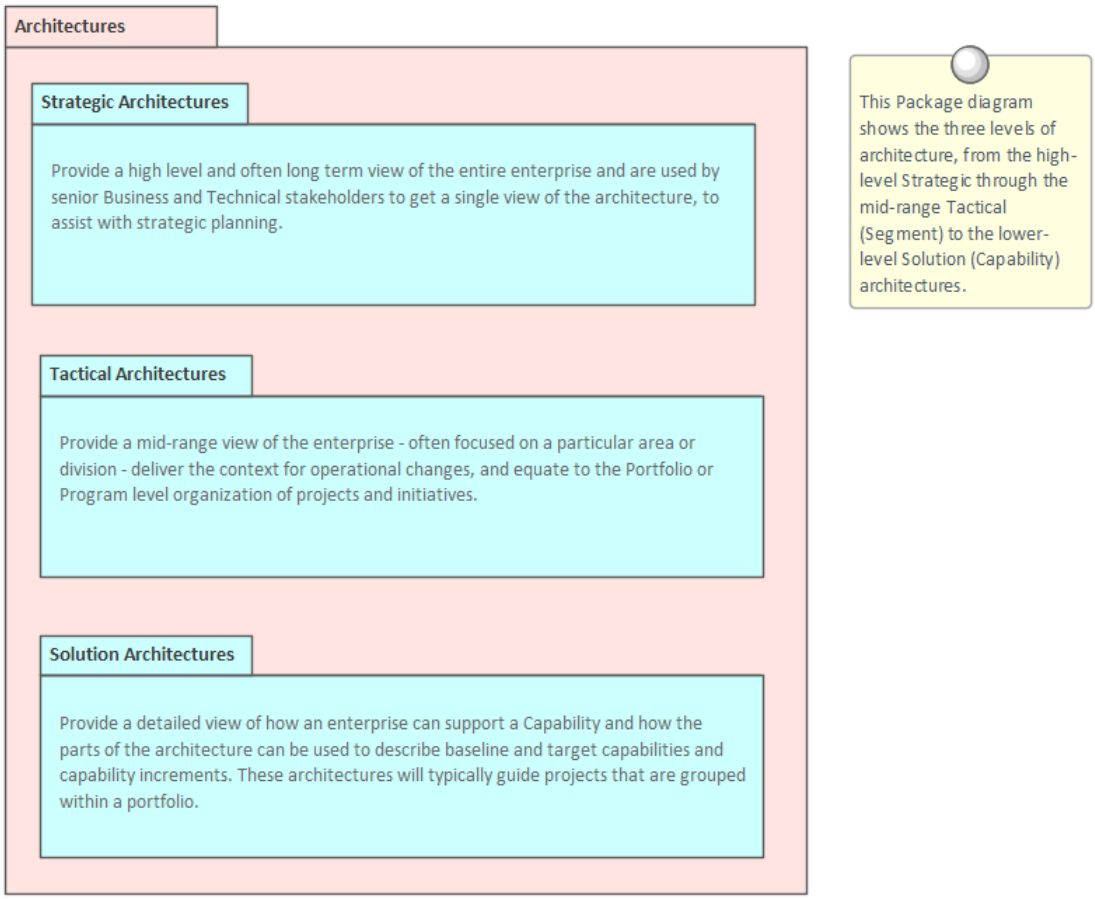
Enterprise Architect is a sophisticated and intuitive platform for developing and managing requirements from modeling stakeholders and visions, business cases, business drivers

and capabilities to detailed functional and non-functional requirements. Requirements can be prioritized, traced and tracked, and changes can be recorded, baselined, versioned and audited. Analysts can work together in a collaborative platform with role based Security, Discussions, Model Library, Model Mail and a range of other tools to encourage best practice and productivity.

Architectures

Architectures are the key organizing mechanism for architectural content; they are the design or solution statement to a proposed problem or opportunity, or the documentation of an existing system. The architectures can be defined at a number of different levels, including:

- Strategic - *Long term in the range of 3 - 5 years*
- Tactical - *Mid term in the range of 1 - 2 years*
- Solution - *Short term in the range of 3 - 12 Months*

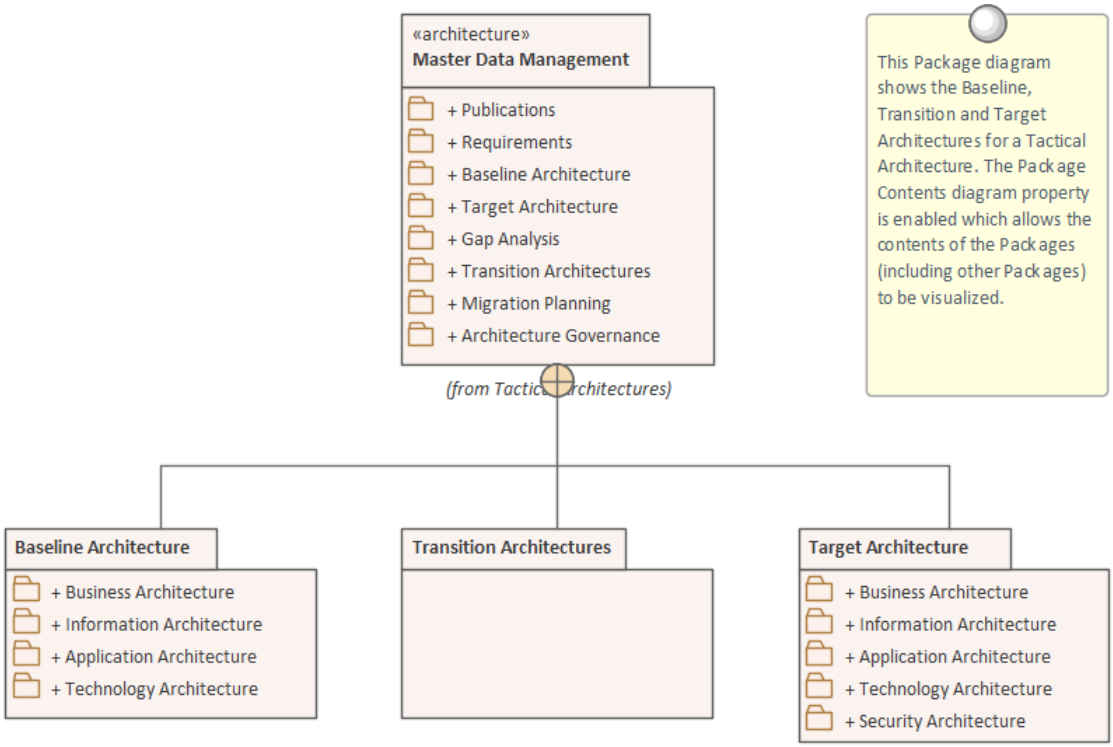


They also span a number of disciplines that are all incorporated into a single cohesive view of the system design. It is not possible for a single person or a single discipline to understand and to articulate the Architecture at all levels. So a Business Architect is required to interpret the Strategic Plan, an Information Architect would categorize the enterprise's data, an Application Architect would articulate the interfaces between Components, and a Technology Architect is required to define the Servers and Devices that ultimately do the work. Their work results in four core domain architectures:

- Business Architecture
- Information Architecture
- Application Architecture
- Technology Architecture

Most frameworks describe analogous or similar subsets of an Enterprise Architecture, as the division is based largely on organizational units performing work in these areas.

Baseline, Transition and Target Architectures can also be defined.


«architecture»
Master Data Management
+ Publications
+ Requirements
+ Baseline Architecture
+ Target Architecture
+ Gap Analysis
+ Transition Architectures
+ Migration Planning
+ Architecture Governance
(from Tactic rchitectures)
This Package diagram shows the Baseline, Transition and Target Architectures for a Tactical Architecture. The Package Contents diagram property is enabled which allows the contents of the Packages (including other Packages) to be visualized.
Baseline Architecture
+ Business Architecture
+ Information Architecture
+ Application Architecture
+ Technology Architecture
Transition Architectures
Target Architecture
+ Business Architecture
+ Information Architecture
+ Application Architecture
+ Technology Architecture
+ Security Architecture

Business Architecture

A well articulated Business Architecture is the cornerstone for a successful outcome of the overall Enterprise Architecture. It defines the business drivers, the business strategy, operational models, goals and objectives that the organization must have to achieve transition in a potentially competitive and disruptive business environment. Architects working in the other architecture disciplines must understand the Business Architecture as the basis for their own architecture descriptions and as a guide to the business outcomes that must be achieved.

The Business Architecture will typically consist of a description of the baseline and target architectures with a series of transitions defined that can be executed and that would be described on Roadmap diagrams.

Enterprise Architect has a wide range of features that can be used to model the Business Architecture and to create compelling visualizations that will help the executive level stakeholders and line managers to understand how their strategies are being addressed by the architectures. When an architecture program is set up or a business architecture is initiated it will typically be found that a number of important architectural materials will be available, such as mission and vision statements, strategic plans, business drivers and goals and - more often than not - business processes and capabilities will be at least started even if not complete. Enterprise Architect has a convenient facility to

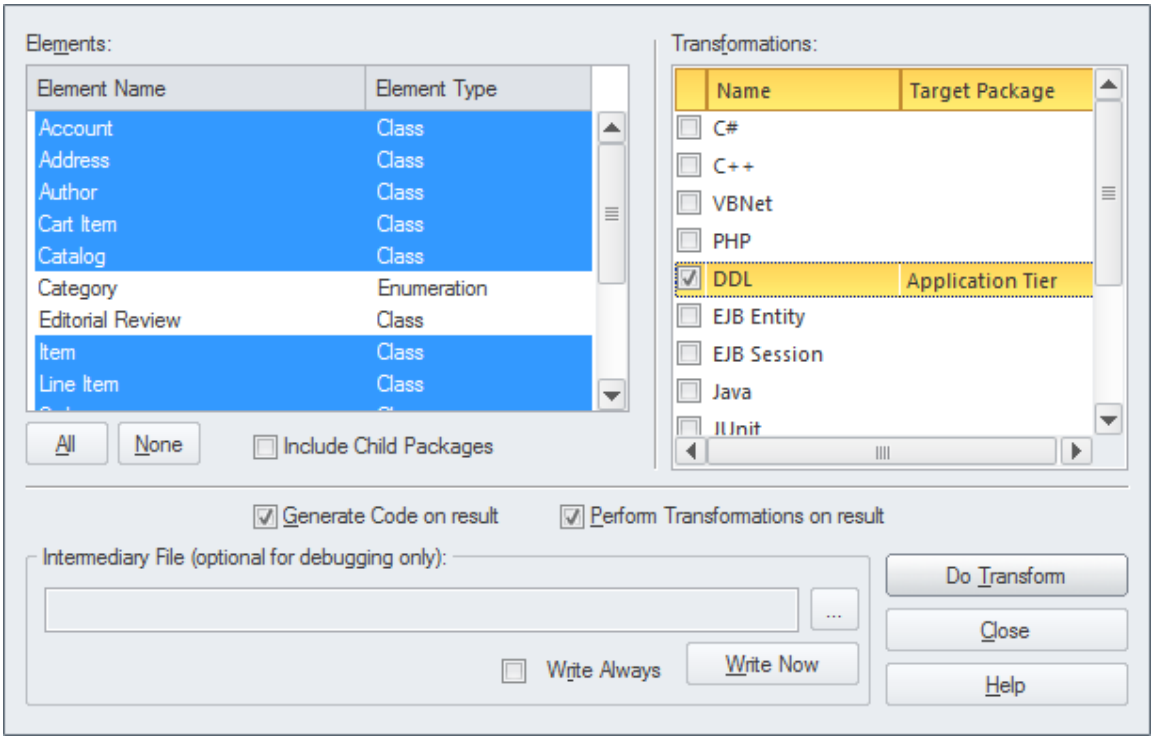
import the content of spreadsheets and word processor tables directly into the repository, saving time and effort.

Information Architecture

Information Architecture is key to the success of an Enterprise Architecture Program, as information is created, consumed and destroyed by the components that make up the other architectures. Understanding which business functions and processes utilize information, which applications serve as the master record, where information is created and destroyed, and which technology components store and manipulate the information, is critical to achieving the business outcomes.

The information architecture will typically consist of a description of the baseline and target architectures, with a series of transitions defined that can be executed and that would be described on Roadmap diagrams.

Enterprise Architect is a profoundly useful tool for creating and maintaining information architectures, with its sophisticated and extensive support for standards and its wide range of tools to support information models, from high level classifications and concepts right down to the level of schemas and the elements and columns they are composed of. Tools such as the Schema Composer and the Database Builder, along with the UML Class diagram and Glossary, and the Model Transformation facility, will be invaluable.



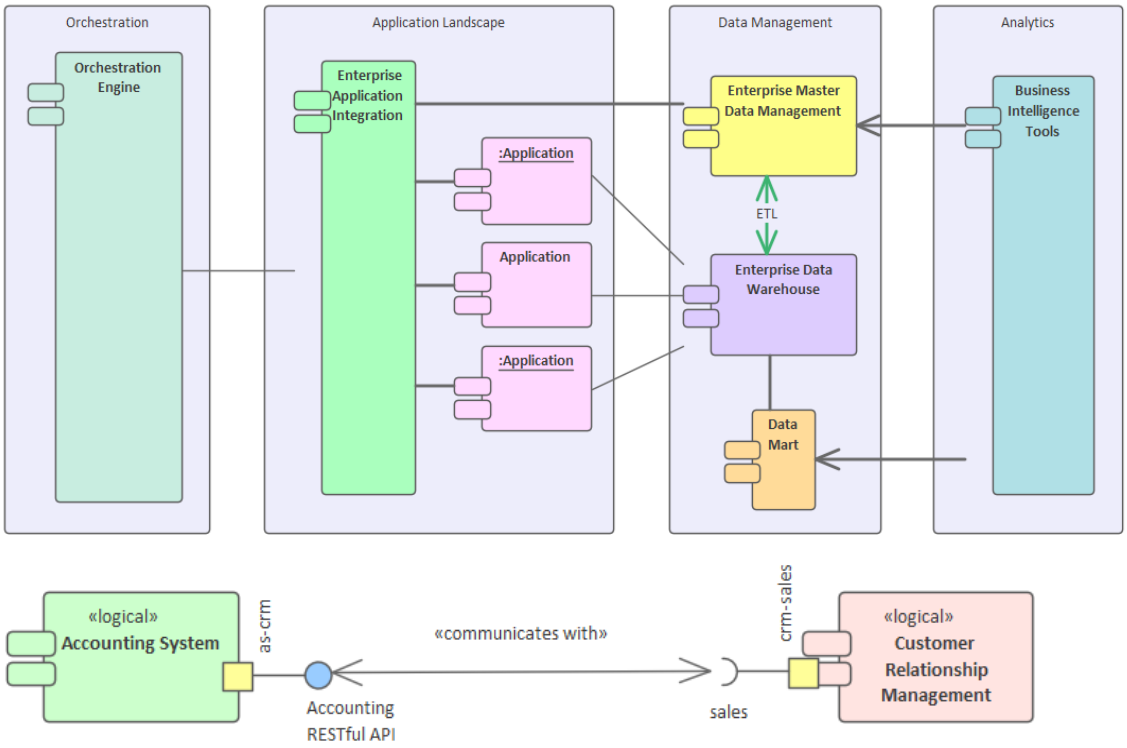
Learn More: Information Architecture

Application Architecture

The Application Architecture provides an important catalog of the applications in the enterprise describing the work that they do to transform, transmit and store information. The architecture also describes the interfaces that are required or provided by the applications and the way the applications interact to carry out the activities described in the business models such as the Business Process diagrams. The catalog of applications, interfaces and the diagrams and matrices that describe their interaction only need to be defined once at the enterprise level. An application architect will be able to draw upon this inventory of existing artifacts to create new architectures, classifying them as part of the baseline and potentially the future state architecture. Where an architecture introduces new applications, these can be added to the description of the target state.

The Application Architecture will typically consist of a description of the baseline and target architectures with a series of transitions defined that can be executed and that would be described on Roadmap diagrams.

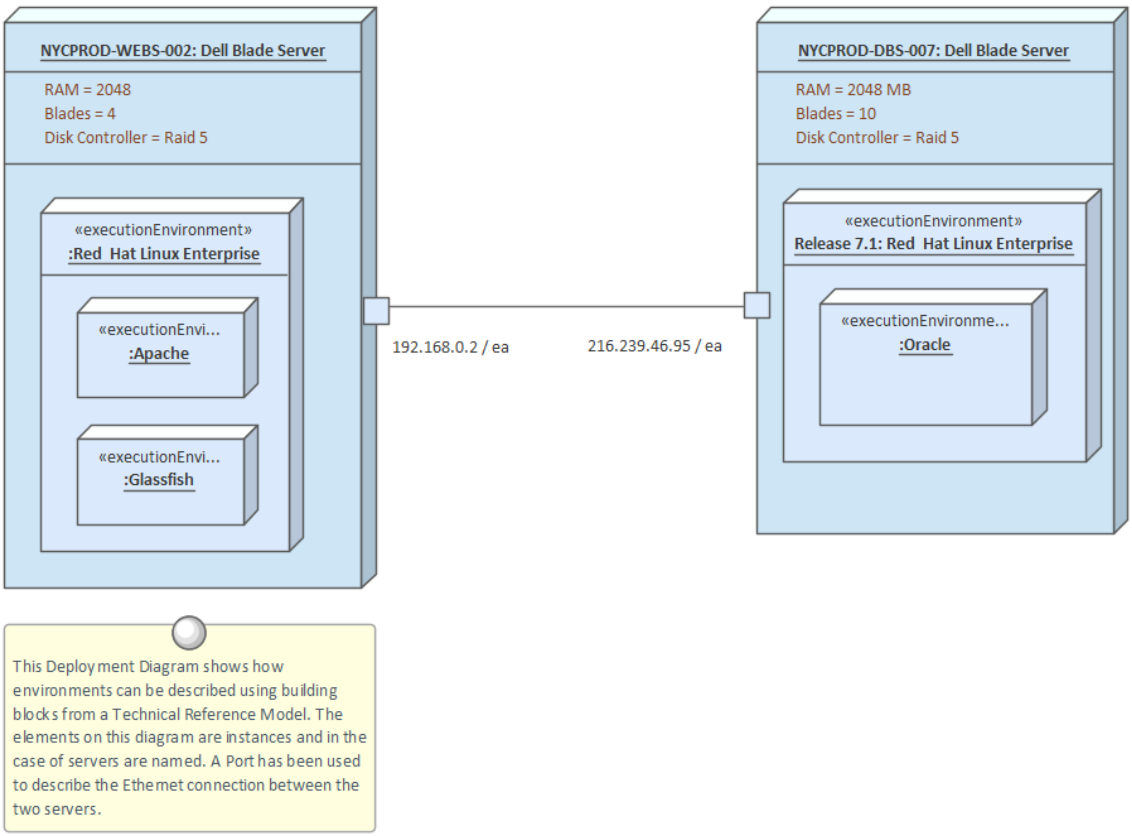
Learn More: Application Architecture



This Component diagrams shows the details of the interface between two Logical Components using Ports and Interfaces.

Technology Architecture

The technology architecture underpins the other architectures, providing a description of the logical, physical and virtual infrastructure that supports the execution of application services, which in turn support information and business functions and services.



The Business Model

Create Powerful and Expressive Diagrams of an Organization from Strategy to Implementation

Change is a constant in business - change in market opportunities, change in source materials, change in customer audience, change in environment - all leading to change in business processes. Re-engineering business processes is a strategic part of any business. The basic requirement that enterprise-class business process engineering principles are hard-wired into operations is applied across the spectrum of different business sectors. In fulfilling this fundamental governance requirement, Enterprise Architect and a number of standards, including the Business Process Model and Notation (BPMN), are essential tools.

Enterprise Architect includes a wide range of modeling tools, diagram types, Patterns, Technologies and capabilities to support modeling a wide range of business-focused and analytic domains. From strategic models that support capturing information and goals pertaining to the overall functioning of a large enterprise, to simple Mind Maps, process modeling, Requirement Management, BPMN models and more, Enterprise Architect is the ideal platform for building and sharing business-based models that can tightly integrate into an overall enterprise or system-level architecture.

In addition to the extensive range of template-driven reports

available, Enterprise Architect includes comprehensive support for building management-level charts and dashboards that integrate tightly with model content. Series Charts, Pie Charts, Heat Maps and others make it possible to build a high level summary of the state of the current model (and of other models using custom connections).

Business Modeling Overview

This table identifies some of the material to be discussed in the Business Engineering section. Note that many of the capabilities listed here are equally applicable to Systems-, Software- and Enterprise-level modeling and design.

Note that some features are edition dependent - see the Sparx Systems web site for details on which edition supports which features.

Feature	Description
Strategic Models	A range of strategy-based model types, including Strategy Maps, Value Chain, Balanced Scorecard, Dataflow and Org Chart. Essential modeling for management and for expressing overall business or enterprise level goals and strategy.
Charts and Dashboards	Material supporting the building of dashboards and the specification of

	Charts based on model content. Charts provide a real-time view of the health and status of a model, and allow for hot-spots and problem areas to be detected early and dealt with accordingly. A number of types of Chart are available, including Pie Charts, Series Charts, Bar Charts and Heat Maps.
Analysis Models	This section identifies a number of analytic diagram types that have an affinity for business engineering and analysis. Mind Mapping, Process diagrams and custom stereotypes targeting business scenarios and terms are some of the features discussed.
Requirements	Coverage of managing Requirements in Enterprise Architect for the business user. A shortened version of the book length "Requirements Engineering" topic also available from Sparx Systems. Covers creating, tracing, managing and generally dealing with all kinds of requirements.
Decision Models	Information about building Decision Tables as per the OMG Decision Modeling and Notation (DMN) standard. Shows how to build a Decision Table and

	how to display that table on a diagram.
Business Rules	An advanced mechanism for creating and working with Business Rules in a manner similar to the Decision Tables specified earlier but with some additional functionality that allows for the translation of those rules into activity charts and into code.
BPMN Support	Enterprise Architect has very detailed and comprehensive support for the Business Process Model and Notation (BPMN) specification. BPMN is a popular business and process modeling notation that targets unambiguously defining business (and other) processes in a visual manner with downstream capabilities for simulation, model exchange and code generation.
BPSim Support	BPSim provides the Business Modeler with a mechanism for defining additional characteristics and parameters for a BPMN model that allows the BPMN model to be simulated with different 'real world' scenarios. Performance data such as time, efficiency, throughput, resources used, resources idle and so on, can be

	captured and reported from a suitable BPSim engine. Sparx Systems provide a BPSim-capable Simulator - the BPSim Execution Engine - which integrates with the BPSim and BPMN models defined in Enterprise Architect, providing the capability to run and store the results from multiple Simulations and to perform convenient comparisons across each configuration's result set. The Sparx Systems BPSim Execution Engine is a pre-requisite for accessing and using the built-in BPSim configuration facilities.
BPEL	Support for modeling and working with the BPEL (Business Process Execution Language), a subset of BPMN that supports export in the BPEL execution language format.
Business Modeling Notation	Stereotypes and modeling conventions from the UML business modeling extensions defined in earlier versions of the UML.
Eriksson-Penker Notation	Support for the Business Process Model and Notation (BPMN), defined by Eriksson and Penker. A useful and

	straightforward process mapping that quickly builds maps of processes, inputs, outputs and goals that constitute a business or enterprise level process.
Case Management Model & Notation (CMMN)	Within Enterprise Architect, the Case Management Model & Notation helps you to create CMMN models for representing actions taken regarding a subject in a particular situation to achieve a desired outcome.

Business Models

Visualize a Wide Range of Business Concepts, Knowledge and Risks in a Unified Model

Business analysis, through the efforts of a number of international organizations, has become a rigorous and regulated discipline. The Business Analyst is expected to discover, analyze and synthesize large volumes of information and be able to present it clearly and meaningfully to heterogeneous audiences. The Business Analyst typically creates models of requirements, policies, business rules, business processes and information. All of these concepts can be created easily in a single repository and integrated, allowing the analyst to spend their time on analysis instead of copying content between tools. At any stage during the analysis, high quality documentation tailored to a specific audience and spanning all the content can be created in a wide range of formats.

Business Model Diagram Types

Diagram Type	Description
Requirements	Create, model and manage requirements directly in line with your other project

	tasks. Trace from high level requirements to deployment artifacts.
Use Cases 	Describe the functional requirements of the system, the manner in which outside things (Actors) interact at the system boundary, and the response of the system.
BPMN 	Capture the behavior and the information flows within an organization or system.
Decision Models 	Define decisions with models and tables using the decision trees or decision tables.
Mind Mapping 	Capture high level ideas and concepts and trace them with down stream actions.
Business Rules 	Identify and store business rules within your model while integrating them with downstream processes.
BPEL 	Generate Business Process Execution Language (BPEL) from processes described using BPMN.

SPEM 	A process engineering meta-model as well as conceptual framework, which provides the necessary concepts for modeling, documenting, presenting, managing, interchanging and enacting development methods and processes.
ArchiMate 	Define your business capability with ArchiMate, an open-standard Enterprise Architecture language.
Eriksson-Penker Extensions 	A framework for UML business processing model extensions.

Business Modeling/Interaction

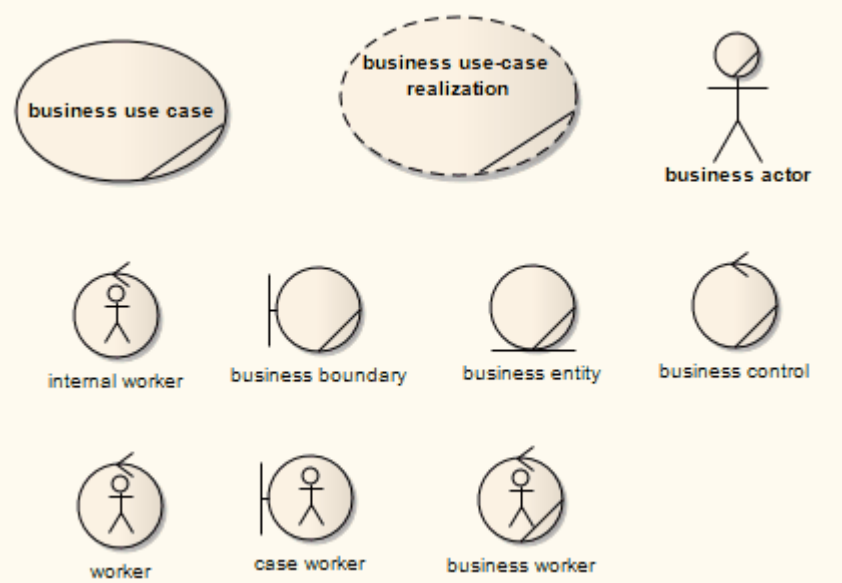
Business Modeling diagrams and Business Interaction diagrams enable you to model both the structure and behavior of a business system. Business Modeling diagrams are based on a Class (UML Structural) diagram, whilst Business Interaction diagrams are based on a Sequence (UML Behavioral) diagram. Both diagram types have the same default Toolbox, which consists of a Business Modeling element page. The available elements include stereotyped Objects, and a stereotyped Actor (Business Actor), Use Case (Business Use Case) and Collaboration (Business Use Case Realization).

Example Diagram

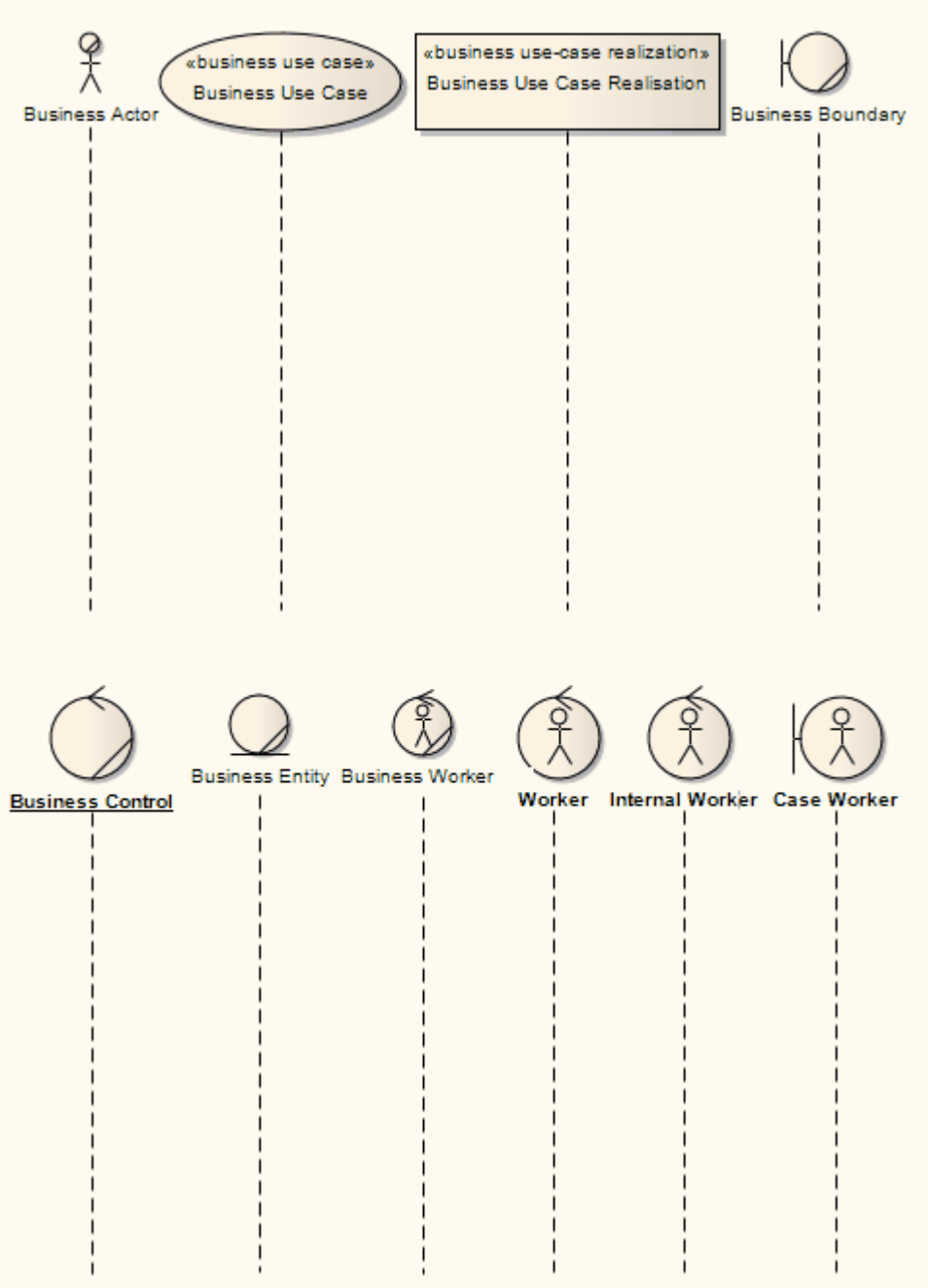
Example Business Modeling Diagram

Example Business Modeling Diagram

This diagram shows the appearance of the elements dragged from the 'Business Modeling' pages of the Diagram Toolbox and dropped onto a Business Modeling diagram.



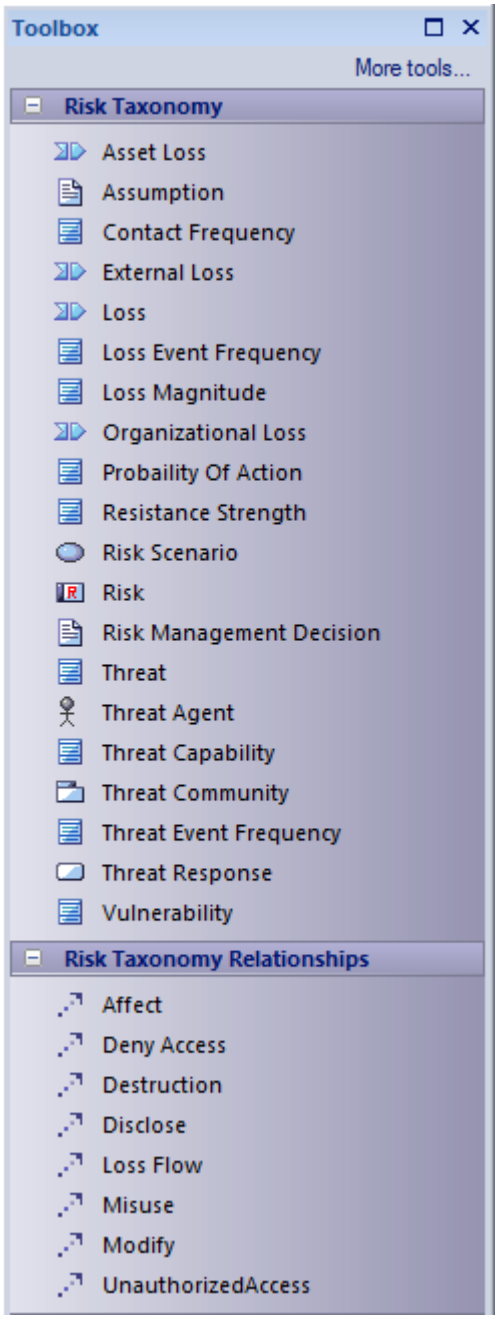
This diagram shows the appearance of the elements dragged from the 'Business Modeling' pages of the Diagram Toolbox and dropped onto a Business Interaction diagram.


Business Actor
«business use case»
Business Use Case
«business use-case realization»
Business Use Case Realisation
Business Boundary
Business Control
Business Entity
Business Worker
Worker
Internal Worker
Case Worker

Risk Taxonomy

The UML Profile for Risk Taxonomy supports the Open Group Standard for Risk Taxonomy (OR-T), Version 2.0, which provides support for modeling risk scenarios and analyzing the risk conditions.

Risk Taxonomy Toolbox



Elements

Icon	Description
Asset Loss	Defines the loss on an asset to the

	organization (in terms of the assets's value/liability and volume). Tagged Values • Liability/Value • Volume • Cost – The intrinsic value of the asset • Criticality – Very High (VH), High (H), Moderate (M), Low (L), Very Low (VL) • LossFactor • LossForm – Productivity, Response, Replacements, Fines and Judgements, Competitive Advantage, Reputation • LossType – Primary, Secondary • Sensitivity • Embarrassment/ Reputation - The information provides evidence of incompetent, criminal, or unethical management; note that this refers to reputation damage resulting from the nature of the information itself, as opposed to reputation damage that might result when a loss event takes place • Competitive Advantage - The information provides competitive advantage (such as key strategies or trade secrets); of the sensitivity

	categories, this is the only one where the sensitivity represents value - in all other cases, sensitivity represents liability • Legal Regulatory - The organization is bound by law to protect the information • General - Disclosure of the information results in some form of loss
Assumption	Captures assumptions made in risk analysis. Tagged Values • Rationale
Contact Frequency	The probable frequency, within a given timeframe, with which a threat agent will come into contact with an asset. Tagged Values • ConfidenceLevel - Commonly expressed as a percentage • MaximumLikelyValue • MinimumLikelyValue • Type – • Random - The threat agent 'stumbles upon' the asset during the course of unfocused or undirected activity • Regular - Contact occurs because of the

	regular actions of the threat agent • Intentional - The threat agent seeks out specific targets
External Loss	Captures external loss factors. Tagged Values • Factors – Event Detection, Legal and Regulatory, Competitors, Media, Other Stakeholders • Cost • Criticality – Very High (VH), High (H), Moderate (M), Low (L), Very Low (VL) • LossFactor • LossForm – Productivity, Response, Replacements, Fines and Judgements, Competitive Advantage, Reputation • LossType – Primary, Secondary • Sensitivity – • Embarrassment/ Reputation - The information provides evidence of incompetent, criminal, or unethical management; note that this refers to reputation damage resulting from the nature of the information itself, as opposed to reputation damage that can result when a loss event takes place • Competitive Advantage - The

	information provides competitive advantage (such as key strategies or trade secrets); of the sensitivity categories, this is the only one where the sensitivity represents value - in all other cases, sensitivity represents liability • Legal Regulatory - The organization is bound by law to protect the information • General - Disclosure of the information results in some form of loss
Loss	Captures loss that a threat can result in. Tagged Values • Cost • Criticality – Very High (VH), High (H), Moderate (M), Low (L), Very Low (VL) • LossFactor • LossForm – Productivity, Response, Replacements, Fines and Judgements, Competitive Advantage, Reputation • LossType – Primary, Secondary • Sensitivity • Embarrassment/ Reputation - The information provides evidence of incompetent, criminal, or unethical management; note that this refers to

	reputation damage resulting from the nature of the information itself, as opposed to reputation damage that might result when a loss event takes place • Competitive Advantage - The information provides competitive advantage (such as key strategies or trade secrets); of the sensitivity categories, this is the only one where the sensitivity represents value - in all other cases sensitivity represents liability • Legal Regulatory - The organization is bound by law to protect the information • General - Disclosure of the information results in some form of loss
Loss Event Frequency	The probable frequency, within a given timeframe, that a threat agent will inflict harm upon an asset. Tagged Values • LossType • ProbableFrequency - Probability is always is based on a timeframe (event X is 10% likely to occur over the next Y) • Rating - Very High (VH), High (H),

	Moderate (M), Low (L), Very Low (VL) • TimePeriod
Loss Magnitude	The probable magnitude of loss resulting from a loss event. Tagged Values • LossType • Rating - Severe (SV), High (H), Significant (Sg), Moderate (M), Low (L), VeryLow (VL)
Organizational Loss	Captures the loss to the organization. Tagged Values • Factors – Timing, DueDiligence, Response, Detection • Cost – The intrinsic value of the asset • Criticality – Very High (VH), High (H), Moderate (M), Low (L), Very Low (VL) • LossFactor • LossForm • Productivity - The reduction in an organization's ability to generate its primary value proposition • Response - Expenses associated with managing a loss event (such as internal

	or external person-hours, logistical expenses, legal defense and public relations expenses) • Replacement - The capital expense associated with replacing lost or damaged assets • Fines and Judgements - Legal or regulatory actions levied against an organization • Competitive Advantage - Losses associated with diminished competitive position • Reputation - Losses associated with an external stakeholder's perception that an organization's value proposition is diminished • LossType – Primary, Secondary • Sensitivity • Embarrassment/ Reputation - The information provides evidence of incompetent, criminal, or unethical management; note that this refers to reputation damage resulting from the nature of the information itself, as opposed to reputation damage that might result when a loss event takes place • Competitive Advantage - The

	information provides competitive advantage (such as key strategies or trade secrets); of the sensitivity categories, this is the only one where the sensitivity represents value - in all other cases, sensitivity represents liability • Legal Regulatory - The organization is bound by law to protect the information • General - Disclosure of the information results in some form of loss
Probability of Action	The probability that a threat agent will act against an asset once contact occurs. Tagged Values • ConfidenceLevel - Commonly expressed as a percentage • LevelOfEffort - Very High (VH), High (H), Moderate (M), Low (L), Very Low (VL) • MaximumLikelyValue - T he threat agent’s maximum value proposition from performing the act • MinimumLikelyValue - The threat agent’s minimum value proposition from performing the act • MostLikelyValue - The threat agent’s perceived value proposition from

	performing the act • RiskOfDetection/Capture - Very High (VH), High (H), Moderate (M), Low (L), Very Low (VL)
Resistance Strength	The strength of a control as compared to a baseline measure of force. Tagged Values • ConfidenceLevel - Commonly expressed as a percentage • LevelOfEffort - Very High (VH), High (H), Moderate (M), Low (L), Very Low (VL) • MaximumLikelyValue • MinimumLikelyValue • MostLikelyValue • Rating - Very High (VH), High (H), Moderate (M), Low (L), Very Low (VL)
Risk Scenario	Defines the environment or situation that involves risk.
Risk	Defines an estimate of the probable frequency and magnitude of future loss Tagged Values • Rating - Very High (VH), High (H), Moderate (M), Low (L), Very Low

	(VL)
Risk Management Decision	The decision to mitigate the risk.
Threat	Defines the harm to an asset that a risk poses. For electronic data storage and flow, also consider the Cybersecurity elements in Threat modeling. Tagged Values • ThreatScenario – Malicious, Error, Failure, Natural • ThreatType – External, Internal
Threat Agent	The source that could inflict harm upon an asset. Tagged Values • AccessMethod • DesiredVisibility • Motive • Objective • PersonalRiskTolerance • Resources • SkillRating • Sponsorship

	• ThreatType – External, Internal
Threat Capability	The probable level of force that a threat agent is capable of applying against an asset. Tagged Values • ConfidenceLevel - Commonly expressed as a percentage • MaximumLikelyValue - The maximum level of skills an attacker might have • MinimumLikelyValue - The minimum level of skills that an attacker might have • MostLikelyValue - The skill level of the most likely attacker • Rating - Very High (VH), High (H), Moderate (M), Low (L), Very Low (VL)
Threat Community	People associated with the conditions surrounding the asset at risk. Tagged Values • ThreatType – External, Internal
Threat Event Frequency	The probable frequency, within a given timeframe, at which a threat agent will act against an asset. Tagged Values

	• Rating - Very High (VH), High (H), Moderate (M), Low (L), Very Low (VL)
Threat Response	The action associated with managing a loss event. Consider also Threat mitigation as through the Threat Modeling facility for Cybersecurity. Tagged Values • ResponsePercentage • ResponseType – • Containment - An organization's ability to limit the breadth and depth of an event • Remediation - An organization's ability to remove the threat agent • Recovery - The ability to bring things back to normal
Vulnerability	The probability that a threat event will become a loss event. Tagged Values • Rating - Very High (VH), High (H), Moderate (M), Low (L), Very Low (VL)

Relationships

Icon	Description
Affect	Represents an impact that a threat has on an asset.
Deny Access	Represents a relationship type of a threat to an asset.
Destruction	Represents loss on an asset, such as destruction or theft of a non-data asset.
Disclose	Represents an agent illicitly disclosing sensitive information.
Loss Flow	Represents mapping between losses.
Misuse	Represents unauthorized use of assets (such as identity theft, or setting up an illegal distribution service on a compromised server).
Modify	Represents unauthorized changes to am asset.
Unauthorized	Represents a simple unauthorized access

Access	relationship on an asset.

Strategic Models

Enterprise Architect contains a number of diagram types that are specifically aligned with creating high quality strategic business models. Strategic models are typically defined by executives, modelers and managers responsible for defining strategic direction and both short and long term goals.

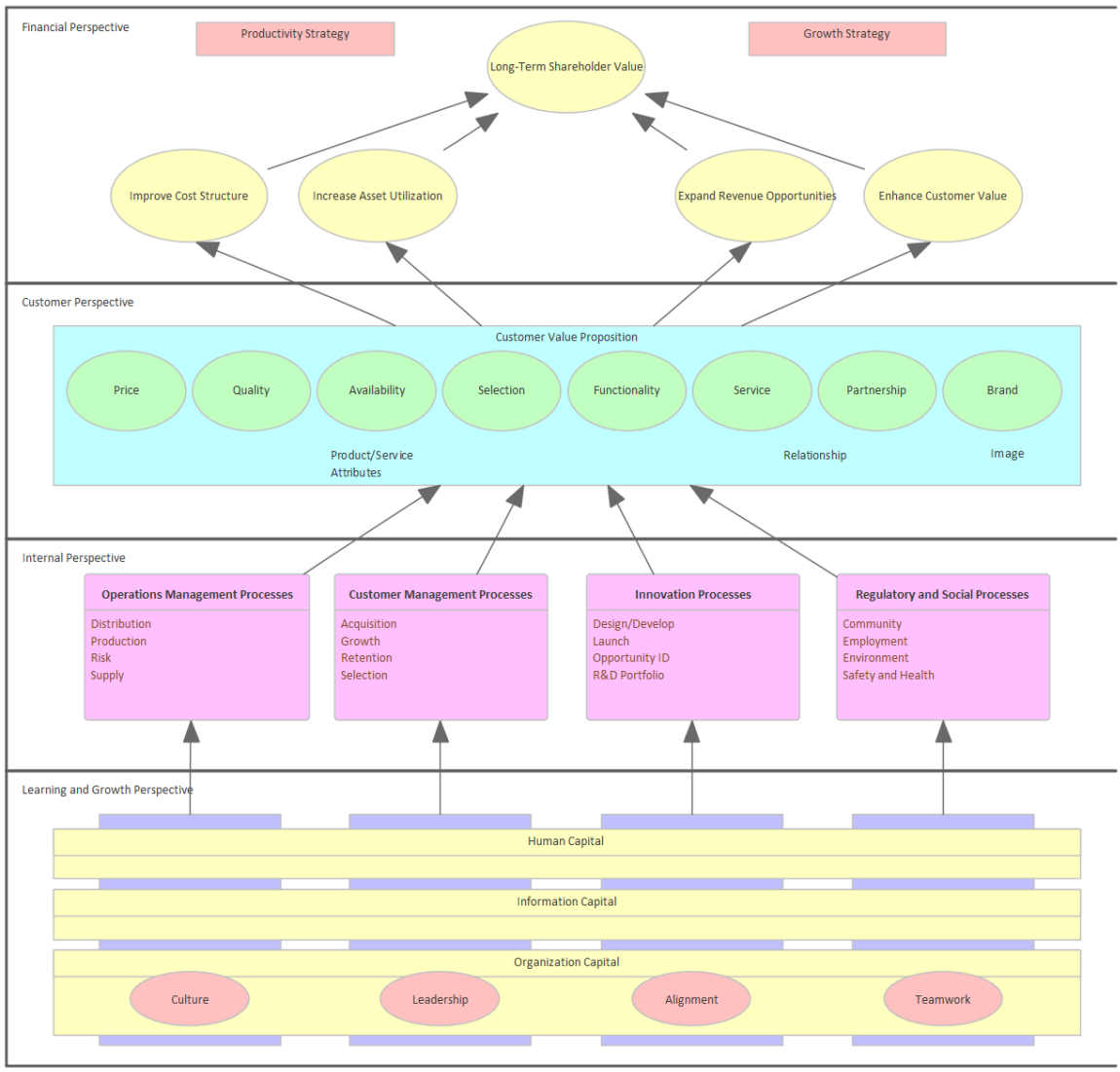
Advantages of Strategic Models

Defining and visualizing strategy as a series of diagrams/models within an Enterprise Architect repository has numerous advantages over defining that strategy in some other format. First and foremost, by modeling strategy within a repository, you immediately link the strategic direction and executive thinking to the processes, architectures, software and capabilities that are modeled within that same repository. This is a highly effective means of ensuring that strategic direction is correctly implemented and tied to the fundamental infrastructure and direction of an enterprise.

With strategic direction, goals, decisions and other critical business information defined up front and always accessible to designers, modelers, managers and others working on implementing a strategy, it is significantly more likely that the preferred strategic direction will be adhered to and less likely that simple errors and misunderstandings will occur.

As the solution implementation takes shape, with full traceability available as models are constructed, it is possible to trace physical and logical solutions right back through requirements and Use Cases to the actual strategic goals that require that solution to exist. Aligning business needs with infrastructure and capabilities is now much easier and less error prone.

This section will discuss the principal strategic modeling types. Note that there are also significant other model types that can be used to convincingly portray strategy, for example Mind Mapping.



Value Chains

Present and Identify Customer and Shareholder Value in Compelling Diagrams

A Value Chain (Porter, 1985) examines the performance and cost of each value-creation activity within an organization. It is a tool that can be used for strategic planning purposes, to identify parts of an organization that provide the best value for customers and shareholders. A Value Chain is often used in conjunction with financial reporting to control cost drivers and identify areas that need improvement.

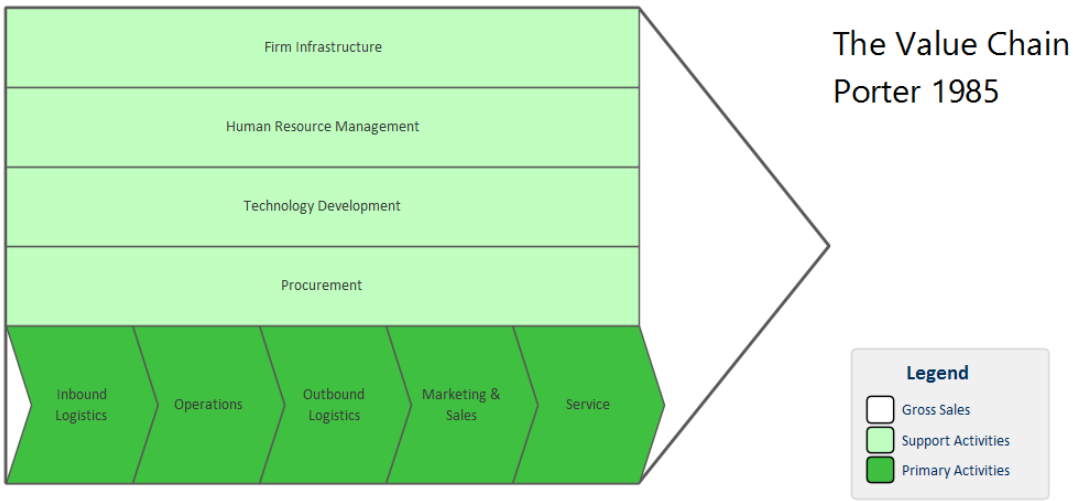
If a company can derive greater value from one of the activities in the Value Chain, they will be able to beat the competition, creating greater value for the shareholder and customer. For example: A business might have access to cheaper raw materials (Inbound Logistics), use a more efficient manufacturing process (Operations) or reduce transport and storage costs (Outbound Logistics). These primary activities are in turn supported by a number of secondary activities such as HR Management and Technology. A business is able to gain a competitive advantage by analyzing each activity in the chain, benchmarking the activity against the competition, improving processes, providing better differentiation and reducing costs.

Enterprise Architect is able to link each activity in the Value

Chain to support documentation, including budgets and benchmark results. The ability to link to competitors' websites and web based market information extends the power of the Value Chain beyond a traditional text based model.

Creating a Value Chain

- Create a new Package called 'Value Chain'
- Create a new Value Chain diagram called 'Our Company' and click on the OK button
- Select the 'Value Chain' Pattern from the Toolbox and place it on the Diagram View



- The Gross Sales element acts as a frame for the Primary Activity and Support Activity elements
- The default Value Chain can be customized to reflect how your company operates
- You can easily modify the color of each activity using the

'Layout' ribbon, 'Style' panel

Strategy Maps

Plan and Communicate Corporate Strategy from Different Perspectives

The Strategy Map extends the Balanced Scorecard approach in order to clarify corporate strategy, using four perspectives and their 'Cause & Effect' relationship. The four perspectives are:

- Financial
- Customer
- Internal Business Process
- Learning and Growth

Strategy Maps are typically used for planning and communicating the overall strategy of a given organization, in order to create sustained value for customers and shareholders in the form of a 'Cause & Effect' chain. Strategy Maps provide a greater emphasis on measuring and managing the value created by enhancing an organization's intangible assets. Balanced Scorecards and Strategy Maps were both developed by Kaplan and Norton in 1996 (*Strategy and Leadership,* Kaplan and Norton 2004).

Intangible assets are items of value that can not necessarily be seen or touched. Examples of intangible assets include your brand, intellectual property, reputation, good will, patents, creativity and staff morale. A large percentage of an organization's shareholder and customer value is associated with intangible assets, yet traditional financial management

systems fail to track, measure or evaluate them. As a result it is important that tools such as a Balanced Scorecard and Strategy Map are used to develop a more balanced approach to Strategic Planning. The ability to see 'Cause & Effect' relationships helps an organization understand the chain of events that sees intangible assets transformed into real value.

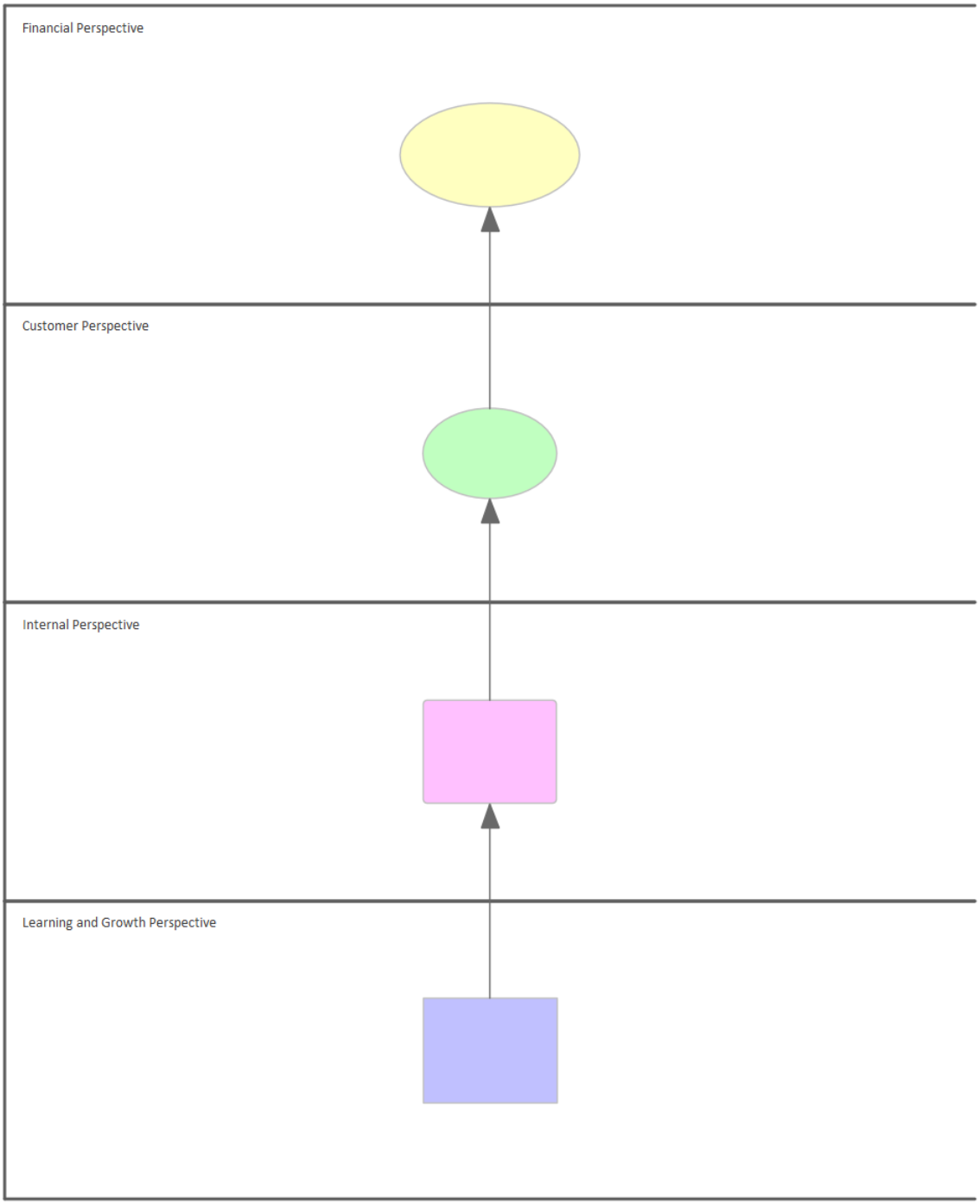
Strategy Map Patterns

Enterprise Architect provides three basic Patterns that can be used to create a Strategy Map. The first Pattern is very basic and provides the greatest flexibility with respect to the layout of your Strategy Map. The Toolbox provides a number of tools such as a Mission and an Intangible Asset. The second and third Patterns provide a more comprehensive overview, with far more internal processing and Intangible Assets.

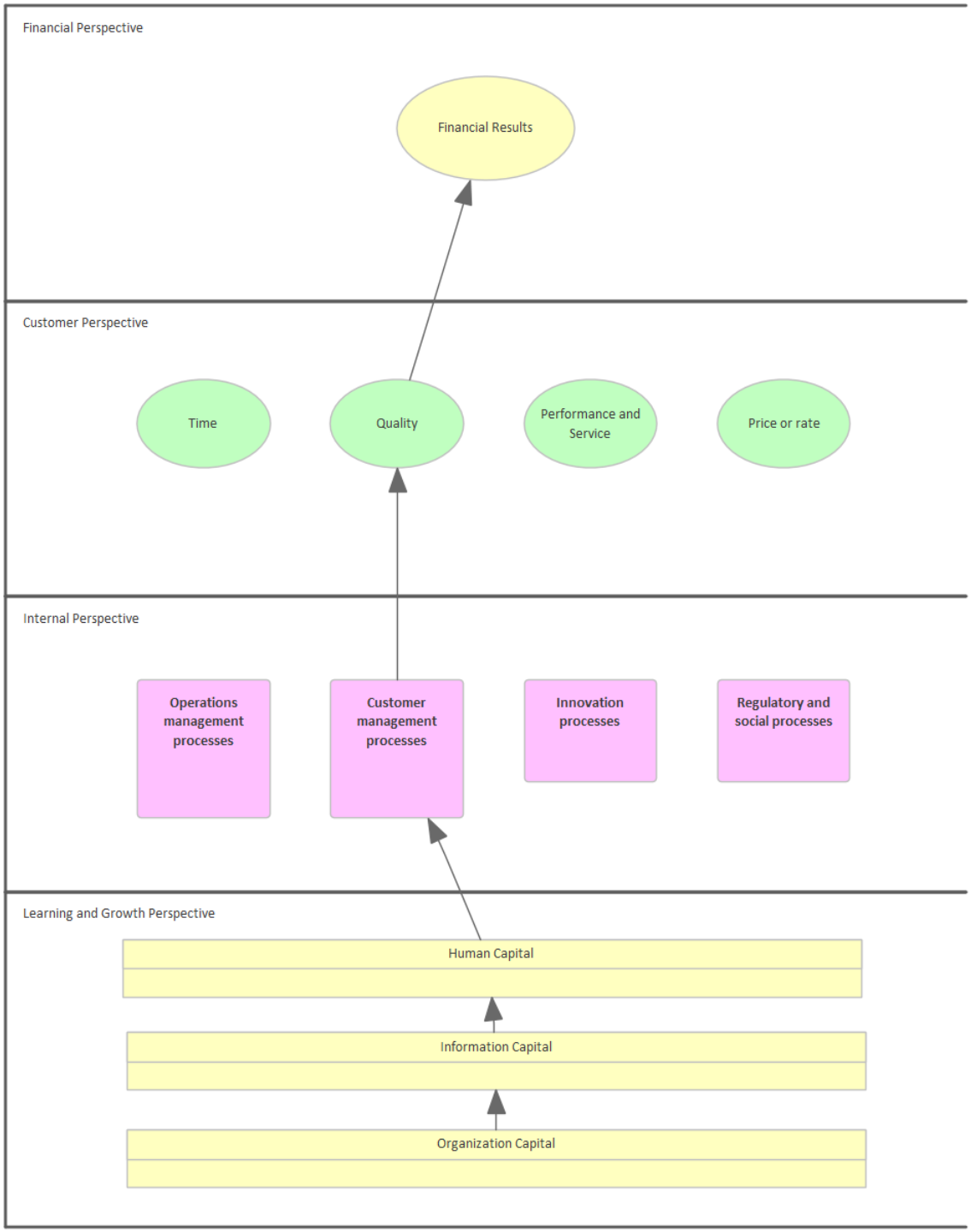
Strategic objectives are represented as an oval.

1. Create a new Package called 'Strategy Map'
2. Create a new Strategy Map diagram called 'Our Strategy'
3. Select the Pattern entitled 'Strategy Map 2' and place it onto the Diagram View
4. This diagram will display all four perspectives and include a detailed breakdown of each
5. Examine the two additional Patterns in order to see the layout of each alternative Strategy Map example

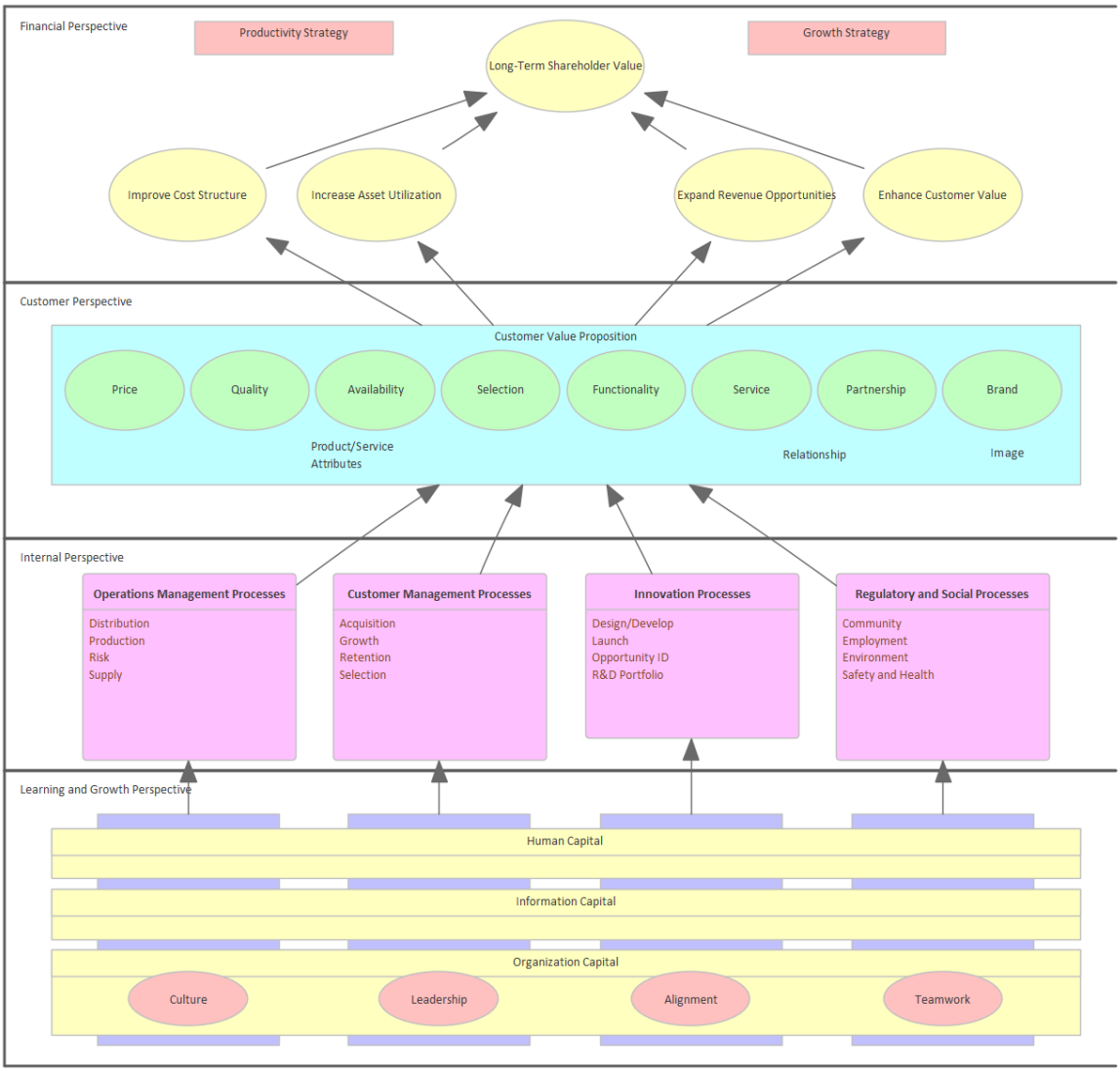
Example Strategy Map (Style 1)



Example Strategy Map (Style 2)



Example Strategy Map (Style 3)


Financial Perspective
Productivity Strategy
Growth Strategy
Long-Term Shareholder Value
Improve Cost Structure
Increase Asset Utilization
Expand Revenue Opportunities
Enhance Customer Value
Customer Perspective
Customer Value Proposition
Price
Quality
Availability
Selection
Functionality
Service
Partnership
Brand
Product/Service Attributes
Relationship
Image
Internal Perspective
Operations Management Processes
Distribution
Production
Risk
Supply
Customer Management Processes
Acquisition
Growth
Retention
Selection
Innovation Processes
Design/Develop
Launch
Opportunity ID
R&D Portfolio
Regulatory and Social Processes
Community
Employment
Environment
Safety and Health
Learning and Growth Perspective
Human Capital
Information Capital
Organization Capital
Culture
Leadership
Alignment
Teamwork

Balanced Scorecard

Identify and Improve strategic management performance metrics

Traditional performance management tools only focus on financial outcomes and often fail to consider other important factors such as loyal customers or skilled staff. A more balanced approach is to consider non-financial performance measures in relation to Company Strategy. For example, customers might be prepared to pay more for goods and services produced from an environmentally friendly company. A Balanced Scorecard (*Strategy and Leadership,* Kaplan and Norton, 2004) gives equal weight to four different perspectives. In a single diagram you can record key financial and non-financial goals, with links to meaningful benchmarks and performance measures. The Scorecard acts as a communication and measurement tool to address strategic planning.

Balanced scorecard - the four perspectives:

- Financial
- Customer
- Internal Business Process
- Learning and Growth

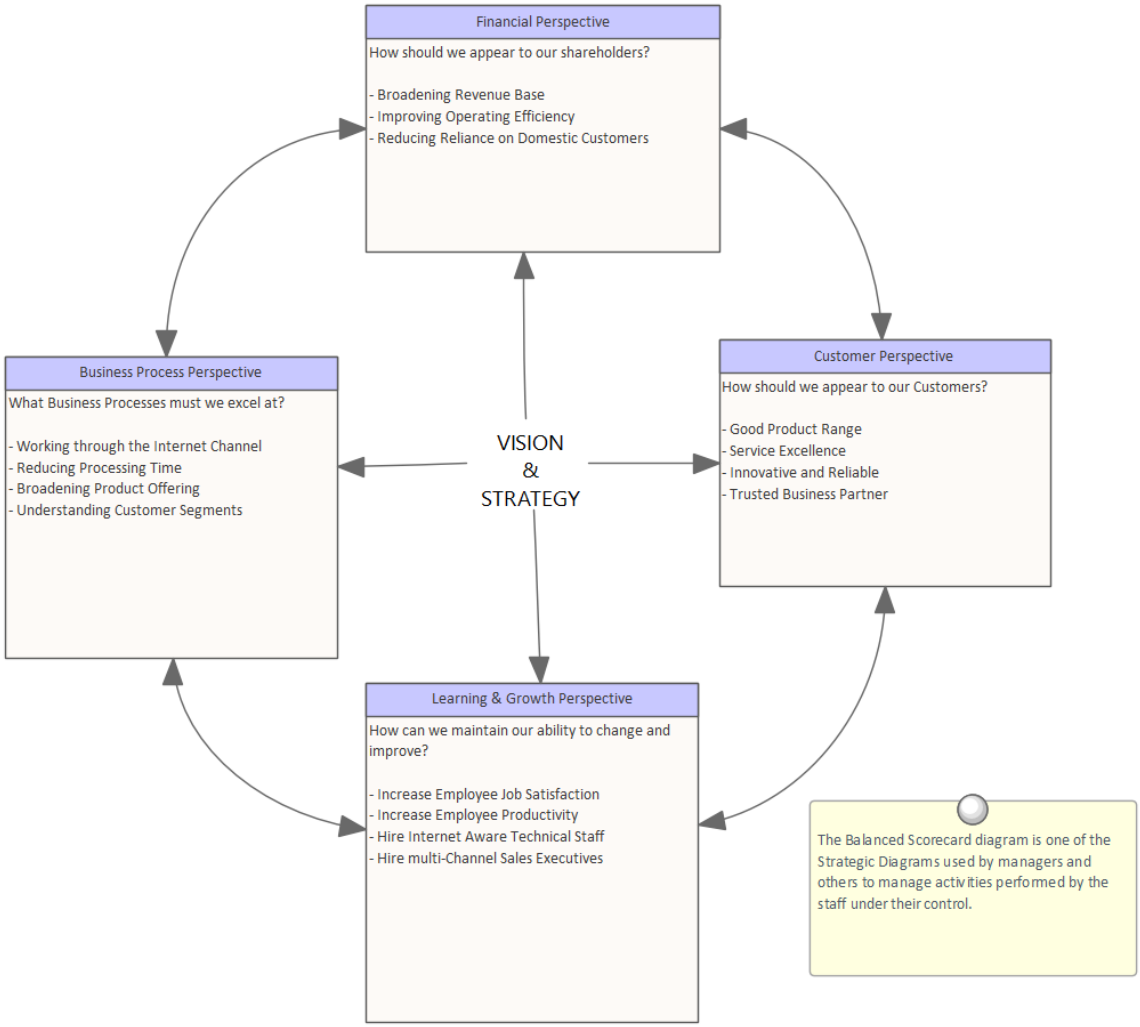
Each perspective must consider these items:

- Strategic Objectives (growth, profitability, customer satisfaction, process excellence)
- Measures (revenue growth, return on equity, number of

complaints, yield)

- Targets
- Initiatives

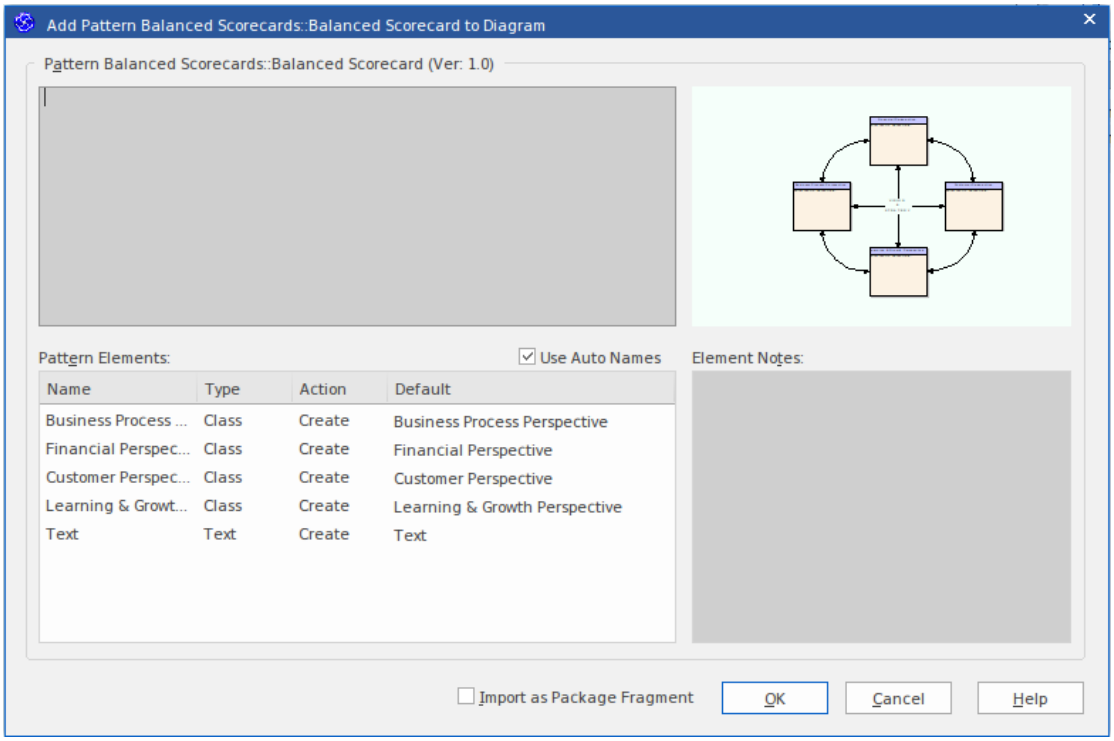
For example:



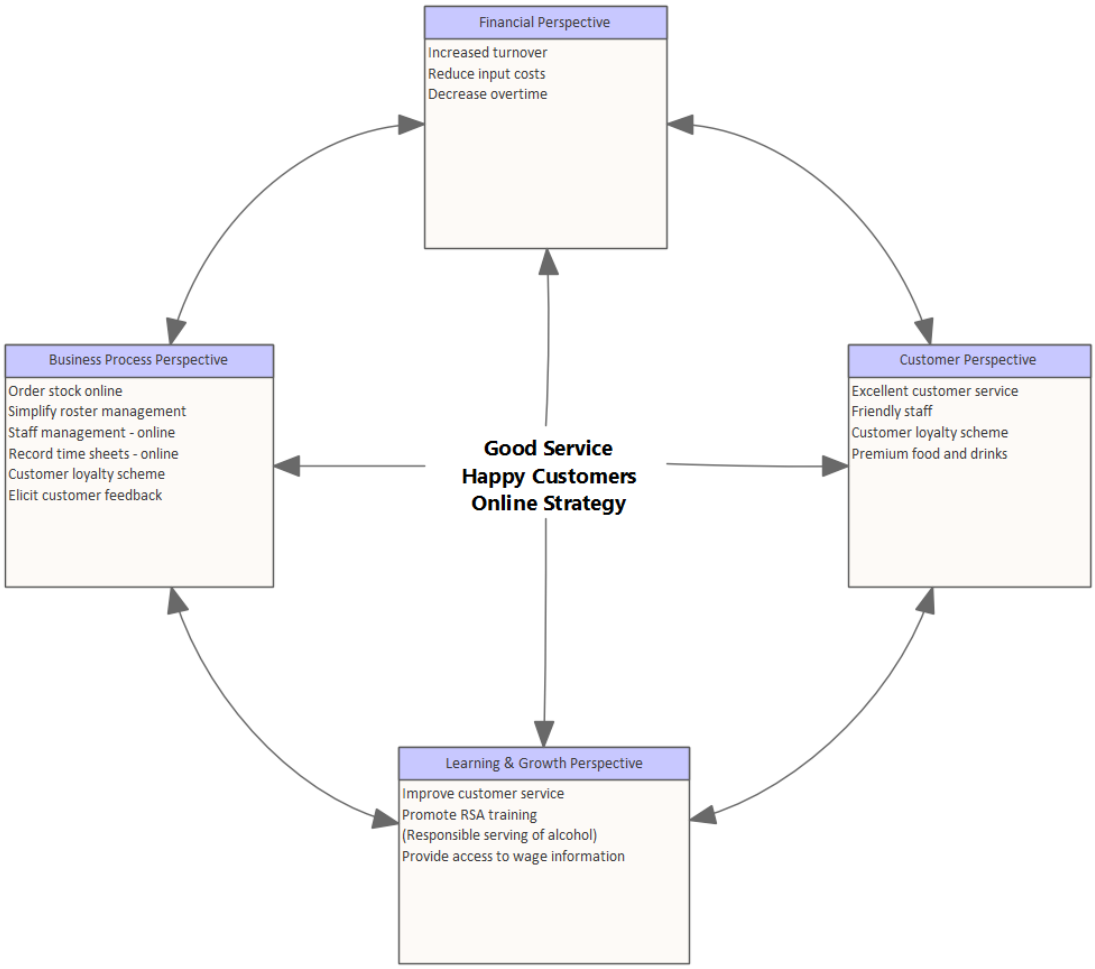
Create a Balanced Scorecard Diagram

A Balanced Scorecard diagram is very easy to create and edit. You simply use the Balanced Scorecard pattern in the Toolbox, and them type in text as notes within the Perspective elements or as separate Notes or Text elements, as described in these steps:

1. Create a new Package called 'Balanced Scorecard'.
2. Right-click on the 'Balanced Scorecard' Package and select 'Add Diagram'.
3. Create a new Balanced Scorecard diagram: enter the name - such as 'Four Perspectives' - then click on the 'Hamburger' icon, select 'Strategy > Strategy > Strategic Modeling > Balanced Scorecard', and click on the OK button.
4. In the Toolbox, expand the 'Patterns' section and drag the 'Balanced Scorecard' Pattern onto the Diagram View. The 'Add Pattern Balanced Scorecards :: Balanced Scorecard to Diagram' dialog displays.



5. Click on the OK button; a confirmatory prompt displays. Click on the Yes button to generate the diagram shown here.



6. Click on each Perspective element in turn and, in the Notes window, type in the text to describe the points relevant to that Perspective. If you prefer you can also:

- Drag a Note element or a Text element onto the diagram and type either more extensive points or more general annotations on the diagram; you can leave the Note or Text elements free standing, or link them to the relevant Perspective using a Notelink connector
- Create a Linked Document on the appropriate element and add text and graphics to that document

The diagram you have created using this procedure illustrates how the company vision and strategy is directly

tied to each of the four perspectives. Resource allocation is based on results and strategy rather than a set of arbitrary financial numbers from the previous financial year. The Balanced Scorecard approach helps to ensure that business activity is engineered to adhere to company values and meet targets. Enterprise Architect allows each perspective to be linked to supporting documentation, model information and rigorous benchmarks that verify whether strategic initiatives have been undertaken and associated targets met.

Organizational Chart

Define an Organization Structure Including Roles and Responsibilities

An Organizational Chart (Org Chart) is a graphical representation of the structure of an organization, describing the roles and responsibilities of each staff member. Enterprise Architect provides an Org Chart Pattern to speed up the creation of Organizational Charts.

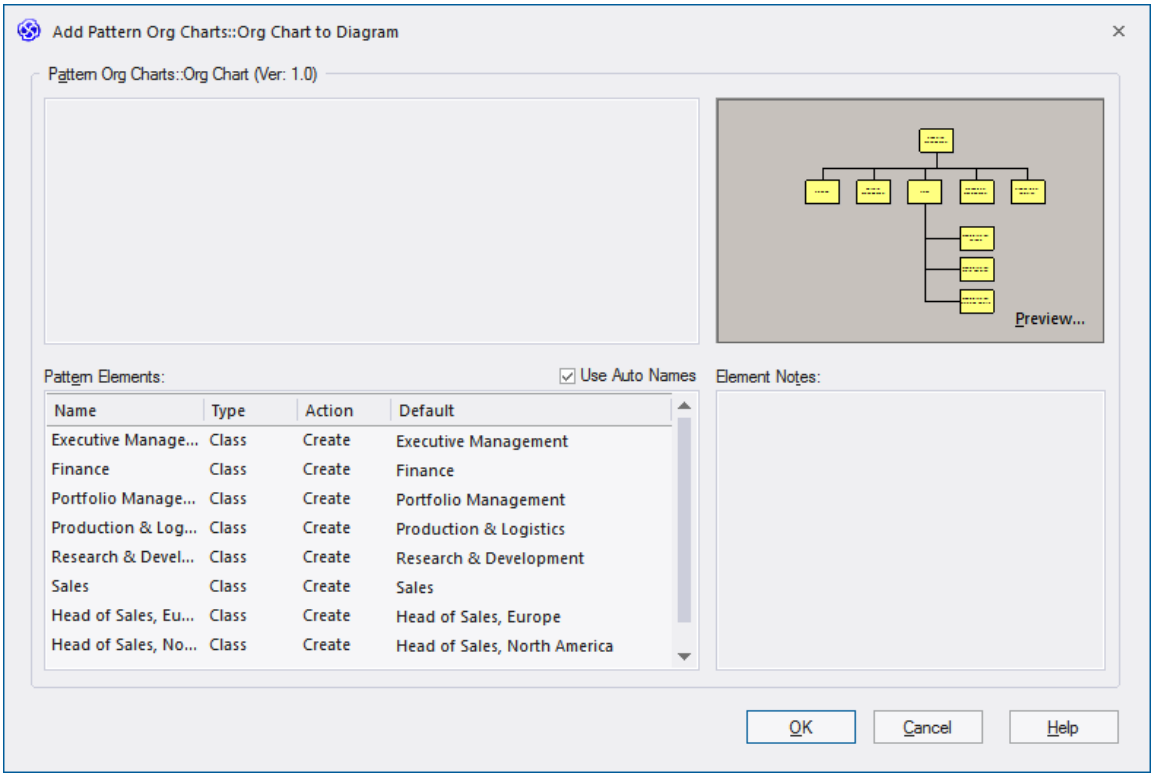
When modeling the structure of an organization it is quite common to use an Organizational Chart to describe the structure in terms of relationships between personnel or departments. Each rectangle represents a person, position or department, while each connector represents a hierarchical relationship or command structure. Enterprise Architect can store notes and Linked Document for each position to provide more detail than a traditional Org Chart.

Creating an Organizational Chart

This example illustrates how easy it is to modify the basic Pattern to suit your individual needs.

Create a new Package called 'The Org Chart'. Right-click on the Package and then select 'Add Diagram'. Create a new Organizational Chart called 'Company Structure' and click on the OK button.

From the Toolbox, select the 'Org Chart' Pattern and drag it onto the diagram. Modify the Pattern elements by selecting the ellipsis button next to each role.



Create these roles in order:

- Marketing Manager
- Sales Manager
- Accounting
- Chief Financial Officer
- Head of Logistics
- Research and Development
- Chief Technical Officer
- Chief Operations Officer
- Chief Executive Office

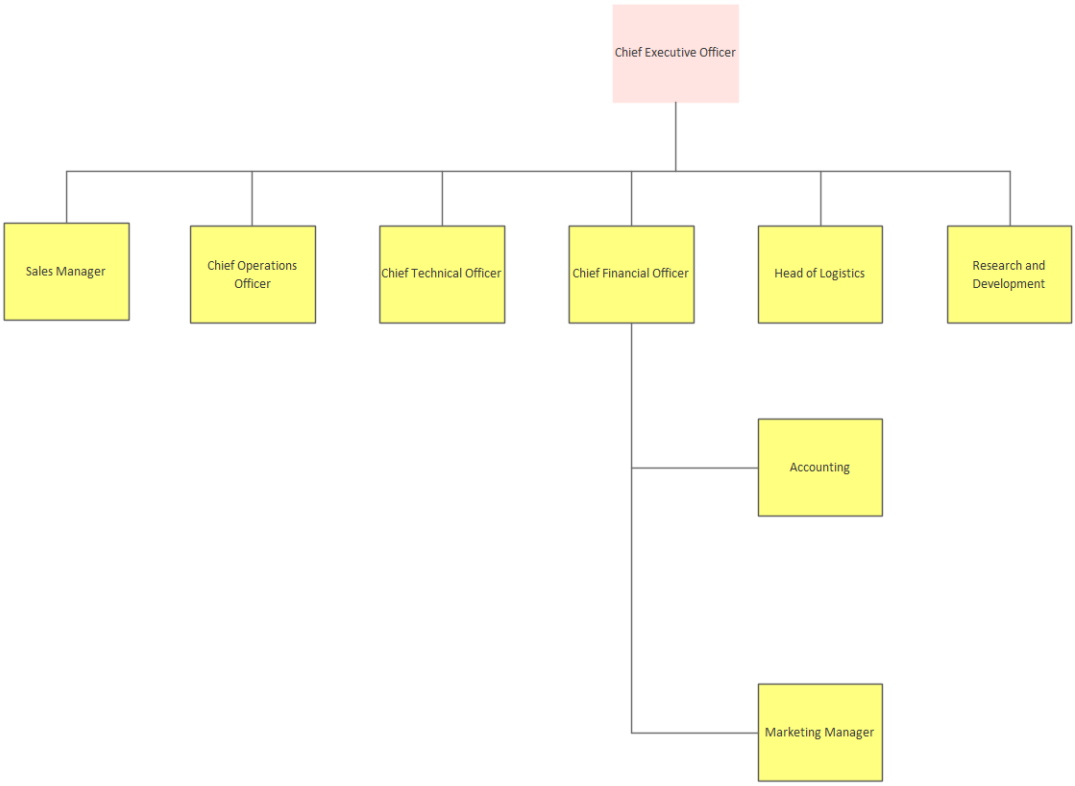
Move the Sales Manager role so that they can report directly to the CEO.

Delete the connector to the Sales Manager role. Move this role next to the Chief Operations Officer. Select the 'Reports to (Vertical Tree)' connector from the Toolbox and link Sales Manager to the Chief Executive Officer.

Select the Chief Executive Officer and change the Line and Fill Color to 'Rose' using the 'Layout' ribbon, 'Style' panel.

Organizational Chart - Completed Example

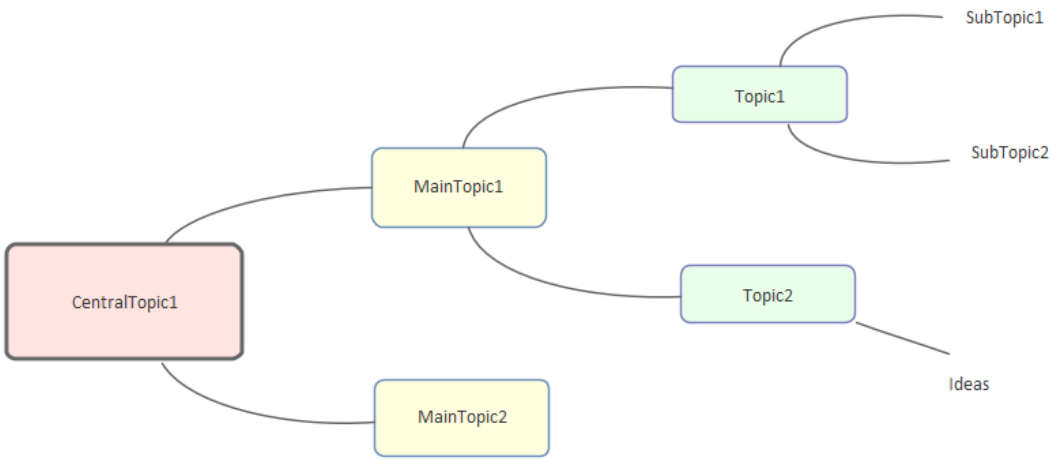
Each Role in this diagram can be linked to a position description, web page or personal CV via the element's 'Properties' dialog or using a Linked Document. Enterprise Architect can also generate RTF and HTML documentation in order to publish the Organizational Chart for distribution to staff.


Chief Executive Officer
Sales Manager
Chief Operations Officer
Chief Technical Officer
Chief Financial Officer
Head of Logistics
Research and Development
Accounting
Marketing Manager

Mind Mapping

Record, Structure and Visualize Ideas during Meetings and Workshops

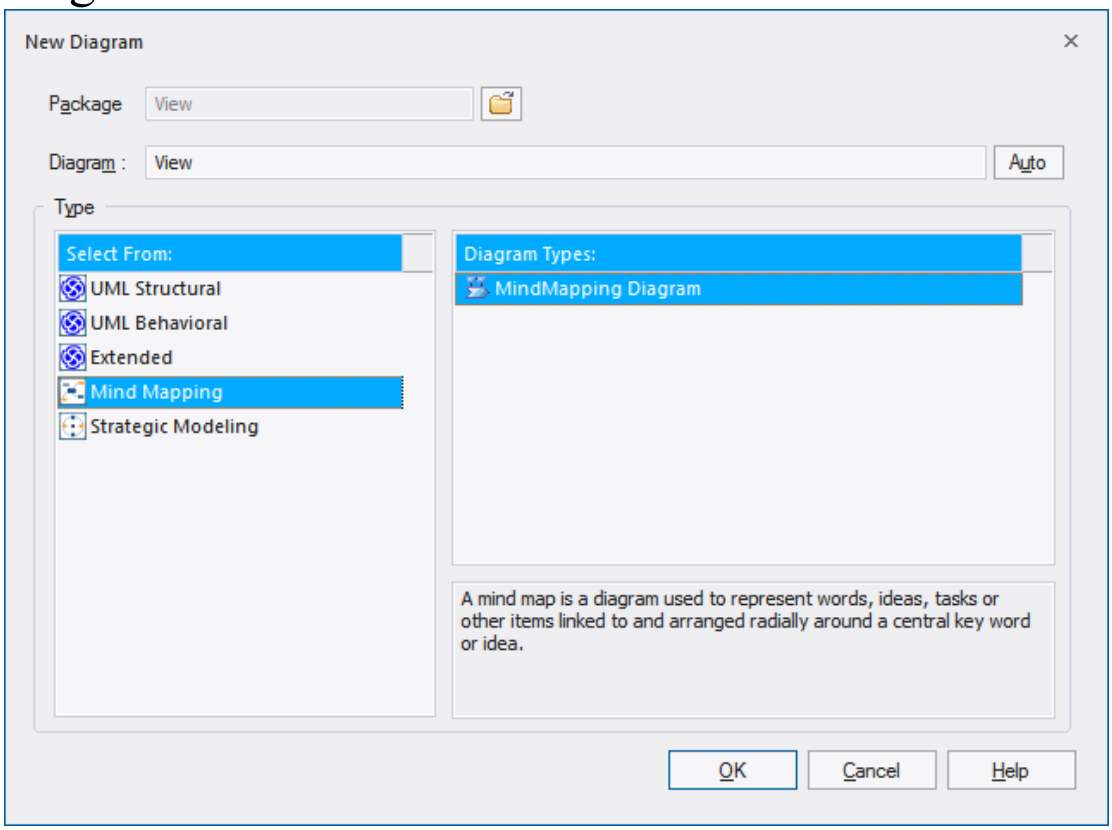
Mind Mapping diagrams offer a flexible, visual notation to convey complex information. Typically they are used for studying, collaborative discussions, problem solving, brainstorming, presenting complex ideas and decision making. Using this non-linear approach allows the information to be interpreted in many ways, producing unique and innovative solutions.



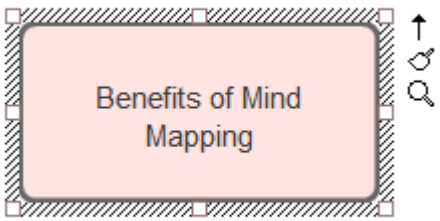
Create a Mind Mapping Diagram

- Open Enterprise Architect and create a new project file called 'StrategicModel.eap'; create a Simple View entitled 'The Enterprise', containing a Package called 'MindMapping'
- Right-click on the 'MindMapping' Package and select the

'Add Diagram' option; create a new Mind Mapping diagram called 'Brainstorm'



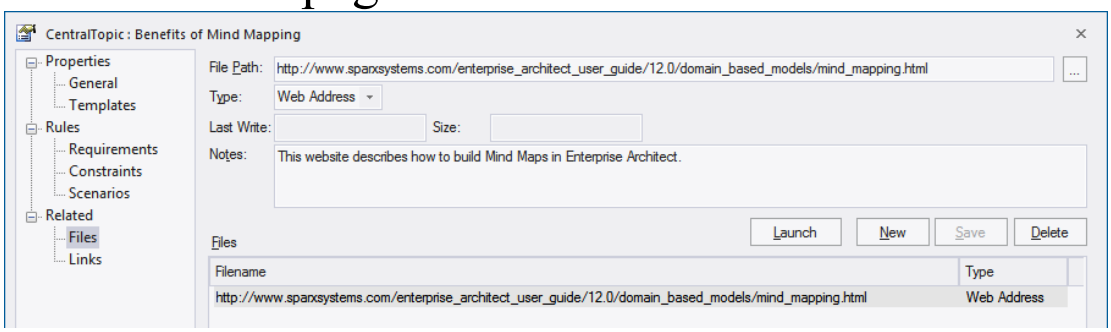
- **Start with the central theme:** Open the Diagram Toolbox and place a Central Topic called 'Benefits of Mind Mapping' in the middle of the Diagram View



- **Create associated topics:** Use the Quick Linker arrow to create three Main Topics called 'Problem Solving', 'Memory Recall' and 'Decision Making'; alternatively you can use the Relationship connector in the Toolbox to link

to related topics

- Select 'Memory Recall' and use the Quick Linker to create three associated Topics called 'Association', 'Visual Memory' and 'The Brain'
- Select 'The Brain' Topic and use the Quick Linker to create four Sub-Topics called 'Emotion', 'Language', 'Imagination', and 'Visual Processing'; your diagram should now contain 11 related topics
- **Create links to on-line resources and important metadata:** Access the 'Properties' dialog for the 'Benefits of Mind Mapping' Central Topic, select the 'Files' tab and enter the URL shown in order to access the Enterprise Architect Help topic *Mind Mapping Diagrams*; Enterprise Architect can easily link to any supplementary material, including local minutes of a meeting, relevant documents, files and web pages

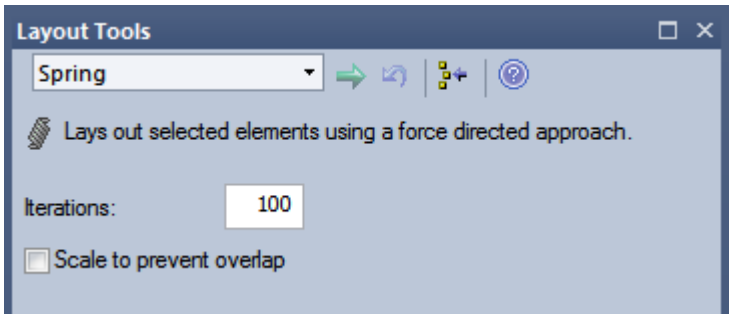


- **Make elements stand out by adding color:** If you want to draw attention to a particular topic or concept, use the 'Layout > Style' ribbon options to alter the font color of the element - for example, change the 'Visual Processing' Sub-Topic to the color blue; color can be used to structure information and enhance readability

- **Make elements different shapes** - If you prefer, make

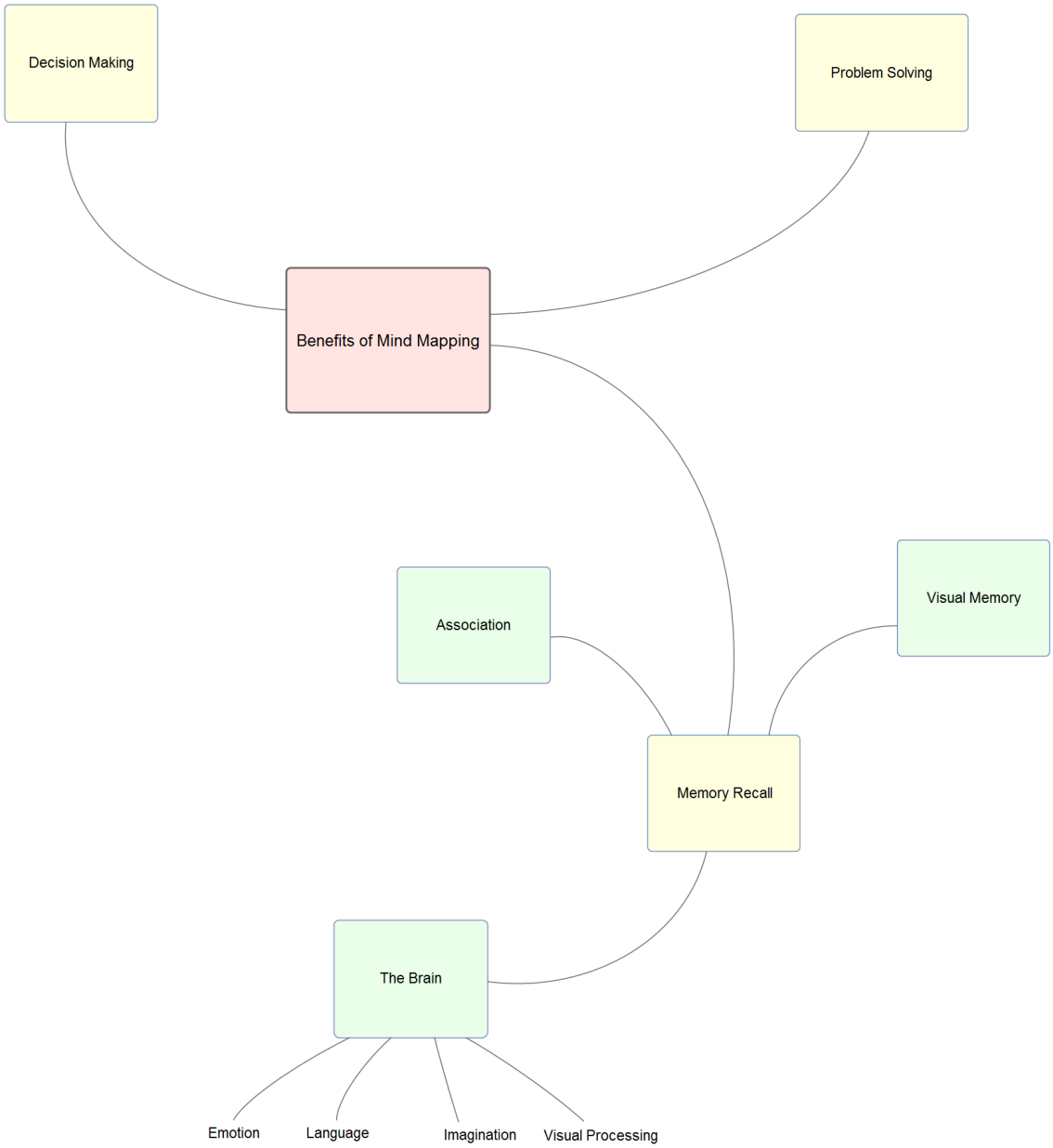
further changes to the shape and configuration of the elements using the Custom Diagram Styles, as described in the *Custom Diagram Styles* Help topic

- **Layout the diagram:** Select the 'Layout > Tools > Diagram Layout > Open Diagram Layout Tools' option to alter the appearance of a diagram - these tools provide a large number of customization options to enhance the visual layout and presentation of your diagrams; please note, Mind Mapping diagrams traditionally use curved lines



- **Adjust the height and width of your diagram:** Right-click your Mind Mapping diagram to modify the layout of the diagram, based on the options 'Align elements (left, right, top, bottom, centers)', 'Modify size (auto-size, same height and width)', 'Space evenly (Equal distance across and down)' and 'Layout selected elements'

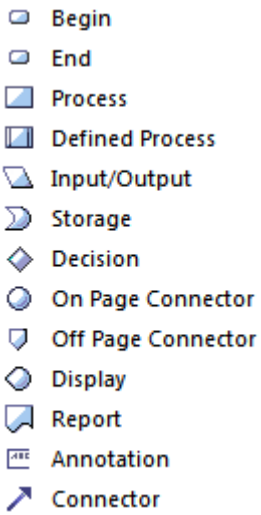
Mind Mapping Example Diagram


Decision Making
Problem Solving
Benefits of Mind Mapping
Association
Visual Memory
Memory Recall
The Brain
Emotion
Language
Imagination
Visual Processing

Flow Charts

Create and Manage Diagrams of Workflows and Processes

A Flow Chart is a graphical representation of a sequence of events, helping decision makers understand the relationship between their decisions and a given outcome. Flow Charts use a range of simple geometric shapes to represent a process, decision, storage or output.



Creating a Flow Chart

Create a flowchart for hiring staff at a Restaurant.

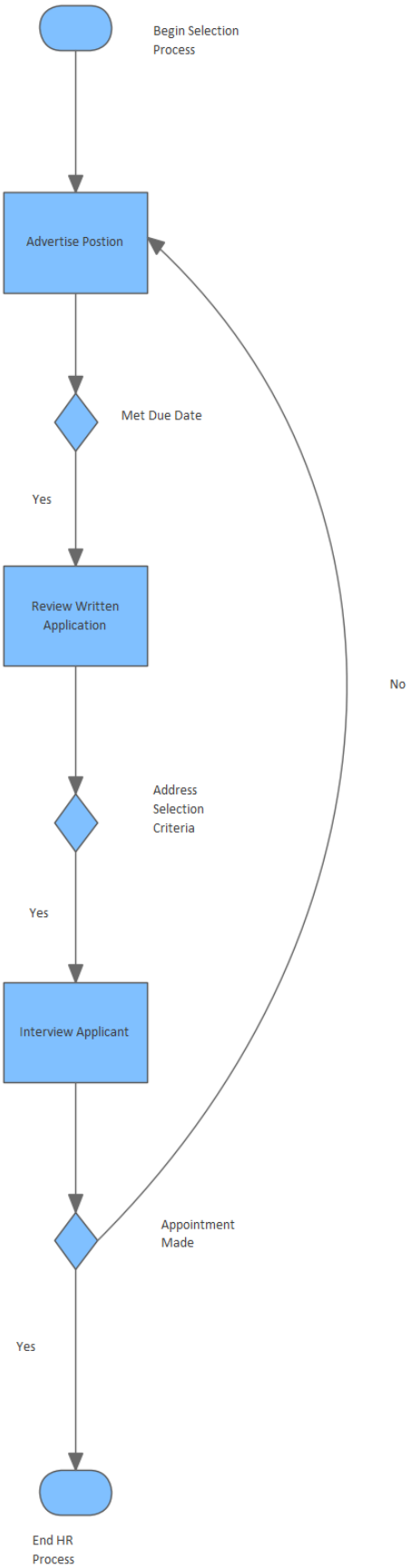
1. Create a new Package called 'Flow Chart'. Add a Flow Chart diagram from the Strategic Modeling category.
2. Place a 'Begin' and 'End' icon onto the diagram. Name the two respective items 'Begin Selection Process' and 'End

HR Process'.

3. Create two different Processes called 'Advertise Position' and 'Review Written Application'.
4. Create a Decision called 'Met Due Date'. This decision will be used to determine if a candidate will proceed to the written application stage.
5. Draw a connector between 'Begin Selection Process' and 'Advertise Position' followed by another connector between 'Advertise Position' and 'Met Due Date'.
6. Right-click on the Connector located between 'Advertise Position' and 'Met Due Date' and select the ControlFlow properties. Name the Connector 'Yes'. By labeling the Connector with the word 'Yes' we assert that if an application is received by the due date then it will be reviewed by staff in the Human Resources department. Any submissions received after the due date will not be reviewed by staff.
7. Create a Decision called 'Address Selection Criteria'. Draw a Connector from 'Review Written Application' to the Decision 'Address Selection Criteria'.
8. Create a Process called Interview Applicant'. Draw a Connector from 'Address Selection Criteria' to 'Interview Applicant' and label the Connector 'Yes' denoting that if the applicant has addressed the selection criteria then proceed to 'Interview Applicant'.
9. Create a Decision called 'Appointment Made'. For the 'Yes' response, draw a Connector to 'End HR Process'.
10. For the 'No' response, draw a connector to 'Advertise

Position'. Note: you could use a Bezier Connector (a curved line) to improve the diagram layout.

11. Select the entire diagram (Ctrl+A) and select 'Align Centers | Align Vertically'.


Begin Selection Process
Advertise Postion
Met Due Date
Yes
Review Written Application
Address Selection Criteria
Yes
Interview Applicant
Appointment Made
No
Yes
End HR Process

Requirements

Manage All Requirements Types from Elicitation through to Validation

Requirement Engineering is the discipline of eliciting, analyzing, specifying and managing requirements. A requirement is essentially a definition of a property that a system or process must be able to perform. Enterprise Architect provides tools and functionality that will assist with all aspects of Requirement Engineering from elicitation through to requirements management.

Requirement Engineering is performed differently depending on the development method being used; traditional processes such as Waterfall will prescribe that the requirements are elicited and analyzed before development work is started, whereas when iterative and incremental methods are used, including Agile, the requirements are incrementally elicited and analyzed.

Requirements also exist at different levels in a process; for example, there are high level requirements such as stakeholder needs, and low level requirements that define how a system component must function. Detailed requirements can be organized into a hierarchy culminating in a high-level requirement, so that satisfying each of the detailed requirements results in meeting the higher-level requirements and ultimately the top-level requirement. This hierarchical structure helps manage the complexity of large

systems, with thousands of requirements and many processes being developed to implement those requirements.

Enterprise Architect the first UML tool to support Requirement Engineering, and it continues to be a leader in this field with a versatile feature set supporting all aspects of Requirement Engineering, including the text-based Specification Manager, hierarchical representations, Requirements diagrams for visualizing, automatic documentation generation and Requirement Management features. Requirements can be given a status that can be conveniently displayed using color codes on a diagram, helping the Requirement Manager and others to gain a quick visual overview of the status of the requirements.

Gathering Requirements

Gathering requirements is typically the first step in developing a solution, be it for developing either a system or a process. Requirements are gathered from all parties expected to use, maintain or benefit from the solution, and are organized into groups, functional areas and hierarchies as necessary. They can be created directly within an integrated modeling tool such as Enterprise Architect, or if they have been transcribed into a spreadsheet or a requirements gathering or management tool, they can be imported into Enterprise Architect.

The management of requirements is one of the more problematic disciplines in software development, for reasons such as:

- Diverse group input into the requirements
- Organizational boundary divisions
- Tool boundary divisions
- Volatility of requirements
- Imprecision and ambiguity in natural languages

These can cause issues with:

- Traceability and
- Integration with change and configuration management systems

Enterprise Architect can reduce or eliminate these problems in Requirements Management.

Requirement Management and Enterprise Architect

Enterprise Architect is one of the few UML tools that integrate Requirement Management with other software development disciplines in the core product, by defining requirements within the model. Within Enterprise Architect, you can:

- Create and view requirements as entities and properties directly in the model, as simple text descriptions or as diagrammatic representations of the elements and their organization
- Collate the requirements in an external CSV file and then

import them into your model
- Detail Use Cases and scenarios directly in the model
- Enter standard attributes (properties) for each requirement, such as difficulty, status and type, and define your own attributes (properties)
- Trace requirements to Use Cases, Business Rules, Test Cases and analysis artifacts (using, for example, the Relationship Matrix)
- Trace and view the impact of changes on requirements (through, for example, the Traceability window) and review the changes themselves
- Create customer-quality MS Word™ and HTML reports on requirements

Notes

- All of these features are illustrated by examples in the EAExample.qea or EAExample.eap model, provided as part of your Enterprise Architect installation in the Enterprise Architect Program Files directory: ...\Program Files\Sparx Systems\EA
- If your project team are not using Enterprise Architect to manage Requirements, they can still access, use and work with them via the Cloud, from an Open Services for Lifecycle Collaboration (OSLC) client tool

Requirements

Enterprise Architect supports requirements definition for enterprise, business, software, hardware and system engineering projects, including functional and non-functional requirements. There are a number of requirement types built into the core product and new types can be added to suit any project. The UML does not formally define a Requirement element, but Enterprise Architect extends the language to provide an element that can be added directly into the repository or through the text based Specification Manager, or created on diagrams. In Enterprise Architect the requirement is treated as a first-class modeling element, and is able to participate in relationships allowing traceability to be established and a project manager or business analyst can track that a project is being designed, built and tested according to the stakeholders needs and its specifications. Use Cases can also be defined and a sophisticated editor helps you to define Scenarios that can be automatically generated to behavior diagrams, enabling the Requirements Analyst to trace to individual steps in a scenario. Using Stereotypes user stories can be modeled, and an Agile Backlog can be defined using the Feature element; as prioritization occurs these can be elaborated into well articulated requirements ready for the development team.

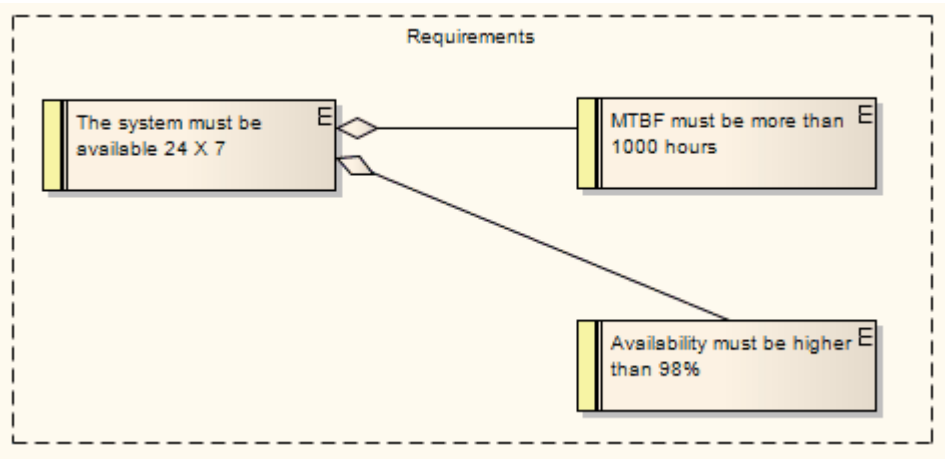
Description

As an analysis step, often it is desirable to capture simple system requirements. These are eventually realized by Use Cases.

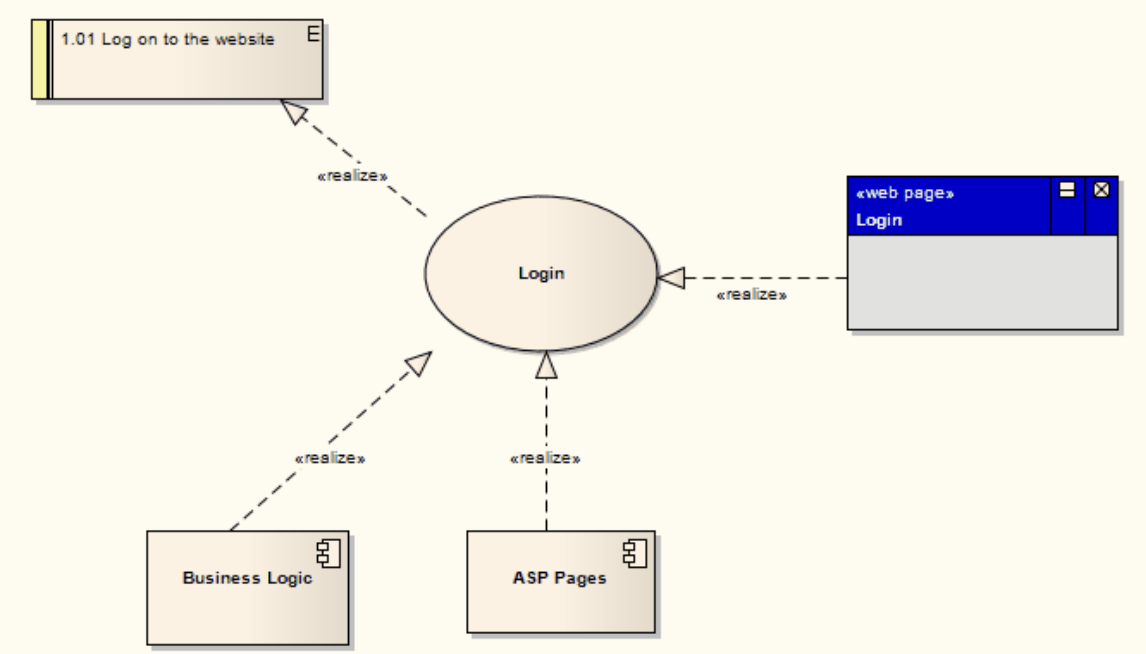
In the initial requirement gathering phase, cataloging requirements can be achieved using the Requirement extension on a Custom diagram.

Examples

Requirements can be aggregated to create a hierarchy, as illustrated by this diagram.



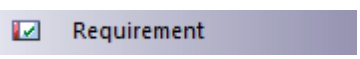
In the next diagram, a requirement that a user can log into a website is implemented by the Login Use Case, which in turn is implemented by the Business Logic, ASP Pages and Login Web Page constructions. Using this approach, you can easily model quite detailed and complex dependencies and implementation relationships.



Notes

- External requirements can be displayed with or without an identifying 'E' (for External) in the top right corner of the element; to toggle the display of this letter, select or deselect the 'Show stereotype icon for requirements' checkbox on the 'Preferences' dialog, 'Objects' page
- The colors on Requirement elements identify the status of the requirement; you change the status - and hence color - on the element 'Properties' dialog, and set the color for each status on the 'Status Types' dialog

Toolbox icon



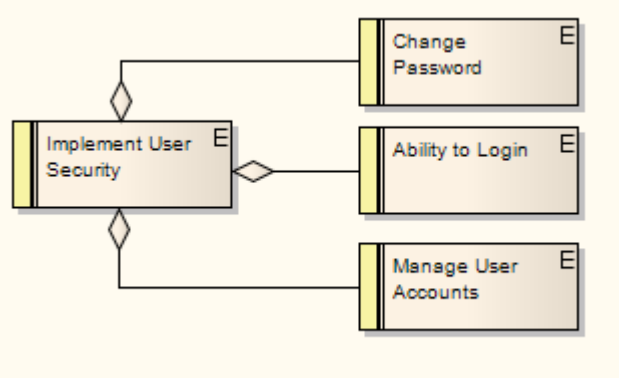
Model Requirements

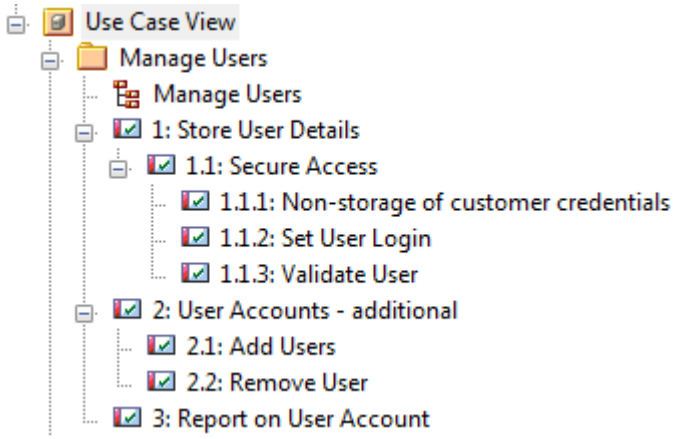
Requirements are fundamental to the success of any project, regardless of the process being used. Many organizations have traditionally stored and managed their requirements in text tools such as spreadsheets and word processors, but these requirements often remain hidden from the development process. Using Enterprise Architect's model based approach to requirements engineering turns requirements into first class modeling elements. They can be displayed on diagrams and related to the stakeholders that own them, and traces can be created connecting them to other down-process model elements such as Use Cases and application Components.

Properties such as status, phase, complexity and difficulty can be assigned to each requirement, which helps you to manage them easily.

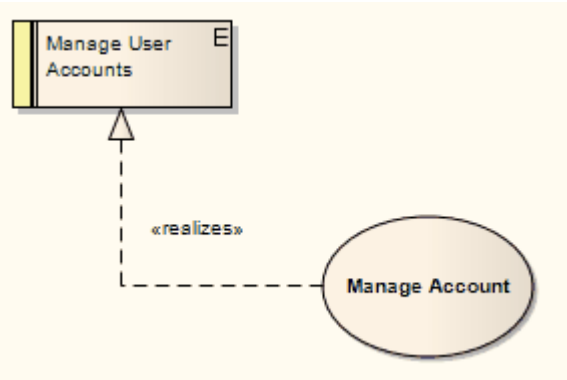
Features

Feature	Detail
Represent Requirements	In Enterprise Architect, a requirement can be modeled as an: • External Requirement - an expectation of the system or process, what the system or process must provide,

	modeled as an *element*; for example, a business requirement or a stakeholder request - Requirements at this level have their own properties and are reported on separately in document reports • Internal requirement – a requirement of an existing element, what the element must do or accomplish, defined as a *property* of the element Requirements Management in Enterprise Architect is primarily concerned with Requirement elements and the elements that implement or realize them.
Requirements in the Model	Requirement elements can be grouped and organized within Requirements diagrams. The Requirement elements are connected to each other by Aggregate relationships to form a hierarchy:  It is quite usual to develop a Package of many hundreds of Requirement elements,

	arranged individually and in hierarchies of varying complexity. You can select a Package and use the 'Design > Package > Manage > Level Numbering' ribbon option to highlight the order and arrangement of the Requirements quickly and easily. This illustration shows a number of Requirements in a Package, where Level Numbering makes the order and arrangement clear:  If elements are added, moved or deleted from the Package, the numbering automatically adjusts. This numbering can also be applied in a document.
Use Cases	Requirements are implemented (realized) by model elements such as Use Cases, Classes, Interfaces and Components. There are many ways to trace either the Requirement for the feature or service

modeled by the elements, or the elements that develop the requirement, most visibly in Traceability diagrams that depict the Requirements and the model elements connected by Realize relationships. The Realize connector enables members of the project team to keep design objectives and development in tandem, and the development path and purpose clear.



The more usual realization relationship is between a Requirement and a Use Case. A Requirement can be realized by one or more Use Cases, and a Use Case can realize one or more Requirements.

Whilst a Requirement defines a condition that must be met, the Use Case is the key to defining and visualizing how that condition is met. A Use Case diagram depicts the logical grouping of actions, processes and components to achieve a

	required result, and through the use of Actor elements also defines the user and/or system roles participating in the process. Each Use Case (as a composite element) can contain a combination of child diagrams that define in greater detail how a particular activity or facility might be implemented - such diagrams include Sequence, Communication, Activity, StateMachine and Business Rule Flow diagrams. The actual implementation of each Use Case is realized by Class, Component and Interface elements organized in their own diagrams. These realizations can also be captured and viewed in Traceability diagrams, depicting the full development pathway from initial requirement through to testing and production.

Requirements Diagram

A Requirements diagram is a Custom diagram used to describe a system's requirements or features as a visual model. Each Requirement is defined as a Requirement element (a Custom element of type Requirement). The actual Requirement, as a text explanation, is the element name (short) or description (long) in the element properties.

Requirement elements can have relationships with other elements, such as other Requirements, Use Cases and Components, to illustrate how a requirement is satisfied by modeling and development. You can track the development arising from a specification or requirement using the Traceability window.

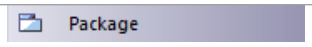
Example Diagram

Example Requirements Diagram

Requirements Diagram Elements

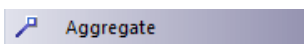
You can create Requirements diagram elements by dragging them onto the diagram from the 'Requirements' pages of the Diagram Toolbox.

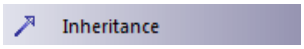
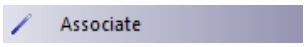
Toolbar Icon	Description

	Packages are used to organize your project contents, but when added onto a diagram they can be use for structural or relational depictions.
	A Requirement element captures the details of a system requirement.
	A Feature is a small, granular function or characteristic expressed in client-valued terms as a satisfaction of a requirement.
	An Object is a particular instance of a Class at run time.

Requirements Diagram Connectors

You can create Requirements diagram connectors by dragging them onto the diagram from the 'Requirements' pages of the Diagram Toolbox.

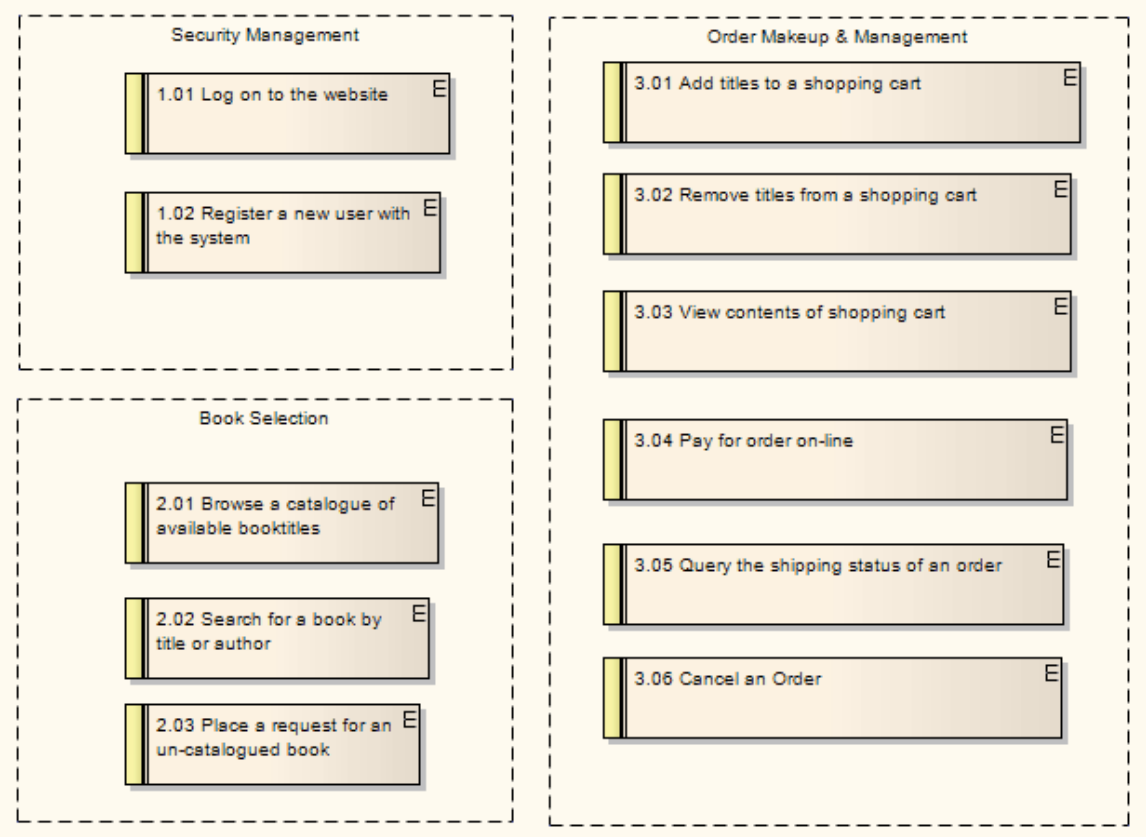
Toolbar Icon	Description
	An Aggregation connector is a type of association that shows that an element contains or is composed of other elements.

	A Generalization is used to indicate inheritance.
	An Association implies that two model elements have a relationship, usually implemented as an instance variable in one or both Classes.
	A Realizes connector represents that the source object implements or Realizes its destination object

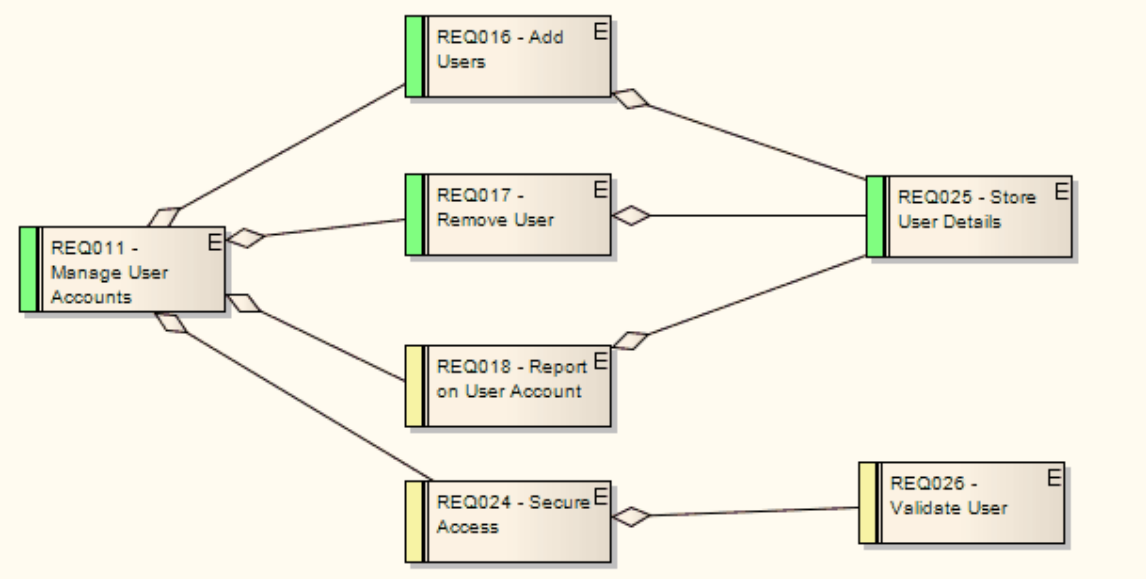
Example Requirements Diagram

These examples illustrate possible structures of a Requirements diagram. Use Cases and Components in the system can be linked back to the Requirement elements to define how a particular system requirement is met.

Example 1



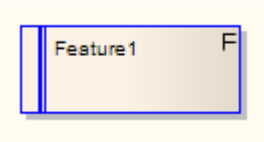
Example 2



Notes

- Change and Defect (Issue) elements resemble Requirement elements and can be coded and managed in the same way

Feature



Description

A Feature is a small, granular function or characteristic expressed in client-valued terms as a satisfaction of a requirement; for example: 'context-sensitive Help', or 'ability to reverse-engineer VB.Net'.

Features are the primary requirements-gathering artifact of the Feature-Driven Design (FDD) methodology. They define the product feature that satisfies what a Requirement element has formalized as a contractual, testable, expected deliverable (for example: requirement - 'every element must provide context-sensitive Help'; feature - 'every element provides context-sensitive Help'). One Feature might realize one or more Requirements, and one Requirement might be realized by more than one Feature.

Features also have relationships with Use Cases. A Use Case defines the interaction a user has with the system in order to satisfy one or more Requirements. The Feature identifies the facility that provides the means for that interaction.

Feature elements are non-UML and are not related to the UML elements of the same name, which are either

BehavioralFeatures (operations, or methods) or StructuralFeatures (Ports, Parts and attributes).

Feature elements are available from the 'Requirements' page of the Toolbox.

Feature elements can be displayed with or without an identifying 'F' in the top right corner of the element. To toggle the display of this letter, select or deselect the 'Show stereotype icon for requirements' checkbox on the 'Preferences' dialog, 'Objects' page.

Toolbox icon

Internal Requirements

In Enterprise Architect, internal requirements are requirements that are specific to one element. For example, an internal requirement to enable the user to log in to the system could be defined for the Login Use Case. This is a requirement of the Use Case - an action it is responsible for carrying out - and it applies only to this Use Case.

Internal requirements form the functional requirements of the system to be built. The meaning of the requirement can vary depending on which element is the host; for example, a business process requirement might mean something different to a Use Case requirement, which again might mean something different to a Class requirement.

Internal Requirements are defined on the 'Requirements' page of the Responsibility window. The significant parameters (or, in Requirement Management terms, attributes) are the Type, Status, Difficulty and Priority. Whilst you can provide a detailed description of the requirement in the 'Notes' field, there is more scope in the name ('Requirement' field) to define the nature of the responsibility. An additional field, 'Stability', indicates the probability of the requirement changing; high stability means a low probability of change.

The example Use Case also has connections to two external Requirement elements, which are system functions that the Use Case implements either in full or in part. You can convert an internal requirement into an external

Requirement element.

You can also create internal requirements for an element using an instance of the Responsibility window within the element 'Properties' dialog. A requirement created in the window displays in the 'Properties' dialog, and vice versa.

Make Internal Requirement External

Elements in Enterprise Architect have internal requirements (what they must do or accomplish). These can overlap or duplicate more formal requirements that the system in general must meet, so you might decide to make a single element's internal requirement into an external Requirement element (where the requirement can perhaps be implemented by multiple elements). You can make this conversion in one operation, using the 'Move External' function.

Access

On a diagram or in the Browser window, select an element and:

Ribbon	Design > Element > Editors > Requirements
Keyboard Shortcuts	Alt+Enter > Requirements Shift+Alt+R
Other	Double-click on element > Requirements

Change an element's internal requirement into an external Requirement element

Step	Action
1	• ('Properties' dialog) In the 'Requirements' list, right-click on the internal requirement to change to an external requirement OR • (Responsibility window, 'Requirements' tab) Click on the drop-down arrow at the end of the 'Requirement' field and, in the list, right-click on the name of the internal requirement to change to an external requirement A short context menu displays.
2	Click the on 'Move External' option. The 'Find Package' dialog displays.
3	Locate and click on the Package to place the new Requirement element in.
4	Click on the OK button. A new Requirement element is created in the target Package, with a Realization connector from the current element to the Requirement.

Notes

- When an internal requirement is made into an external Requirement element, the 'Stability' field and its value for the internal requirement are translated into the Stability Tagged Value in the external Requirement

Create Requirements

Requirement Analysts tend to work in a variety of ways, some preferring to work with lists of elements as they would in a spreadsheet, or with textual representations as in a word processor document, or visually as in a drawing tool. Enterprise Architect embraces this variation and provides functionality for Requirement Analysts to work using their preferred method for creating and editing requirements. This can be in diagrams, in lists such as the Diagram List and Package List, and in the Specification Manager. The Specification Manager is a versatile and flexible textual tool that simulates working in a word processor but that allows you to edit names, descriptions and properties of a requirement in a single interface.

Enterprise Architect also assists with creating Requirement elements by allowing them to be dragged into the model from external text documents or by importing them from spreadsheets and other documents, and it can integrate with large requirements management tools such as IBM Rational Software Architect (formerly Telelogic) DOORS.

Ways to create Requirement elements

Within Enterprise Architect you can create external Requirement elements in a number of ways, such as:

- Typing or copy/pasting a text description into the body of the Specification Manager

- Dragging a Requirement icon from the Diagram Toolbox into a specific diagram
- Generating an element within a specific Package in the Browser window
- Dragging text from a text file onto a diagram, to generate a requirement based on that text
- Importing requirements from a spreadsheet application such as Excel, via CSV
- Creating Requirement elements on the Package Browser or Diagram List for the selected Package or diagram
- Converting an internal element requirement (responsibility) into an external Requirement element, in a selected target Package
- Importing requirements from another requirements management tool, such as IBM Rational Software Architect (formerly Telelogic) DOORS (in this case via the Sparx Systems MDG Link For DOORS integration tool)

All methods that add a Requirement to a diagram or window also add the Requirement to the diagram's parent Package in the Browser window.

Create Requirement elements from text

This procedure converts a text section heading into an element name and the section text into the element's Notes text. You can use this procedure to generate elements of a

range of types; however, it is particularly useful for generating Requirements from a requirements specification document.

Step	Action
1	Open a Requirements diagram in the Diagram View.
2	Open the document file containing the text you want to generate Requirement elements from (this can be opened in any common text editing tool).
3	Highlight the required heading and associated text and drag them from the text file into the diagram. The 'Toolbox Shortcut' menu displays.
4	Navigate through the menus and select the required element type (in this case, click on Common and Requirement).
5	Enterprise Architect creates a (Requirement) element in the diagram, and displays the 'Properties' dialog with the section heading in the 'Name' (or equivalent) field and the text in the 'Notes' field; the element is also added to the diagram's parent Package.

Notes

- The Requirement element name can be simply descriptive text, with or without a manually-typed reference number; however, as requirements often have to have a unique reference for external checking, you can use the Enterprise Architect auto-numbering facility to automatically apply a numbering system with or without prefixes and suffixes - set the element type to Requirement
- Requirement elements can be displayed on a diagram with or without an identifying 'E' in the top right corner; to toggle display of this letter, select or deselect the 'Show stereotype icon for requirements' checkbox on the 'Preferences' dialog, 'Objects' page
- Requirement elements can be color coded on a diagram to indicate their status

Requirement Properties

Requirement properties differ slightly from the properties of other elements; they include information related to the difficulty and priority of the Requirement. The 'Notes' field is also important, as it describes precisely what requirement the element represents. Requirement naming requires careful consideration and could reflect either a categorical naming convention, or simply a loose English description of the Requirement.

Access

On a diagram or in the Browser window, select a Requirement element and:

Ribbon	Design > Element > Editors > Properties Dialog
Keyboard Shortcuts	Alt+Enter
Other	Double-click on Requirement element

Fields

Field	Action
Name	The name of this Requirement, which could include numbering, plain English text or some other formal specification. If you have set autonaming and autonumbering, the field contains the auto-counter text.
Notes	The description of this Requirement.
Stereotype	Either type in the name of a stereotype to be assigned to the Requirement, or click on the ... button and use the Stereotype Selector to locate and select the stereotypes to assign.
Alias	An alias to be used for this Requirement. If you have set alias autonaming and autonumbering, this field contains the auto-counter text.
Status	The current status of this Requirement.
Version	The version of this Requirement.
Phase	The phase of this Requirement.

Language	Click on the drop-down arrow and select the appropriate programming language, or the <none> option.
Filename	Click on the ... button and select the appropriate file location for the Requirement.
Difficulty	An estimate of the difficulty in meeting this Requirement; select from: • Low • Medium • High
Priority	The relative importance of meeting this Requirement compared to other Requirements; select from: • Low • Medium • High
Package	The name of the Package that contains the Requirement element.
Complexity	Click on the drop-down arrow and select the appropriate development complexity (used for project estimation). You

	normally select from three levels: • Easy • Medium • Difficult You can change your user defaults to add 'Extreme' and 'Unknown' to this list of options.
Created	Read-only field specifying when this Requirement was first created.
Modified	Read-only field specifying when this Requirement was last changed.
Key Words	A set of words that could be used to index or define the subject of this Requirement.
GUID	The globally unique identifier (GUID) of the Requirement element.
Author	The author of this Requirement.

Notes

- In Requirement Management tools and texts, the

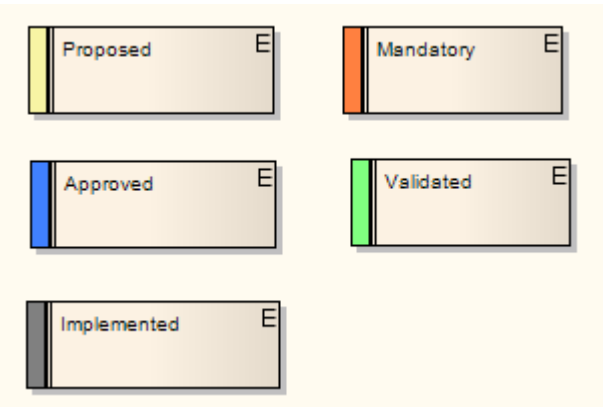
characteristics of a requirement are commonly called 'attributes'; however, in UML the term 'attribute' refers to a different type of feature, and the Requirement characteristics are defined as 'properties' - in this Enterprise Architect documentation, the term 'properties' is used

- In a project, it might be necessary to define more information in a Requirement than is provided by the standard properties; for more information on extending the Requirement properties, see the *Extend Requirement Properties* Help topic

Color Code External Requirements

External Requirement elements can be color coded on a diagram, to provide quick visual cues indicating the status of a requirement. The color code requirements use these default conventions:

- Yellow for 'Proposed'
- Blue for 'Approved'
- Green for 'Validated'
- Orange for 'Mandatory'
- Black for 'Implemented'



You can change these colors, and add or remove status types, using the 'Status Types' dialog.

Access

Ribbon	Start > Appearance > Preferences > Preferences > Objects > Show status colors on diagrams

Enable color coded external requirements

Step	Action
1	Select the 'Show status colors on diagrams' checkbox to enable the status of external requirements to be represented by color coding.

Extend Requirement Properties

A project might apply further properties to a requirement, such as cost, lateness penalty or risk to the business if not met. You can add these properties to specific Requirement elements, or configure them to be automatically available in all Requirement elements on creation, using Tagged Values. (These are sometimes referred to as User-defined attributes.)

Extended element properties are not visible unless you open the 'Tags' tab of the Properties window for the element. Alternatively, you can display the additional properties on the element image on its diagrams.

Add Tagged Values to Existing Requirements

To add a property to a single Requirement as a Tagged Value, simply click on the Requirement, display the 'Tags' tab of the Properties window, and enter the name of the property as the tag name and the value of the property as the tag value.

It is likely that any property you add to one Requirement would also apply to others. You might therefore use a predefined Tagged Value Type to identify your Requirement property, so that you can select it whenever required. The predefined Tagged Value Type also enables you to define specific values for the Tagged Value.

If the appropriate predefined Tagged Value Type does not

exist, a Technology Developer can create it to add to the structured tags, reference tags, or customized tags collections.

Configure Requirements to be Created with Extended Properties

If it is necessary to create all Requirements with the same extended set of properties, you can create a Requirement Template diagram and either create a special Requirement that defines those properties (as Tagged Values), or drag an existing Requirement with those properties onto the diagram. You then set the Requirement Template diagram as the template for all new Requirement elements, so that those new Requirements automatically have all of the properties you want.

However, this then excludes other Requirement element formats, including the standard Requirement format. If you want to use another Requirement format, you have to replace or cancel the current Template. Alternatively, you can create a Profile.

A Profile also defines exactly what a new Requirement element should contain, and how it should display in diagrams. However, a Profile is a collection of alternative element definitions, so it does not override the default Requirement format, nor does it prevent you from defining several different types of Requirement element. You can

therefore have separate and parallel definitions of elements for business requirements, system requirements, project requirements, or any other category of requirement you decide to work with.

For information on importing and using existing Profile files, see the *Using UML Profiles* Help topic. For information on creating new Profiles, see the *Developing Profiles* Help topic.

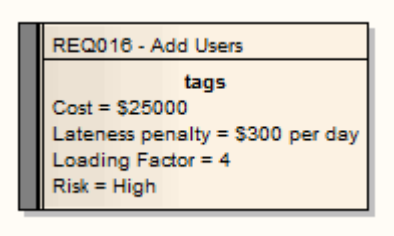
Display Tagged Values On Diagrams

If you have extended the properties of a Requirement, you might want to make those properties visible in the Requirement elements in your diagrams, by switching on display of the element tags compartment.

You can do this in one of three ways:

- To display the tags compartment on all elements on a diagram, double-click on the diagram background and select the 'Elements' tab of the diagram's 'Properties' dialog; select the 'Tags' checkbox and click on the OK button
- Alternatively, to show the tags compartment on all elements in the diagram, click on the diagram background, press Ctrl+2 to display the Properties window (if it is not already visible), and then click on the 'Compartments' tab of the window; select the 'Tags' checkbox
- To display the tags compartment on a specific element on a diagram, regardless of the compartment setting for the diagram, right-click on the element and select the 'Compartment Visibility' option; select the 'Tags' checkbox in the 'Show Element Compartments' panel of the 'Compartment Visibility' dialog, and click on the OK button

The Tagged Values are then displayed in the Requirement element on the diagram.


REQ016 - Add Users
tags
Cost = $25000
Lateness penalty = $300 per day
Loading Factor = 4
Risk = High

Connect Requirements

Aspects

Aspect	Detail
Abstract	A Requirement element can be connected to other Requirements, most commonly using Aggregate relationships to form a hierarchy of requirements. Requirements are also connected to other types of element, most commonly Use Cases, by Realize or Implements relationships. These relationships are very important, both in identifying how the Requirements are organized and used in the model, and in tracing the development from the Requirements throughout the model. Both of these tasks are very simple in Enterprise Architect, because once a connector on a Requirement exists, Enterprise Architect automatically lists the Requirement in the: • Traceability window (an important tool in examining the role of Requirements in the model)

	• Specification Manager • 'Requirements' tab of the target element 'Properties' dialog • 'Links' tab of the Requirement element 'Properties' dialog • Responsibility window • Relationships Window • Dependency and Implementation reports • Standard document output The connector itself is also listed in the 'Links' tab of the target element 'Properties' dialog, and in the Relationship Matrix. There are, therefore, many ways to locate, view and track Requirement relationships.
Connect On Diagram	Relationships can be created on a diagram by clicking on the appropriate connector icon from the Requirement and Common pages of the Toolbox, clicking on the source (originating) element, and dragging to the target element. If you are connecting elements in different Packages, you can drag elements from the Browser window onto a common diagram and set up the relationships there.

Quick Generation Of Realize Connector	You can quickly generate a Realize connector by dragging an existing Requirement element from the Browser window into a diagram, over the element that implements the Requirement (usually a Use Case). Enterprise Architect interprets this as a request to create the Realize connector and does so automatically. The Requirement element is not added to the diagram. However, if you subsequently drag the Requirement onto the diagram the connector is already in place.
Connect Off Diagram	You can also connect a Requirement element to other elements without necessarily having the elements on the same diagram, or having a diagram open. Use the Relationship Matrix to create relationships for requirements; this is a convenient way of quickly building up complex relationships and hierarchies.

Import Requirements and Hierarchies in CSV

You can import Requirements from a spreadsheet application in CSV format. Before doing this you must create a CSV import file specification that:

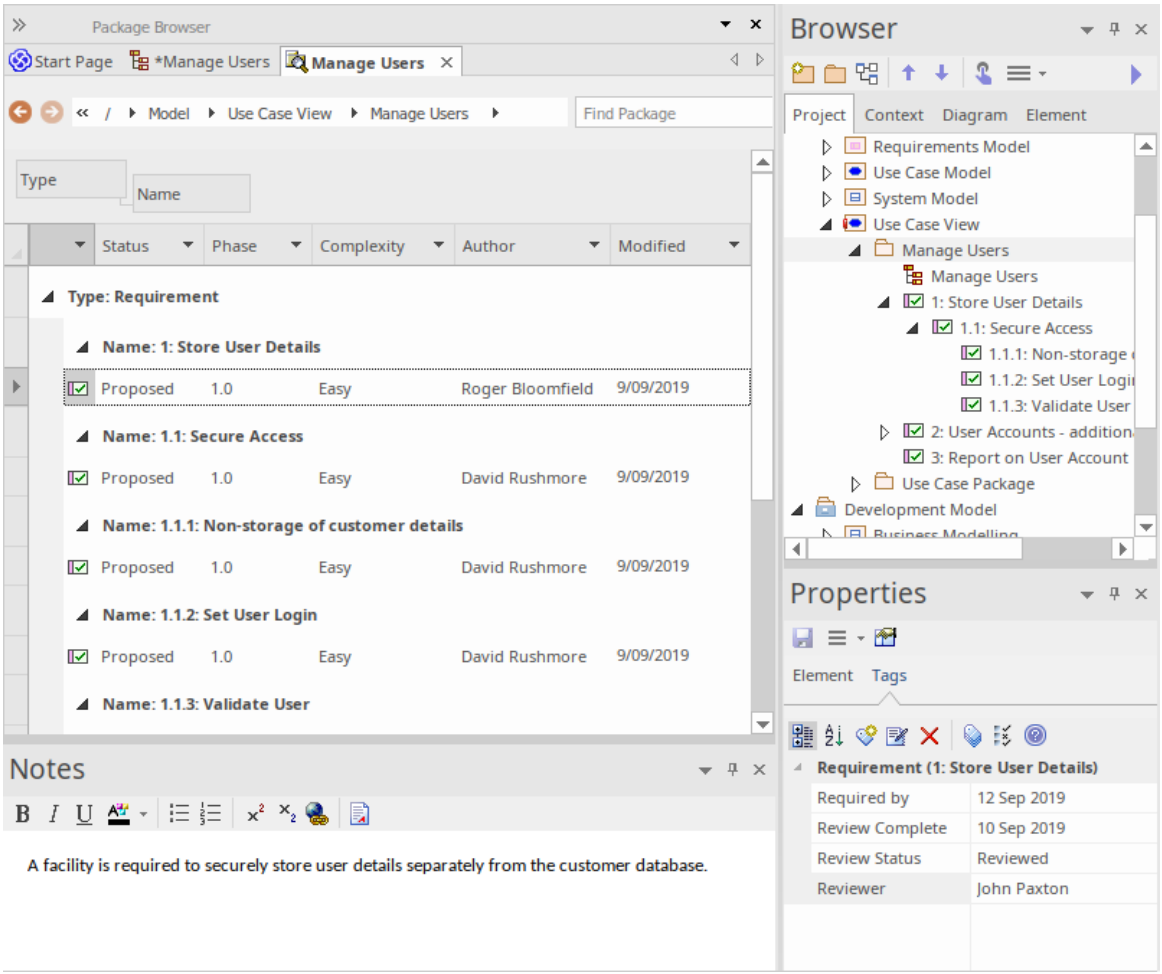
- In the 'Default Types' field has the value 'Requirement Package' to import requirements and a Package structure to contain them
- Has the 'Preserve Hierarchy' checkbox selected
- Identifies the data fields on the spreadsheet that are to be translated into Enterprise Architect, in the order in which they are plotted across the spreadsheet
- Is to operate on a spreadsheet containing the 'CSV_KEY' and 'CSV_PARENT_KEY' fields (which, if not generated by a CSV export from Enterprise Architect, you have added and populated yourself)

This enables you to import the individual and grouped requirements from the spreadsheet into Enterprise Architect, and to reconstruct the hierarchies of Requirements in the target Package in the Browser window.

Manage Requirements

Requirements Management is the process of maintaining requirements and ensuring they are traced, prioritized, assigned to phases, iterations or sprints in a development life cycle, and also ensuring those changes are communicated effectively to stakeholders. Enterprise Architect has built-in functionality that allows you to assign properties to each requirement for management purposes, such as status, priority, phase, difficulty and detailed notes. This model-based approach to requirements management allows the requirements to be traced to up-process elements such as stakeholders and goals and to down-process elements such as Use Cases and application components. Requirements Managers can work in a variety of ways and Enterprise Architect allows them the flexibility of viewing requirements as visual elements in diagrams or in lists and textual representations. The Specification Manager and list views allow the requirements metadata to be visualized and edited in a single interface. There is also a versatile chart and graph facility where the requirements metadata such as status and priority can be displayed in compelling charts and graphs. For example, a Pie chart could be created showing the percentage of elements with particular statuses.

Example



This display shows the position of the Store User Details Requirement element in the model, and how it relates to other Requirements (Browser window); the default characteristics of the Requirement (Package Browser) and the extended characteristics ('Tags' tab of the Properties window), and a detailed description of the Requirement (Notes window). You can configure some of these windows to display more information, and/or use other windows and facilities.

View Requirements

Use these windows and facilities to: locate and list Requirement elements in the model; add, move and delete the elements; display and edit the properties and characteristics of individual elements; and generate reports on Packages or specific elements.

- Project tab of the Browser window - shows the content and structure of your model
- Specification Manager - shows Requirements (and other element types) in a simple text format, and helps you to create and manage these elements
- Diagram List - lists the elements in a diagram, filtered and sorted according to the settings you define; shows all or selected default properties of each element
- Package Browser - lists the elements in a Package, filtered and sorted according to the settings you define; shows all or selected default properties of each element
- (Requirements) Diagram - shows the arrangement of a group of Requirements, and can show whether the elements are in the same Package or different Packages
- Model Search - enables you to locate Requirements in general in the model, or specific Requirement elements, according to the search criteria you use
- Model Views - enables you to maintain links to commonly-used elements, and to rapidly show developments and changes in (Requirement) Package

contents through either reports or slide shows of selected diagrams

- Properties - shows every standard property of a selected element, whether updated by the user or maintained automatically by the system
- Tagged Values - shows extended properties of a selected Requirement element
- Element tab of the Inspector window - shows every added-on property, such as attributes, operations, Tagged Values and constraints
- Notes - displays the detailed description of a requirement, and any other additional information recorded on the requirement

Trace Use of Requirements

Having investigated the representation of Requirements in your model, you could review either how they have been used to direct development through the model, or how a particular development was initiated. The windows and facilities you might use to follow development from Requirements are briefly described here; detailed information is also available in the Traceability topics.

The significant feature in tracing Requirements and development is the connectors between the elements.

Facilities

Facility	Detail
Relationships Window	Using the Relationships window, you can quickly identify every relationship of which a selected Requirement element is a member and the partner element in that relationship, whether or not the relationship is visible in the current diagram. If the partner element is not in the diagram, you have the option of adding it.
Traceability	The Traceability window is a very useful

Window	tool in showing chains of relationships that include the selected element. The window can show, for example, that: • Requirement A is realized by a Use Case X, and • Use Case X also realizes Requirement B, and • Requirement B in turn is also realized by Use Case Y You can control the type and extent of these relationship chains. As the system checks the connectors and partner elements of every relationship within the limits you impose, if you specify broader limits the system can take some time to produce the final results.
Relationship Matrix	The Relationship Matrix is a significant tool in mapping the relationships between the Requirements elements in a Package and other elements in either that Package or a different Package. Where a relationship is missing, you can add it; if an existing relationship is misplaced, you can delete it.
Requirements tab,	On the 'Properties' dialog for elements other than Requirements - particularly

Properties dialog	Use Cases - the 'Requirements' tab shows all internal requirements of the element (and, where an internal requirement has been converted to an external Requirement element, that element).
Responsibility Window	The Responsibiliy window - as for the 'Properties' dialog - shows the internal requirements of the selected element, and the scenarios and constraints under which the requirements are being realized.
Validation	It is useful to review the way that you have modeled your requirements, to check that they are correctly set up and connected to other elements. The Validation facility has a number of configuration options for validating various aspects of model development, as well as an option for specifically validating Requirements Management. This can reveal, for example, which of your Requirement elements do not yet have a Realization connector.

Manage Requirement Changes

Because requirements are statements of what a system or process must do or provide, they have a great impact on the modeling and development of the system. A new requirement might initiate an extensive program of work, and changes to or removal of that requirement can therefore have a major effect on the model. Issues concerning requirements, and changes to Requirement elements, must both be carefully managed.

The first steps in managing changes to requirements would be to raise specific Issue and Change request items against the Requirement element. You could monitor the appearance of these items using the filtered searches of the Model Views. You might then review the Requirement properties and/or its relationship hierarchies. During model development, you might capture periodic Baselines and use these to review the changes and, if necessary, roll them back to a previous point. You might also use the Auditing facility to monitor changes as they are made, and to ensure that no unauthorized or potentially risky changes are being made in the model.

Facilities

Facility	Detail

Changes and Issues	A change is, very broadly, an item defining an addition or alteration to a requirement. An issue identifies either a failure to meet a requirement, or a risk in meeting the requirement. Changes and issues can arise in development at a number of levels, being raised for problems that apply system-wide down to within a specific element. There are two mechanisms that can be used to identify a change or issue, and the work required to resolve it: • Change and Issue (or Defect) elements - structured comments that identify a problem at system-level, although they can also be attached to a specific element from which a problem arises. Both types of element resemble the Requirement element, and can be linked to one or more other elements that have to be reviewed, with relationships such as Association, Dependency and Realize. The two types of element can also form hierarchies or groups, where complex problems arise • Maintenance items raised against a specific element, and recorded for that element in a Maintenance window.

	Maintenance items enable the distinction between Defects (a failure to meet a requirement) and Issues (a risk factor that might affect satisfying the requirement). They also include Tasks, which record work items associated with the element Maintenance items are very specific, but if there is a possibility of an item having a wider impact on other elements or the system in general, you can translate the item into a Change or Issue element, or any other type of element that best identifies the problem and its solution.
Model Views	Model Views are very useful for trapping changes and issues in the model, especially on Requirements. You can set up searches to identify the appearance of new Change or Issue elements, or to detect changes in the properties of the Requirement elements themselves.
Baselines	A Baseline is a snapshot of a Package or a model branch at a particular point in time, which you determine. You can use the Baseline as a distribution mechanism for changes to the model, but the main use is to enable you to compare the

	current model with a previous stage, and detect what changes have been made since the Baseline was captured. If you do not want a change to remain in the model, you can roll the affected elements back to the state they had in the Baseline. Therefore, if you maintain your requirements in a specific Package or branch, you can capture Baselines of the Package and ensure that changes conform to your change management process or, if not, can be reversed.
Auditing	The Auditing facility enables you to capture any changes made to your model within the selection criteria that you define. You can, for example, configure the Auditing facility to specifically record changes to Requirement elements. As auditing is continuously monitoring, you can detect changes as they are made, and verify that they are acceptable. You can also store the log of changes, and review it later on. Note that you cannot reverse the changes automatically, as you can with Baselines. You might therefore use Auditing to identify changes to investigate more fully and - if necessary - reverse in a Baseline

	comparison.

Report on Requirements

Enterprise Architect provides two report generation facilities that help you to output document reports and web reports on your model structure and components.

The document reporting facility is especially comprehensive, and contains a number of features that provide particular support to Requirements Management:

- A requirements report template that extracts details of external requirements in the model; you can copy and tailor this template for your particular requirements
- Options in the Specification Manager, Diagram List, Package Browser and Model Search to generate reports on selected (Requirement) items from the collected information
- The Implementation Report, which lists for a selected Package the elements that require implementers, together with any source elements in Realize (Implements) relationships with those elements
- The Dependency Report, which lists for a selected Package any elements that are dependent on another element for their specification; for example, a Use Case derives its specification from the Requirement that it realizes

Threat Modeling and Cybersecurity

Any project, development or system can encounter numerous kinds of physical, biological or electronic hazard, which can be identified and scrutinized in a model. You can perform such modeling using the Risk Taxonomy feature. However, as businesses and processes around the world grow increasingly computerized and distributed, the target at biggest risk is the storage and flow of electronic data, the greatest threats are deliberate damage to or infiltration of the data, and the growing business of identifying and mitigating such threats is Cybersecurity.

Enterprise Architect supports the evaluation of Cybersecurity within your organization by providing the Threat Modeling facility, based on the STRIDE Methodology (discussed later in this topic).

Threat modeling overlays process modeling in areas such as UML Activities, BPMN Processes, ArchiMate Processes and Data Flow diagrams. You use the Threat modeling features to encompass areas of the process model and evaluate the threats that might exploit weaknesses in the process. The Threat model depicts the processes, data stores, external entities and their connecting data flows in the business or other system, in order to visually illustrate the potential vulnerabilities of the system. Threat modeling aims at identifying threats that can harm electronic assets, and ensuring that adequate controls to mitigate these threats are covered by security requirements. In essence, Threat

modeling is a form of Gap Analysis geared to identifying missing security requirements. The Threat modeling exercise identifies entry and exit points of the system under development that an attacker can exploit. It provides the development team with a perspective of the system from the viewpoint of an attacker or hostile user. It also provides the team with the necessary information to design and test overall project trade-off decisions, by providing insight into the areas that require further investigation from a security aspect.

Threat Modeling:

- Brings a solid foundation for building secure and safe solutions addressing confidentiality, integrity and availability
- Proactively identifies potential security threats and addresses them prior to production
- Identifies vulnerabilities in an existing solution

A Threat model is encapsulated in a Trust diagram, a specific version of a Data Flow diagram. From the diagram, potential threats are identified and, for each threat, mitigations are proposed. In some cases, the mitigation is a change in the design itself, in which case the new or changed elements must be analyzed in an additional iteration. When the mitigations have been implemented, the product or service is validated against the Threat model to ensure that the mitigations work and that design functionality and performance are above standard. If the design has serious security issues, revisiting the design and the Threat model would be appropriate. You can use Threat

modeling to shape your application's design, meet your company's security objectives, and reduce risk.

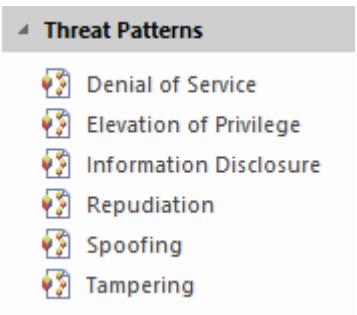
Accessing the Threat Modeling Facilities

Pattern	Method
Model Pattern	Select the parent Root Node, View or Package in the Browser window and: 1. Click on  in the top right corner of the Enterprise Architect screen. 2. Select Management > Threat Modeling. The Start Page 'Create from Pattern' tab (Model Wizard) displays, showing the 'Threat Modeling Perspective' and the 'Threat Model with Multiple Trust Boundaries' pattern. Click on the Create Model(s) button to load the pattern into the selected Package.
Diagram Template	Select the parent View, Package or element in the Browser window and either: • Click on the  icon in the Browser window toolbar, or • Right-click and select the 'Add

	Diagram' (for View or Package) or 'Add | Add Diagram' (for element) menu option, or • Select the 'Design > Diagram > Add' ribbon option The 'New Diagram' dialog displays. 1. Provide a name for the diagram and, in the 'Type' field, click on the drop-down arrow and select 'Management | Threat Modeling'. 2. In the 'Select From' panel click on 'Threat Modeling Diagram' and in the 'Diagram Types' panel click on 'Trust diagram'. 3. Click on the OK button. The Diagram View displays ready to populate with objects, and the 'Trust Diagram' page opens in the Diagram Toolbox (see the *Trust Diagram Toolbox Page* section in this topic).
Diagram Patterns	When you open a Trust diagram, the 'Trust Diagram Modeling' page opens in the Diagram Toolbox. This is accompanied by a 'Threat Patterns' page, which contains icons to generate a Threat element and Mitigation Checklist element for each of the STRIDE Threat types (see the *Security Threats Taxonomy Based on*

	the STRIDE Methodology section, next). Click on the required pattern icon and drag it onto your diagram to generate the elements.

Security Threats Taxonomy Based on the STRIDE Methodology



Threat	Description
Spoofing	Property: Authentication Spoofing threats involve an adversary creating and exploiting confusion about who is talking to whom. Spoofing threats apply to the entity being fooled, not to the entity being impersonated. So, external elements are subject to a spoofing threat when they are confused about what or whom they are talking to.

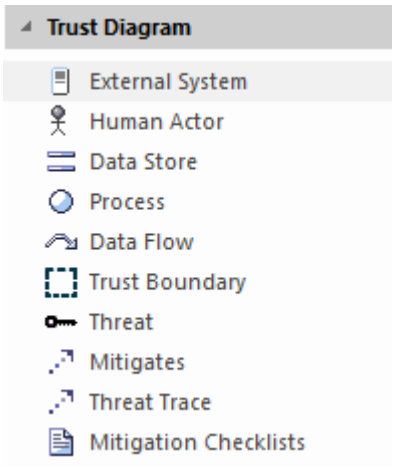
Tampering	Property: Integrity Tampering threats involve an adversary modifying data, usually as it flows across a network, resides in memory, or is stored on disk or in databases.
Repudiation	Property: Non-repudiation Repudiation threats involve an adversary denying that something happened.
Information Disclosure	Property: Confidentiality Exposing information to someone not authorized to see it.
Denial of Service	Property: Availability Deny or degrade service to users.
Elevation of Privilege	Property: Authorization Gain capabilities without proper authorization.

Trust Diagram Toolbox Page

You generally pin the Trust Diagram page of the Toolbox in with the Toolbox pages of the process modeling language

you are using - UML, BPMN or Archimate.

Note that in order to freely create the relationships in this Toolbox page, you might need to deselect the 'Strict Connector Syntax' checkbox on the 'Links' page of the 'Preferences' dialog.



Icon	Represents
External System	An external interactor.
Human Actor	A user.
Data Store	A generic data store.
Process	A generic process.
Data Flow	A directional or unidirectional flow of data between elements.
Trust Boundary	A border for the boundaries of trust in relation to one or more threats. You use a

	boundary to enclose a part of the system or process that is either at risk of attack or is secure from attack. The crucial object is the relationship that crosses the borders of both Trust Boundaries. For example, Element A might have an inherent weakness and so is enclosed in a Trust Boundary. Element B is secure and enclosed in a separate Trust Boundary. However, data flows between A and B, crossing the Trust Boundaries and therefore exposing B to a threat from A.
Threat	A specific type of threat.
Mitigates	A relationship that assigns a countermeasure, realized by a Mitigation checklist, to a Threat element.
Threat Trace	A relationship that defines a trace between a Threat element and the threatened elements of a Trust diagram.
Mitigation Checklist	A Checklist element representing predefined mitigations of a Threat.

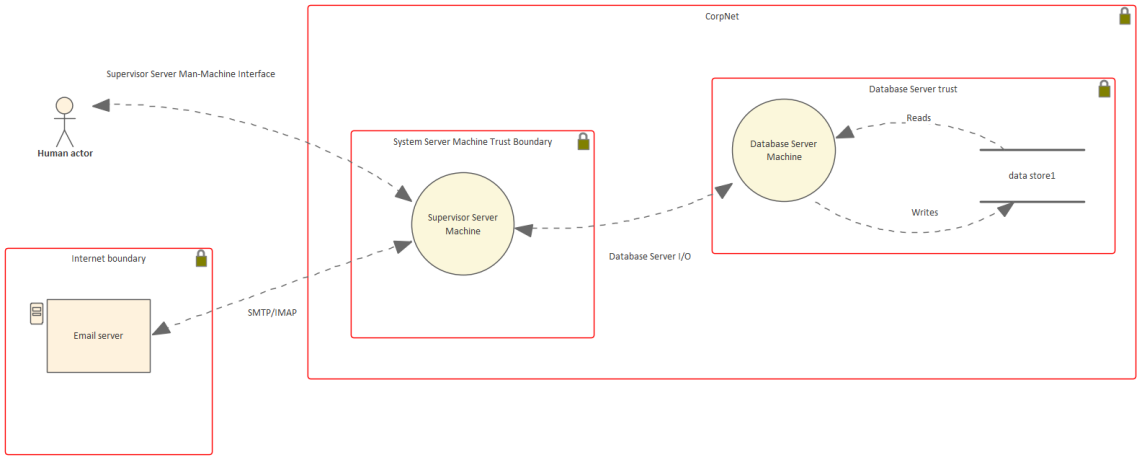
Trust Diagram

A Trust diagram can serve different purposes, depending on what you want to model and examine. Initially, the Trust diagram might capture sections of a process and show how those sections are locked in to each other or separate from each other by enclosing the elements in Trust Boundaries. You might then have a separate diagram to show what threats to the system exist, what particular elements are open to those threats, and what measures might be put in place to mitigate the threats. This could be one diagram representing all of the system under investigation, or several diagrams each representing one segment or one threat type.

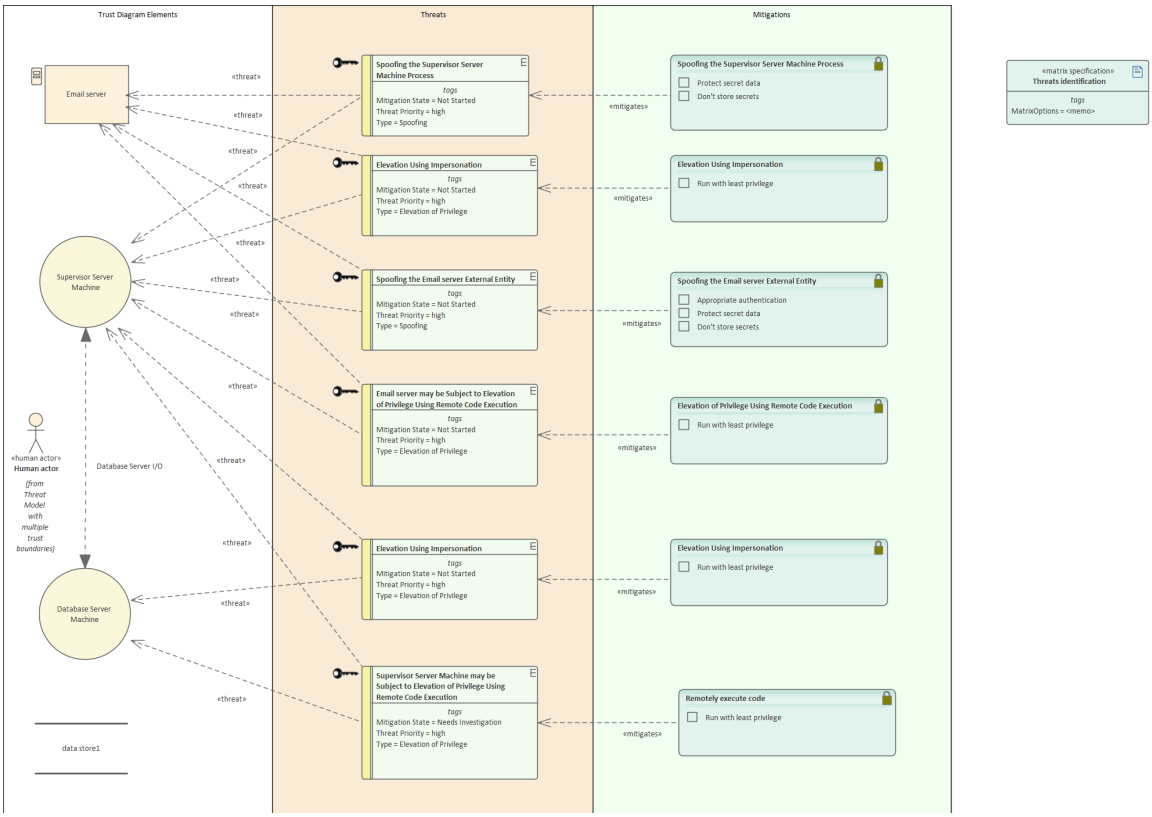
Example Diagrams

The *Threat Model with multiple trust boundaries* pattern in the 'Create from Pattern' tab (Model Wizard) creates an example of a Threat Model structure with Packages for Trust diagram elements and identified threats. In addition, it provides the concept for establishing traceability between the identified threats and the Trust diagram elements that the threat is associated with.

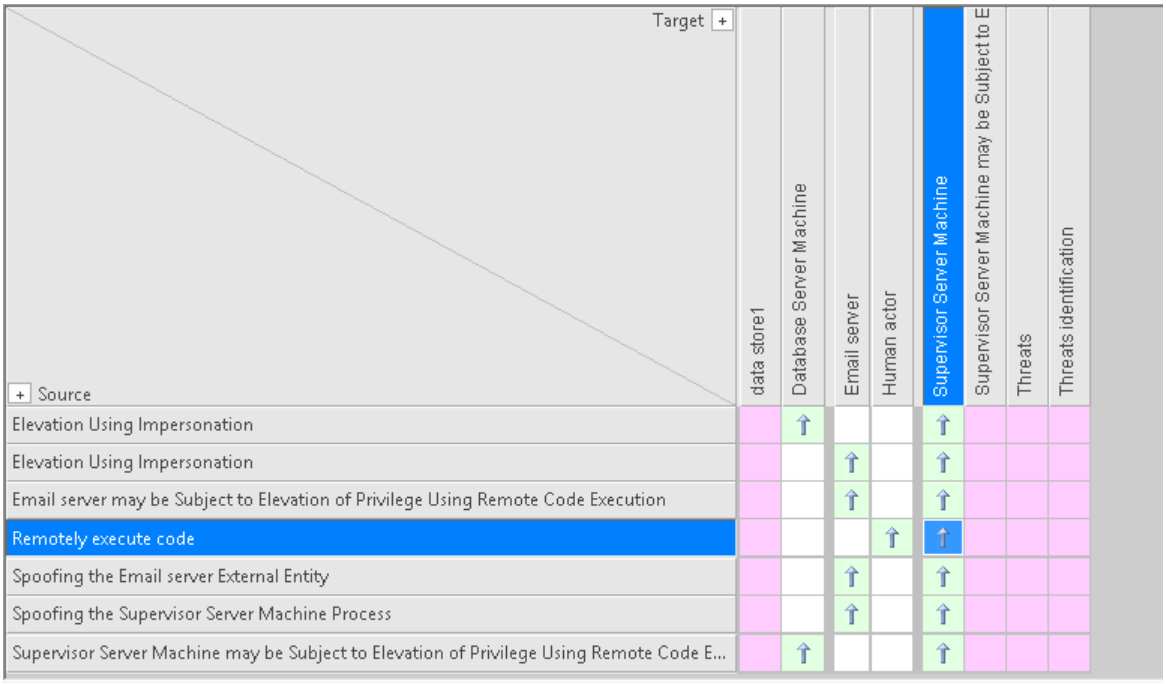
Derived from this pattern, the first figure shows several trust boundaries differentiating between specific security constraints.



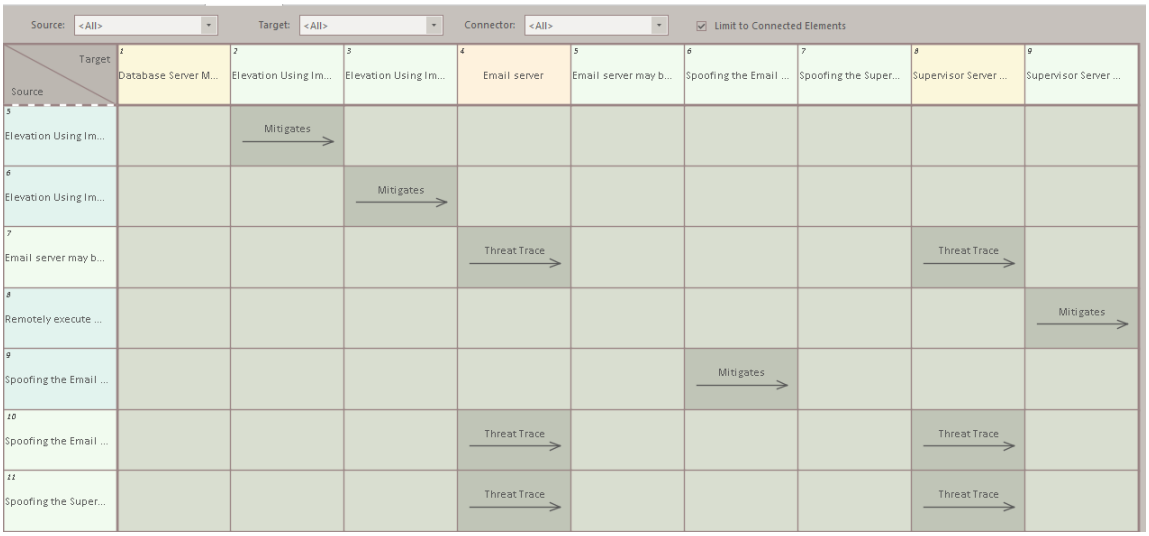
The next figure shows a list of identified threats with appropriate Tagged Values used for threat evaluation.



The third figure illustrates the existing Trace relationships between Threats and Trust diagram elements in the Relationship Matrix.



The final figure shows the existing Trace relationships between Threats and Trust diagram elements in the Relationship Matrix special diagram view.



Useful Enterprise Architect Tools in Threat Modeling

Enterprise Architect provides a wealth of tools for designing, developing, documenting and testing processes and systems. Most of these tools can be applied in creating and analyzing Threat models, but the tools identified in the next table are of particular value in this domain.

Tools in Threat Modeling

Tool	Description
Traceability Window	The Traceability window automatically displays the relationships that exist between a selected element and other model elements, including up-process and down-process elements. The traceability tree view can be conveniently expanded to show deeper relationships, and is invaluable for revealing the effects of an element on others not immediately linked to it.
Relationship Matrix	The Relationship Matrix provides a spreadsheet view of the relationships between two groups of elements. It can

	be a used as an analysis mechanism to visually indicate how elements are related and to discover which elements are missing relationships.
Discussions	The Discussions facility is a fully-featured collaboration tool, allowing modelers, model viewers and reviewers to communicate with each other about specific model objects, directly inside the repository. Modelers using the full client, or occasional viewers using WebEA, can both post and reply to discussions and communicate and engage in chat.
Diagram Layout	The Diagram Layout tool allows you to automatically lay out an entire diagram, selected elements or sections of a diagram to make it easier to read and see how relationships flow between elements. There is a wide range of layout types to choose from, and some types have additional filters that can be applied.
Pan and Zoom	The Pan and Zoom facility can be used to navigate around a large diagram; often the resolution of a diagram must be reduced to ensure it is wholly visible, but using the Pan and Zoom window allows

	you to leave the diagram at a readable resolution but to pan across to areas of interest, zooming in when necessary. Trust diagrams can contain many elements and relationships, so it is very useful to be able to see the whole picture yet have a clear view of how particular objects in it are interacting.
Alternative Images	Most standard elements allow an alternative image to be defined for an element, to be used in place of the graphical notation for the element either on a selected diagram or as a default on all diagrams. These images can help you represent the objects or processes under investigation more accurately and with immediate recognition.
Document Generator	The Document Generator is a convenient Enterprise Architect facility that allows a Database Engineer or other stakeholder to create high quality corporate or technical documentation directly from the Threat model, suitable for internal or external audiences.

Modeling Business Rules

Manage, Refine and Apply Simple and Complex Rules that Govern a Business

In any business action or process, the start, progress and end result are usually determined by reference to a set of rules. These rules can be very simple, such as 'the client must present documentary evidence of being at least 18 years old', or very complex, such as the actuarial rules that determine what a tailored insurance policy will and will not cover.

Whether the rules of your business are simple or complex, there are two sets of considerations to take into account:

- How to manage the rules - How are they initially identified? Where are they held? Are the rules easily maintained and updated? How are they refined and tested?
- How to use the rules - How easy is it to identify which rules apply in a specific context? How easily can any specific rule be recognized and applied? How are the rules executed in the process - can they be integrated with the process? Can execution of the rules be automated in the process?

Both sets of considerations can be easily managed by modeling your business processes in Enterprise Architect, and using the Business Rule Model facility. Business Rule modeling captures the rules that govern a business, and their relationships with the entities and specific tasks within the organization or system.

Managing Rules

Broadly, modeling your business processes can clarify:

- Your business requirements (from which many business rules are ultimately derived)
- The Use Cases - and the scenarios in each Use Case - to satisfy those requirements, and
- The exact processes, stages, objects, actions and data structures that support those Use Cases, represented by Classes

This process will also clarify which of your current business rules are applicable to which points in each process, and what refinements or new business rules are required. You can then map your business rules to existing or new Classes, using two specific Business Rules models; the:

- Business Domain model, in which you group the business objects (represented by Classes) involved in a process or application, and develop a Rule Flow that defines the tasks (as Rule Task elements) associated with the process as a whole or specific objects in the process
- Business Rules model, in which you create a specific Business Rule element for each business rule and associate it with the Rule Task to which the rule applies

When you have defined all the tasks, their sequences, and the rules that apply to each one, you can compose the rules per task to define the values and conditions of the rules and how they take effect in the task. You can then validate the

rules for the task to ensure that they are logical.

A valuable resource that you have created in this process is a database of business rules associated directly with the tasks and procedures they apply to, easily explored (according to the naming and/or numbering convention you have used) with the Model Search and other navigation and display facilities, and documented through the document or web reporting facilities. You can also record further information on each rule using internal or external notes, Tagged Values and Linked Documents.

Using Business Rules

Having set up the business rules database, your users can access the models or their documentation as a reference. As explained previously, the context of any given rule, or the rules applicable to a context, can be quickly established using the search, navigation or Traceability facilities.

However, you can use Enterprise Architect to model and create applications and user interfaces that can apply the business rules you have defined, and a further facility of Business Rule modeling is to generate the behavioral code for the rules in a specific task. You can merge this into your code to prompt for or even automate the correct use of the business rules in performing a task.

Advantages of modeling Business Rules

Whether you create a database of rules, or applications that apply the rules, you have a modular solution to a business process requirement. This provides an advantage in localization. Business Rules can vary between locations; for example, car hire operates in roughly the same way in most countries, but the legal driving age differs between the countries, as do the models of car available for hire. You can easily create different localized rule modules and switch the appropriate one for the current location into the common model.

Notes

- Business Rule Modeling is available in the Unified Edition and the Ultimate Edition of Enterprise Architect

Develop a Business Rule Model

In modeling Business Rules, you first develop the model structure to represent the rules in the context of their use, and then effectively compile (or compose) the rules to make them operational within that context. From the compiled rules you can either create a spreadsheet for reference or generate behavioral code for applications that apply the rules, or both.

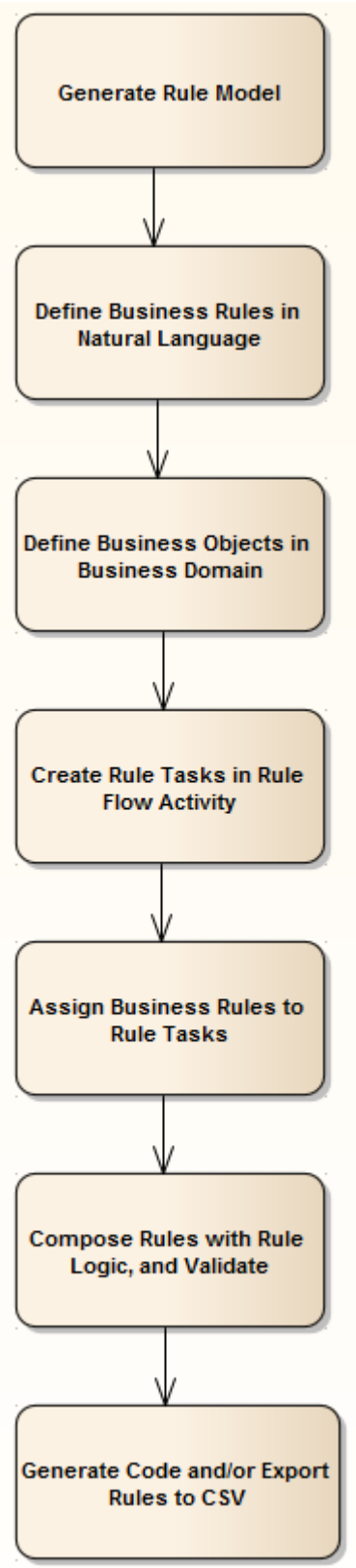
Process Summary

1. Use the 'Create from Pattern' tab (Model Wizard) to generate a Business Rule Model in which to define the business rules.
2. In the generated Business Rules diagram, begin to identify the business rules as Business Rule elements, each element representing a specific business rule.
3. In the generated Business Domain Model diagram, edit the existing Classes - and create others - to represent the Business Objects in the business domain or process; these provide the business vocabulary as the context for the business rules.
4. One of the Classes represents the actual application of the rules; under this Class is a Rule Flow Activity and Activity diagram, in which you create the Rule Tasks under which the business rules are grouped, in the

sequence in which the business rules are executed. If you require a higher level of grouping to define different areas of rule application, you can create other rule-processing Classes with their own Activities.

5. Return to the Business Rules diagram and drag in the Rule Task elements from the Browser window, assigning to each Task the corresponding Business Rule elements.
6. Compose and validate the business rules, using the Rule Composer.
7. If you want code that applies and executes the business rules, generate it from the Class elements that contain the Business Rule Activities.

The steps are represented graphically in this flow:


Generate Rule Model
Define Business Rules in Natural Language
Define Business Objects in Business Domain
Create Rule Tasks in Rule Flow Activity
Assign Business Rules to Rule Tasks
Compose Rules with Rule Logic, and Validate
Generate Code and/or Export Rules to CSV

Notes

- Business Rule Modeling is available in the Unified Edition and the Ultimate Edition of Enterprise Architect

Generate a Business Rule Model

The Business Rule Model captures:

- The rules that apply to a business process
- The business tasks or objects that the rules take effect on, and
- The actual processing that takes place to apply the rules and produce a decision or result

As a very useful starting point in setting up your Business Rule model, you can generate the model structure and initial components using the Enterprise Architect Model Wizard (Start Page 'Create from Pattern' tab).

Access

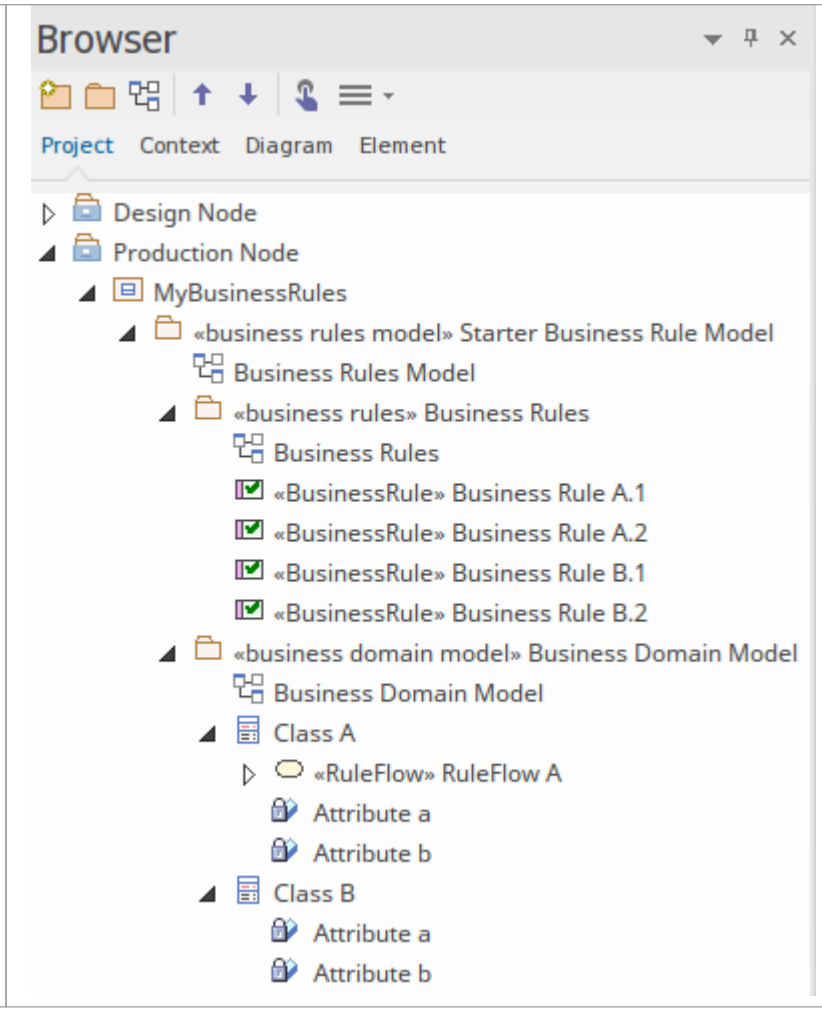
Use any of the methods outlined here to open the Model Wizard (Start Page 'Create from Pattern' tab).

Ribbon	Design > Package > Model Wizard
Context Menu	Right-click on Package or Model Root node | Add a Model using Wizard
Keyboard Shortcuts	Ctrl+Shift+M

Other	Browser window header bar | 

Generate Business Rule Model from Model Wizard

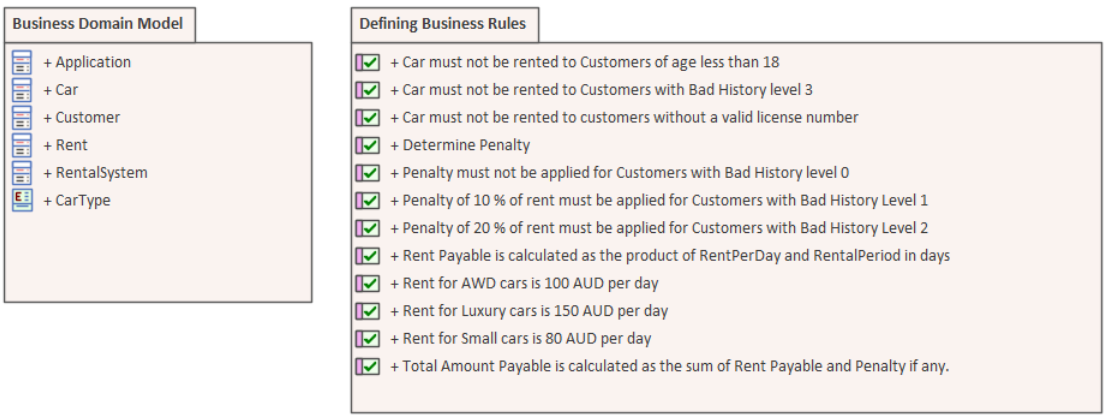
Step	Action
1	Select the 'Model Patterns' tab.
2	Click on the Perspective drop-down arrow and select 'Requirements | Business Rule Model'.
3	Click on the 'Starter Business Rule Model' option. If you prefer, you can read the information on this model in the right-hand panel.
4	Click on the Create Model(s) button. A Business Rule Model structure is generated in the Browser window, as shown.



Example Diagram

This Business Rule Model diagram was generated at the top level of the model and encapsulates the components of the Business Rule model.

Business Rule Model



Notes

- Business Rule Modeling is available in the Unified Edition and the Ultimate Edition of Enterprise Architect
- To see an example of a Business Rules model, open the EAExample model provided with the installer, and expand:

  Example Model > Analysis and Business Modeling > Business Domain Model > Business Rule Model > Business Domain Model and > Defining Business Rules

Model Business Rules

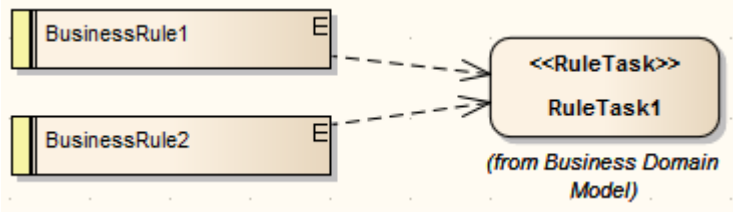
In the Business Rule Model, you initially define each business rule as a Business Rule element and later group these rules by linking them (using Dependency connectors) with Rule Task elements. In the:

- First stage you assemble a collection of rules
- Second stage you organize the rules into groups and sequences through the Rule Task elements created in the Business Domain model, and refine them by adding further Business Rule elements (and, if appropriate, further Rule Task elements)

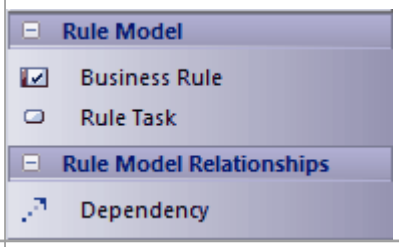
In the Car Rental example in the EAExample model, rules have been defined and grouped to perform an eligibility check on a customer, to determine if the customer is eligible to rent a car.

Define Business Rule elements

Step	Action
1	In the generated Business Rule Model, expand the Business Rule Package and double-click on the Business Rule diagram to open it. The diagram shows two example Business Rule elements connected to an example Rule Task.



In the Diagram Toolbox, the 'Rule Model' pages display.



2 If you should need to create another (empty) Business Rules diagram, right-click on the Business Rules Package and select 'Add Diagram'.

On the 'New Diagram' dialog, type an appropriate name in the 'Diagram' field, click on the 'Select From' header and select the 'Project Management > Complete' Perspective group, and then scroll down and select the Business Rule Model Perspective from the panel beneath the header.

Select 'Rule Model' in the 'Diagram Types' panel and click on the OK button.

3 For each business rule you want to identify, drag the 'Business Rule' icon from the Toolbox.

Type the rule (or a shortened version of it) in the 'Short Description' field of the element 'Properties' dialog. This displays as the name of the element in

	the diagram. You will later define the parameters of the rules using the Rule Composer.
4	When you have created all the required Business Rule elements, create the Business Domain model, the Rule Flow Activity diagram and the Business Task elements.

Associate Business Rules with Business Tasks

After you have created the Business Domain Model and Rule Flow Activity (which contains the Rule Task elements in the context of the business process), you can link the Business Rule elements to the Rule Task for the action in which the rules take effect. You can apply a Business Rule to more than one Rule Task, if it has an effect in different contexts.

There are several ways to establish the relationship between Business Rule and Rule Task.

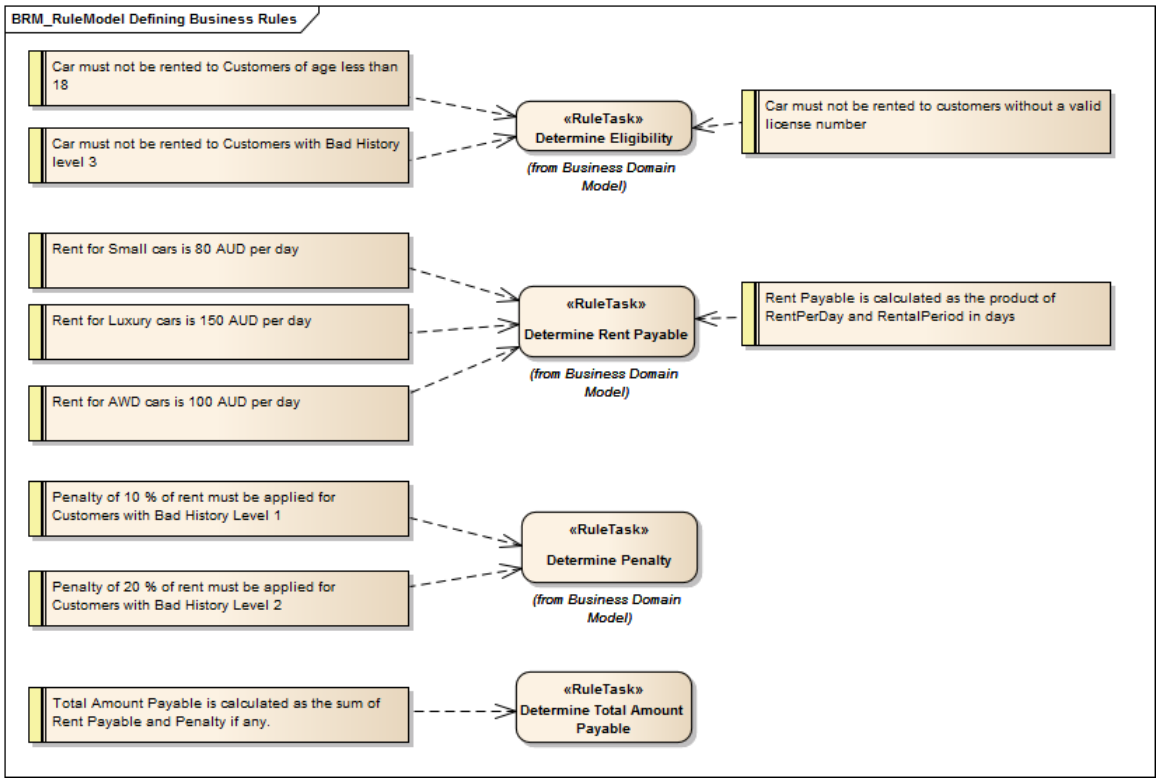
Method	Action
Use the Business Rules diagram	Open the Business Rules diagram and drag onto it a Rule Task element from the RuleFlow Activity in the Browser window.

	Create a Dependency relationship to the Rule Task from each Business Rule element that applies to it (by dragging the connector from the Toolbox, or by using the Quick Linker arrow on the Business Rule element). Repeat the process for the next Rule Task element from the Rule Flow Activity. Create any additional Business Rule and Rule Task elements that are necessary (this should not be a common event); you must add any new Rule Tasks to the Rule Flow Activity diagram. Save the diagram and open the Rule Composer for the first Rule Task element.
Assemble existing elements through the Rule Composer	In the Browser window, open the Rule Composer on a Rule Task element, and drag each applicable Business Rule element from the Browser window into an empty row of the Rule Statements table. This establishes a Dependency relationship between the Rule Task element and each Business Rule element. Continue to compose the rule conditions.
Use the	You can quickly create all the

Relationship Matrix	Dependency relationships between a number of Rule Task elements in the Business Domain Model Package and the Business Rule elements that apply to each one. Set the source element 'Type' to 'BusinessRule' and the target element 'Type' to 'RuleTask'. Set 'Link Type' to 'Dependency', and 'Direction' to 'Source->Target'. Set the 'Source Package' to 'Business Rules' and the 'Target' to 'Business Domain Model'. In the cell at each intersection of a Business Task column and the appropriate Business Rule row, right-click and select the 'Create new relationship | UML:: Dependency' option. When you have finished creating the relationships, close the Relationship Matrix. Open the Rule Composer for the first Rule Task element in the Browser window.

Example

However you connect the Rule Task and Business Rule elements, if they are added to a diagram it might display as shown.

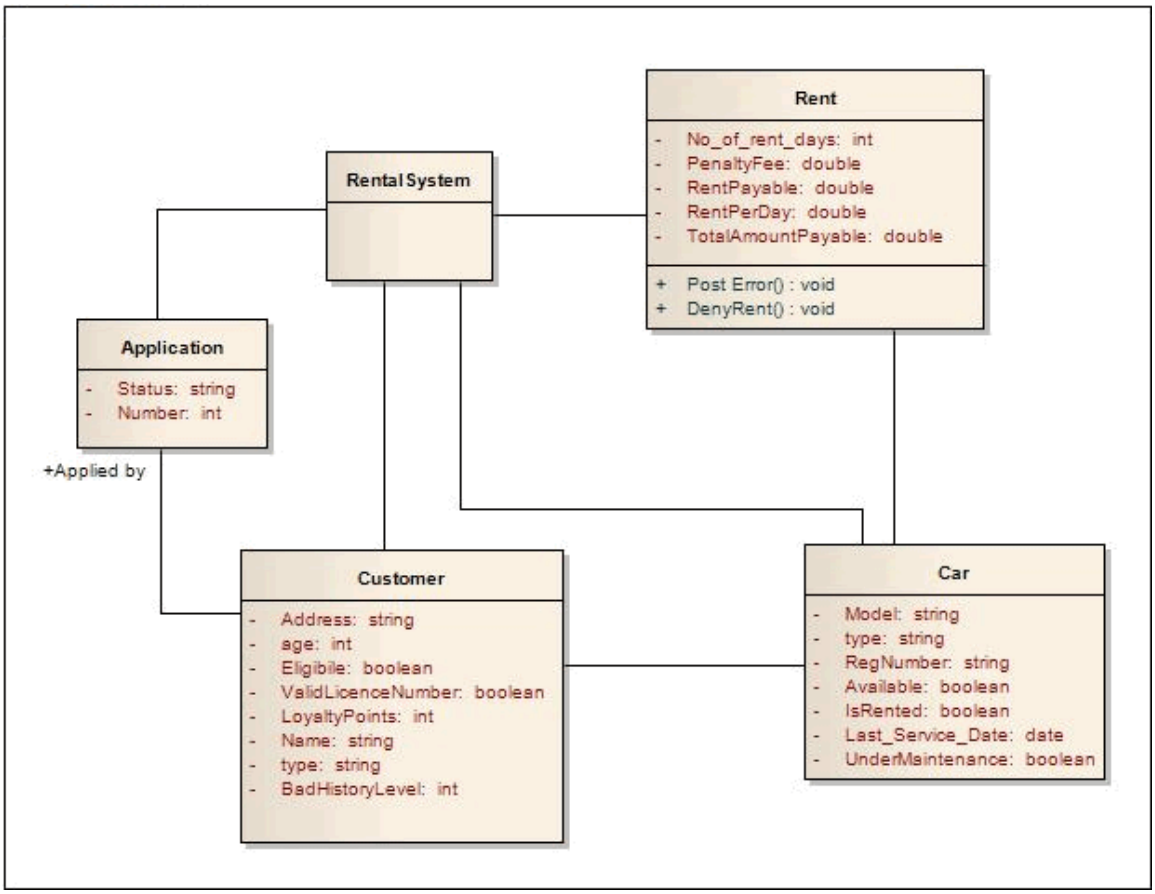


Notes

- Business Rule Modeling is available in the Unified Edition and the Ultimate Edition of Enterprise Architect

Create a Business Domain Model

The Business Domain Model provides the business vocabulary - the terms and facts - on which Business Rules can be modeled. In Enterprise Architect a Business Domain model is represented as a conceptual Class diagram, as illustrated by this diagram from the Car Rental System model from the EAExample model.



In the Business Domain model shown in this diagram, the Classes Rent, Customer, Car and Application, together with their attributes and operations, provide the terms for the business vocabulary for the car rental system. The operations and attributes identify the conditions that must be met, the actions that must be taken, and the calculations that

must be made to filter and apply the rules to provide a specific value or outcome.

The Class Rental System processes the rules; to make this possible, you add a Rule Flow Activity as a behavior for this Class.

When you create a Rule Flow Activity under a Class, you model the events and sequence as a structure of Rule Tasks (Actions). When you generate code for the Class (in the example, Rental System) the rule flow behavior is rendered as a method inside the Class.

Alternatively, if you have existing operations in the Class that already suit the purpose, you can model business tasks in those operations. When code is generated for the Class, the rules logic is generated as the method body for the corresponding operation.

Notes

- Business Rule Modeling is available in the Unified Edition and the Ultimate Edition of Enterprise Architect
- When you create Classes in the Business Domain model, select the correct language for code generation to ensure that the correct data type is set for attributes and operation parameters
- Business Rules code generation is supported for these languages:
    - C++
    - C#

- Java
- VB.Net

Create a Rule Flow Activity

When you set up a Business Domain model within a Business Rules model, you create a Rule Flow Activity as a behavior for one of the domain Classes, to enable that Class to process a set of rules. In the Rule Flow Activity you create a number of Rule Task elements, which are stereotyped Actions that group Business Rules for a specific task. The Rule Flow Activity automatically generates a Rule Flow diagram, in which you create the Rule Task elements and model the sequence in which they are executed.

Add a Rule Flow Activity to a Class

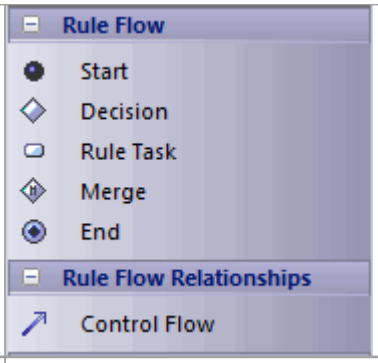
Step	Action
1	On the Business Domain model diagram, right-click on the Class that processes the rules (in the Car Rental example in the EAExample model, this would be Rental System).
2	From the context menu select the 'New Diagram | RuleFlow Activity' option. A new Rule Flow Activity is created with a Rule Flow diagram, which immediately opens. Go on to create the Rule Task elements.

	Code generation for a Rule Flow model renders each RuleFlow Activity as a set of operations or methods. Depending on what these methods are to do, you might want to pass in parameters to be used within the Rule Flow Activity.

Add Rule Task elements to the Rule Flow Activity diagram

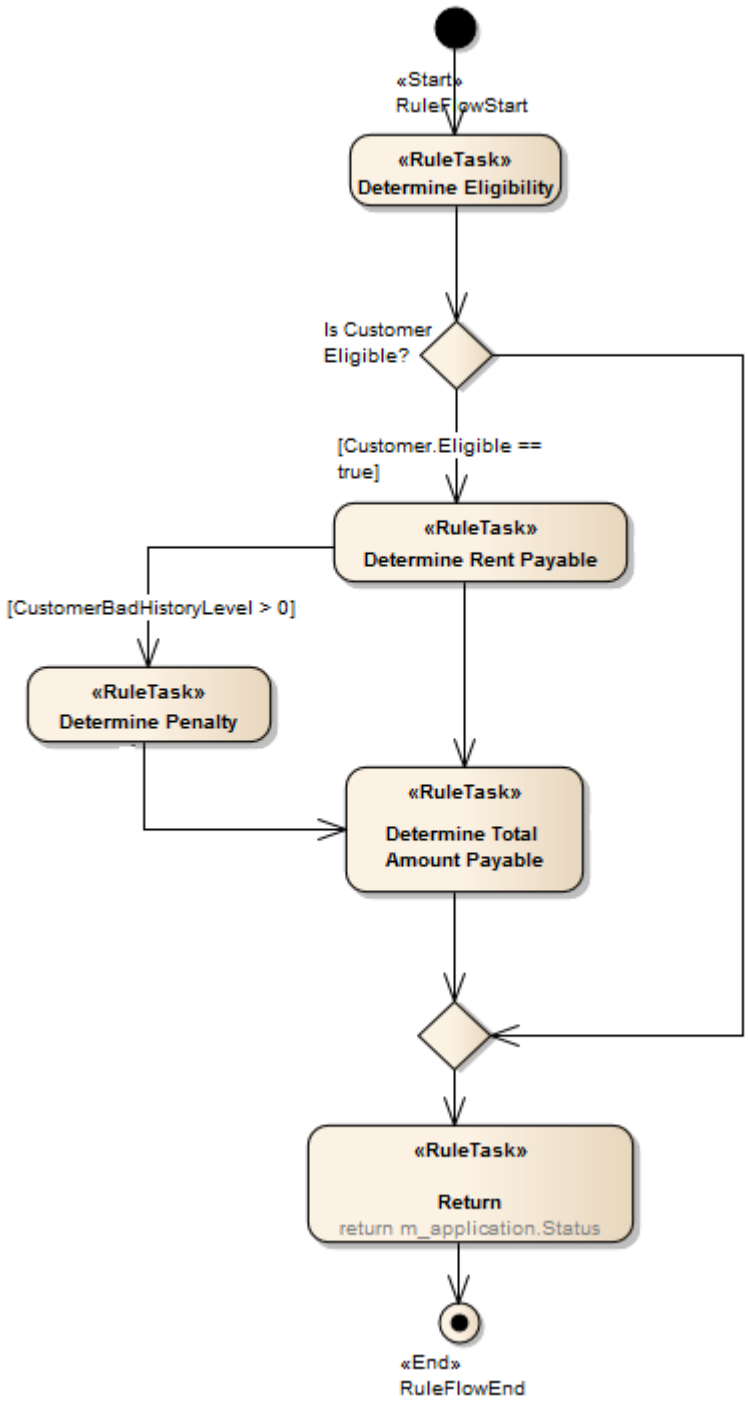
You can create Rule Task elements directly under the Rule Flow Activity in the Browser window, by clicking on the 'New Element' icon in the Toolbar and selecting the UML::Activity toolset, Action element type and RuleTask stereotype. However, it is much simpler to create the elements on the Rule Flow diagram, and at the same time to organize them into their processing sequence.

Step	Detail
1	(If necessary) Click on  to display the 'Find Toolbox Item' dialog and specify 'Rule Flow'.

	
2	Drag the 'Rule Task' icon from the Toolbox onto the diagram and give the element, as a name, the title of the task it represents, such as Calculate Debit Charge or Determine Eligibility. Create a Rule Task element for each task or action in the process. You can also use the Quick Linker arrow to create the new elements and Control Flow connectors.
3	Organize the Rule Tasks into a sequence of events, initiated and terminated by the Start and End elements, and representing any branching and rejoining with Decision and Merge elements. All elements are connected by Control Flow connectors. See the example diagram.
4	Go to the Business Rules model diagram and group the Business Rule elements on their appropriate Rule Task element.

Example

This Rule Flow diagram is from the EAExample model Car Rental example.



The Rule Task elements Determine Eligibility, Determine

Rent Payable, Determine Penalty and Determine Total Amount Payable group the business rules for the specific task indicated by the element name.

Notes

- Business Rule Modeling is available in the Unified Edition and the Ultimate Edition of Enterprise Architect
- In a Rule Flow diagram, every Decision node has a matching Merge node to ensure proper code generation
- For code generation, the Rule Task elements must be grouped inside the appropriate Rule Flow Activity in the Browser window (Business Rule elements can be defined anywhere in the model, as they can be used in more than one Rule Task)

Pass Parameters to Rule Flow Activity

When you generate code for a Rule Flow model, each RuleFlow Activity is rendered as a set of operations or methods. You can pass in Activity Parameters to be used by the Rule Tasks within the Rule Flow Activity, to define what you want the methods to do. You can use the parameters as condition variables or action variables in the Business Rule Decision Table, or as rule variables in the Computation table for any of the Rule Tasks.

If the Activity Parameter is not accessible to a Rule Task, the system displays an error message.

Access

Context Menu	Browser window | Double-click Rule Flow Activity element > Parameters

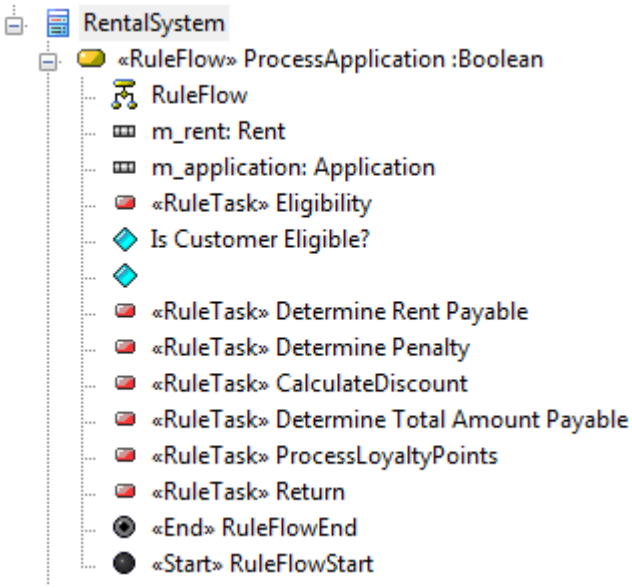
Define parameters to be used within a Rule Flow Activity

Step	Action

1	On the 'Parameters' page, create and define each parameter, in particular the 'Type' and 'Default' values.
2	Save each parameter and, when you have finished setting all parameters, close the 'Properties' dialog.

Example

In this hierarchy, the parameters m_rent and m_application can be used by any of the Rule Tasks under the ProcessApplication Rule Flow Activity.



Model Rules In an Operation

You can model business rules as Business Rule elements in the Business Rules Package, and attach them to the Rule Task elements in a RuleFlow Activity diagram.

Alternatively, in the Business Domain model, if you have operations in the rules processing Class that represent business action, you can define each of those operations as a Rule Task and attach the Business Rules to these operations on the Business Rules diagram or within the Rule Composer.

Access

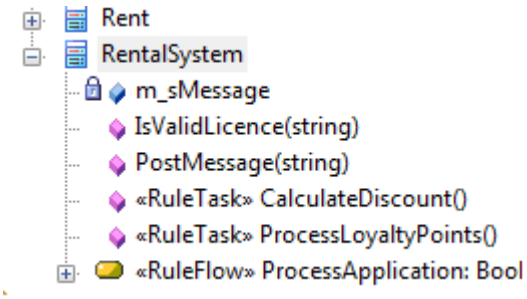
Context Menu	Browser window| Double-click on Operation > General

Model Business Task on an operation

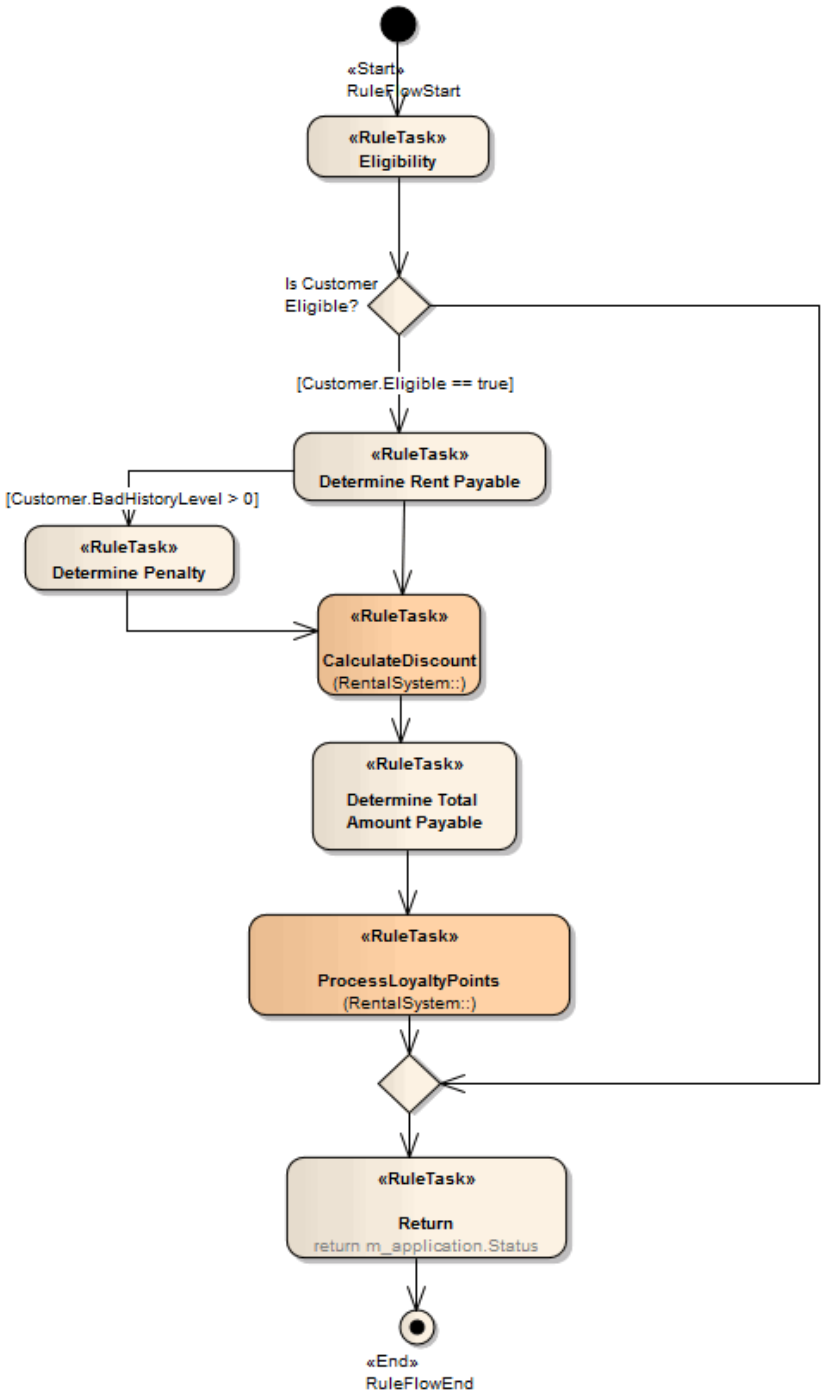
Step	Action
1	In the 'Stereotype' field type 'RuleTask'.

2	Click on the Save button and on the Close button.
3	Drag the Operation from the Browser window onto the RuleFlow Activity diagram. Also assign Business Rules to the operation, as for a Rule Task element.

Representation of Operation Rule Tasks

View	Detail
Browser Window	The operations stereotyped as RuleTask display in the Browser window as shown:  To pass the parameters for these operation calls, open the operation 'Properties' dialog and select the 'Call' page, then set the 'Behavior field to the operation to be called; under the 'Arguments' field, click on the Edit button and set up or edit the argument values to

	be passed. On code generation, the code for rules logic is generated in the method body.
On the Rule Flow Diagram	When you drag and drop a RuleTask-stereotyped operation onto a Rule Flow diagram, an operation Call Behavior Action is created, as shown by the two darker elements.



You must build the Call Behavior Actions into the activity flow, as for the normal Rule Task Actions.

Notes

- Business Rule Modeling is available in the Unified Edition and the Ultimate Edition of Enterprise Architect

Compose Business Rules

In modeling your Business Rules, you first define each business rule as a conceptual-level, plain text string within a Business Rule element, and then group the rules by association with Business Task elements. Your next step is to define exactly how the rules operate within the task, setting up the values, conditions, actions and computations that define the actions of a single rule or a combination of rules. For this you use the Rule Composer, with which you transform each conceptual-level business rule statement into a logical level, technology-specific tabulated statement that you can either:

- Generate code from or
- Download to a spreadsheet application such as Microsoft Excel, via a CSV file

Access

Context Menu	Browser window or open diagram | Right-click on a Rule Task element | Rule Composer

Rule Composer Tables

The Rule Composer displays as a view in the central work area, divided into three tables.

Table	Detail
Rule Statements	The Rule Statements table lists the rules associated with the selected Rule Task; you add a rule to the table by dragging an existing Business Rule element from the Browser window onto an empty row of the table. You do not create new rules within the table.
Decision	The Decision Table is used to model conditional rules (for example: Cars must not be rented to customers of age less than 18). The table has three sections: • Rule Conditions – to model condition variables • Rule Actions – to model action variables • Rule Bindings – to link the rule in the rule table
Computation Rule	Using the Computation Rule table, you model rules that require a calculation to

	be performed on the source information, and/or the interaction of rules. The table has these columns: • Computation Rule Actions • Expression • Rule Bindings • Rule Dependency

Notes

- Business Rule Modeling is available in the Unified Edition and the Ultimate Edition of Enterprise Architect
- To assist with traceability as you complete the relationships across the Rule Composer, selecting an entry in one table automatically highlights the corresponding rows and columns of the other tables; for example, if a Rule Statement is selected, the related rule column in the Decision Table and row in the Computation Rule table are highlighted
  Similarly, if a Computational Rule is selected, the corresponding column in the Decision Table and row in the Rule Statements list are highlighted
- If the table columns are not wide enough to display their contents, you can drag the column header margins to increase the width; all tables on the 'Decision Table' tab and the 'Computation Rule' tab are linked, so if you

increase the column width on one table it changes the width on all tables

- The Rule Composer can be opened and the rules logic viewed in the Lite Edition of Enterprise Architect

Add and Remove Rules

When you open the Rule Composer for a Business Task, all Business Rules currently linked to that task are already listed in the Rule Statements table. You can add further rules that are available in the Browser window, or remove a rule from the table that is no longer applicable to the task.

You do not create or delete the rules themselves in the Rule Composer.

Access

Context Menu	Browser window or open diagram | Right-click on a Rule Task element | Rule Composer

Add a business rule to be associated with the selected Rule Task

When you add a Business Rule to the Rule Statements table, if the Business Rule element is not already on the Business Rules diagram, the element is added to the diagram. A Dependency relationship is created between the 'new'

Business Rule and the selected Rule Task.

When you open the Rules Composer	Action
There are no rules listed	The Rule Statements table has a single empty row. Select the Business Rule in the Browser window and drag it onto this empty row. Another empty row is automatically created underneath the first row. Drag the next Business Rule from the Browser window onto the Rule Statements table.
There are rules listed	The 'Rule Statements' table has no empty rows. Click on the 'No' column and select the 'Add Row' context menu option; an empty row is added to the table. Drag a Business Rule from the Browser window onto the table.

Remove a rule that is no longer required in the Rule Composer

Right-click on the appropriate 'No' field and select the 'Remove Rule' option. The rule is removed from the Rule Composer and the Dependency relationship with the Rule Task element is deleted.

The Business Rule element is not deleted from either the Business Rule diagram or the Browser window (where, in either case, it might be in use with other Rule Task elements).

Define Rule Conditions

When you create the Business Domain model, you set up a number of Classes that define the business terms and entities (such as Customer) and their associated attributes and operations. You create the attributes and operations as you set up the Class, with at least some values or parameters, and you tailor some of the features to the rules to define the conditions through which a rule takes effect, in the Rule Conditions table.

For example, in a Class that defines the properties of Car, the attribute Type might be used to set the condition 'Car is: Small, Medium or Large', through which the rules defining base rental charge would be filtered and applied.

Access

Context Menu	Open diagram | Right-click on a Rule Task element | Rule Composer > Decision Table

Model Rule Conditions

Step	Action
1	From the appropriate Class element in the Browser window, drag and drop the condition attribute (such as age) or operation (such as IsValidLicense()) onto the first empty field in the 'Rule Conditions' column. • The 'Rule Conditions' fields apply Intelli-sense to display possible entries for the field; press Ctrl+Space in the field to display a list of possible Classes, double-click on the selected Class, type . (a period) and double-click on the attribute or operation in the automatically-displayed list • If the 'Rule Condition' is of type enum, the 'Allowable Values' field is automatically set with the enum literals; the procedure then ends here
2	Determine a range of accepted values for the Rule Condition.
3	Right-click on the 'Allowable Values' column and select: • For an attribute, the 'Edit Allowable Values' option; the 'Edit Allowable Values' dialog displays • For an operation, the 'Edit Parameters' option; the 'Edit Parameters' dialog displays (see step 5) Type each required value or range of values in the 'Value' field, and click on the Save button to display

	the value in the 'Allowable Values' list box *age* could have the values: • <18 • >18 and <50 • >50 *IsValidLicense()* could return: • True • False
4	Click on the OK button to save the values and close the dialog; for an attribute, a new constraint AllowableValues is created. • You can check this constraint by opening the 'Properties' dialog for the attribute and selecting the 'Constraints' tab • If the Rule Condition references an enumeration, the enum literals are not editable in the 'Edit Allowable Values' dialog
5	If the Rule Condition is an operation, you can pass parameters to it. Right-click on the 'Allowable Values' field, and select the 'Edit Parameters' option; the 'Edit Parameters' dialog displays, listing the parameters that already exist for the operation. Select the parameters and type their values into the 'Value' text box; click on the OK button to cancel the

	dialog. • You can add an operation as a Rule Condition more than once, so you can call the operation with different sets of parameters
6	To add another Rule Condition, right-click on the 'No' column and select the 'Add Row' option; an empty row is added to the table. To remove a Rule Condition from the table, right-click on the appropriate 'No' field and select the 'Delete Row' option. • If the condition is based on an attribute, this does not affect either the original attribute or the new constraint in the model; you can either re-use the attribute with its constraint, or use the attribute 'Properties' dialog to remove the constraint
7	If any of the condition values invoke an action or a decision, you can define it as a Rule Action.

Define Rule Actions

In the Classes in the Business Domain model, the attributes and operations that you set up to define the conditions through which a rule takes effect might in their turn call a decision attribute or an operation, which you can assign as an action in the Rule Actions (Outcome) table. You create the attributes and operations as you set up the Class, with at least some values or parameters, and you tailor some of these features to define the actions that determine the effect of the rule.

Access

Context Menu	Open diagram | Right-click Rule Task element | Rule Composer > Decision Table

Model Rule Actions

Step	Action

1	From the appropriate business term Class element in the Browser window, drag and drop the decision attribute (such as Eligible - Yes/No) or operation (such as post error) onto the first empty field in the 'Rule Actions (Outcome)' column. • The 'Rule Actions' fields apply Intelli-sense to display possible entries for the field; press Ctrl+Space in the field to display a list of possible Classes, double-click on the selected Class, type . (a period) and double-click on the attribute or operation in the automatically-displayed list
2	For an attribute, if the dropped action variable is of type 'enum', the 'Allowable Values / Parameters' fields are automatically set with the enum literals. Otherwise, double-click on the 'Allowable Values / Parameters' field (or right-click and select the 'Edit Allowable Values' option); the 'Edit Allowable Values' dialog displays. Click on the New button, type a possible value in the text box (such as 'Yes' or 'Accept'), click on the Save button, and repeat the process for the next possible value (such as 'No' or 'Reject').
3	When you have entered all possible values, close the dialog and select the appropriate response in the 'Result<n>' column field underneath the appropriate 'Value<n>' field for the calling condition.

4	For an operation, a checkbox displays in each of the 'Result<n>' column fields; to call the operation as the response to a value of the corresponding condition, select the checkbox in the 'Result<n>' field underneath that condition 'Value <n>' field.
5	To pass parameters to the operation, double-click on the 'Allowable Values/Parameters' field (or right-click and select the 'Edit Parameters' menu option); the 'Edit Parameters' dialog displays. Select the parameters and type the values into the 'Value' text box; click on the Save button and close the dialog. You can add an operation as a Rule Action more than once, so you can call the operation with different sets of parameters.
6	Click on the Save button in the Rule Composer toolbar to save the values.
7	To add another Rule Action, right-click on the 'No' column and select the 'Add Row' option; an empty row is added to the table. To remove a Rule Action from the table, right-click on the appropriate 'No' field and select the 'Delete Row' option; this does not affect the original attribute or operation in the model.

Bind Rules to Conditions and Actions

After defining which rules are applied, which conditions are evaluated and/or which actions are taken in the completion of a Rule Task, you bind the rules to the conditions and actions to define:

- Which rules are evaluated against a condition
- What values of that condition are applicable to each rule
- What decisions or actions are taken on each value output from evaluation of the rule, and what the result of that decision or action is

You bind the rules, conditions and actions using the 'Rule Bindings' table, just above the 'Rule Conditions' table.

Access

Context Menu	Open diagram | Right-click on a Rule Task element | Rule Composer > Decision Table

Bind rules

Ste	Action

p	
1	Click on the 'Rule Bindings' field immediately above the appropriate 'Value<n>' and/or 'Result<n>' field. Click on the number of the rule that is evaluated against the condition for the specific value and to generate the action to provide the specific result identified in those two fields.
2	Ensure that the values set in the 'Value<n>' and/or 'Result<n>' field underneath the rule number do all operate under the rule.
3	Click on the Save icon in the Rule Composer toolbar. Continue to bind business rules to conditions and actions having logical values for an instance of the rule.

Example

- Rule 2 is 'A car must not be rented to a customer of age less than 18'
- The first Rule Condition is Customer.age
- The 'Value1' field contains the value '< 18' against that Rule Condition

- The first Rule Action (or decision) is Customer.Eligible
- The Result1 field contains the value 'No' against that Rule Action
- The second Rule Action is Application.Status
- The 'Result1' field contains the value 'Reject' against that Rule Action

You would select '2' in the 'Rule Bindings' field over the 'Value1/Result1' column, so that a when a customer's application to rent a car is tested against rule 2, if the customer is found to be under 18 they are automatically classified as ineligible and the application rejected.

Define Computation Rules

In a business process, some actions and effects are dependent on the interaction of rules with either another rule or a calculation. For example:

- Rule 1) Cars cannot be rented to customers aged under 18 years, Rule 2) Customers must have a valid driving licence - Rule 1 must be satisfied before Rule 2 is tested
- Rule 4) Penalty of 10% of total rental cost applies to customers with Bad History Level 1 - If the customer has a Bad history Level 1, multiply the rent payable value by 1.1

Access

Context Menu	Open diagram | Right-click on a Rule Task element | Rule Composer > Computation Rule Table

Define a computation rule

Step	Action

1	In the Browser window, expand the appropriate business entity Class in the Business Domain model, and drag the attribute that represents the computed action into the 'Computation Rule Actions' field. Alternatively, you can use Intelli-sense in both the 'Computation Rule Actions' field and the 'Expression' field to display a list of possible Classes and attributes: 1. Press Ctrl+Space in the field to display the list of Classes. 2. Double-click on the required Class, and type . (period) after it; a list of attributes for that Class displays automatically. 3. Double-click on the required attribute.
2	In the 'Expression' field, complete the expression to be evaluated. For example: If computing Rule 4, and you have selected *Rent.RentPayable* in the 'Expression' field, then type **1.1* immediately after it.
3	In the 'Rule Bindings' field, click on the drop-down arrow and select the number of the rule being modeled (as listed in the Rule table) to link the computation data to the rule.
4	If the rule depends on another rule being satisfied first (as for Rule 2), type the number of that rule in

	the 'Rule Dependency' field.
5	Click on the Save icon in the Rule Composer toolbar to save the computation rule. If the computation rule is also a Rule Conditions rule, add the condition variable in the Decision Table and bind the appropriate rule in the 'Rule Bind' section.

Validate Business Rules

Before you generate code for the Rule Task elements, or export them to a CSV file, it is recommended that you validate the business rules in the Rule Composer. When you do this, the business rules on the Rule Composer are parsed and any errors or warnings that might indicate incomplete or unfavorable code generation are displayed on the 'Rule Composer Validation' tab on the System Output window.

Access

Context Menu	Open diagram | Right-click on a Rule Task element | Rule Composer

Validate Rules

Step	Detail
1	Click on the 'Validation' icon (green tick) in the Rule Composer toolbar: 

	The System Output window displays, showing the status of the validation - Validation complete, plus any validation errors or warnings - on the 'RuleComposer Validation' tab.
2	Double-click on each warning or error message in turn to highlight and investigate the faulty data that caused that message, in the Rule Composer.

Export Composed Rules to CSV

It is possible to export composed Business Rules to an external spreadsheet - such as Microsoft Excel - as a .csv file. You can then maintain the spreadsheet in that tool for use as a reference for the organization, if you have not created an application to automatically apply those rules.

Access

Context Menu	Open diagram | Right-click on a Rule Task element | Rule Composer

Export the contents of the Rule Composer to a CSV file

Step	Action
1	Click on the 'Export to CSV' icon () in the Rule Composer toolbar. The 'Windows Browser' dialog displays.

2	Browse to the required file location and type in a name for the .csv file to export the data into.
3	Click on the Save button to export the data.

Code Generation For Business Rules

After you have modeled rules for all Rule Task elements in the Rule Flow diagram, the Business Domain model is ready for code transformation. The internal code templates provided for generating technology-specific rule code work hand-in-hand with the EASL code templates to generate the code from the rules-processing Class and its Rule Flow structure.

Return a value from the Rule Flow behavior

Step	Action
1	Double-click on the last Rule Task element before the end node of the Rule Flow diagram. The element's 'Properties' dialog displays.
2	Click on the 'Effect' tab.
3	In the 'Effect' field, type the return statement; for example, return True.
4	Click on the Save button, and on the OK button to close the dialog. Generate code for the Class containing

	the rule flow behavior (in our initial example, Rental System); the code for business rules logic is generated, with the rule statements expressed in natural language as comments.

Example

This code snippet was generated from the Rental System Class element in the EAExample model:

```
///////////////////////////////////////////////////////////
// RentalSystem.cs
// Implementation of the Class RentalSystem
// Generated by Enterprise Architect
// Created on: 26-July-2016 2:39:23 PM
///////////////////////////////////////////////////////////
    using System;
    using System.Collections.Generic;
    using System.Text;
    using System.IO;

    public class RentalSystem
    {
        public Customer m_Customer;
```

```
        public Car m_Car;
        public Rent m_Rent;

        public RentalSystem()
        {
        }
        ~RentalSystem()
        {
        }
        public virtual void Dispose()
        {
        }
        /* Begin - EA generated code for Activities and
Interactions */
        public bool ProcessApplication(Rent
m_rent,Application m_application)
        {
            // behavior is an Activity
            /*CAR MUST NOT BE RENTED TO
CUSTOMERS WITHOUT A VALID LICENSE
NUMBER*/
            if(m_Customer.ValidLicenseNumber ==
"FALSE")
            {
                m_application.Status = "Reject";
```

```
            m_Customer.Eligible = false;
        }
        /*CAR MUST NOT BE RENTED TO
CUSTOMERS OF AGE LESS THAN 18*/
        if(m_Customer.age < 18)
        {
            m_application.Status = "Reject";
            m_Customer.Eligible = false;
        }
        /*CAR MUST NOT BE RENTED TO
CUSTOMERS WITH BAD HISTORY LEVEL 3*/
        if(m_Customer.BadHistoryLevel == 3)
        {
            m_application.Status = "Reject";
            m_Customer.Eligible = false;
        }
        if (Customer.Eligible == true)
        {
            /*RENT FOR SMALL CARS IS 80 AUD PER
DAY*/
            if(m_Car.type == Small)
            {
                m_rent.RentPerDay = 80;
            }
            /*RENT FOR AWD CARS IS 100 AUD PER
```

```
DAY*/
            if(m_Car.type == AWD)
            {
                m_rent.RentPerDay = 100;
            }
            /*RENT FOR LUXURY CARS IS 150 AUD
PER DAY*/
            if(m_Car.type == Luxury)
            {
                m_rent.RentPerDay = 150;
            }
            /*RENT PAYABLE IS CALCULATED AS
THE PRODUCT OF RENTPERDAY AND
RENTALPERIOD IN DAYS*/
            m_rent.RentPayable = m_rent.RentPerDay *
m_rent.No_of_rent_days;
            if (CustomerBadHistoryLevel > 0)
            {
                /*PENALTY OF 20 % OF RENT MUST
BE APPLIED FOR CUSTOMERS WITH BAD HISTORY
LEVEL 2*/
                if(m_Customer.BadHistoryLevel == 2)
                {
                    m_rent.PenaltyFee = m_rent.RentPayable
* 0.2;
                }
```

```
                    /*PENALTY OF 10 % OF RENT MUST
BE APPLIED FOR CUSTOMERS WITH BAD HISTORY
LEVEL 1*/
                    if(m_Customer.BadHistoryLevel == 1)
                    {
                        m_rent.PenaltyFee = m_rent.RentPayable
* 0.1;
                    }
                }
                else
                {
                }
                    /*TOTAL AMOUNT PAYABLE IS
CALCULATED AS THE SUM OF RENT PAYABLE
AND PENALTY IF ANY.*/
                    m_rent.TotalAmountPayable =
m_rent.RentPerDay + m_rent.PenaltyFee;
            }
            else
            {
            }
            return m_application.Status;
        }
        /* End - EA generated code for Activities and
Interactions */
    }
```

```
//end RentalSystem
```

Notes

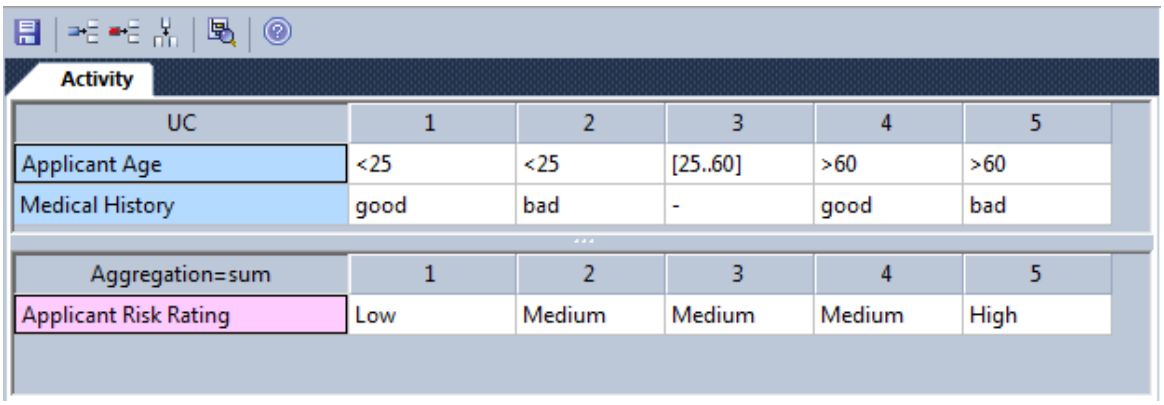
- Business Rule Modeling is available in the Unified Edition and the Ultimate Edition of Enterprise Architect

Decision Models

Create Expressive Models of Juncture Points Using Decision Tables

Enterprise Architect provides the Decision Tables facility as a simple method of applying the Decision Model and Notation (DMN) standard to model how a decision is made, and generate code for the decisions.

This example illustrates how you could model a decision on an insurance Risk Rating based on the applicant's age and medical history.



On the diagram containing the Activity or Action element that owns the table, you can replace the element shape with the Decision Table itself, showing the rules as either columns or rows. You specify the table format, or whether to redisplay the element shape, using options on the element context menu.

Enterprise Architect also supports a more extensive and complete implementation of the Decision Model and Notation (DMN) standard; see the *DMN Modeling* section

of the Help.

Access

Ribbon	Design > Element > Decisions > Find Decision Models
Context Menu	On a diagram: • Right-click on Activity element | Simple Decision Table | Decision Table or • Right-click on Action element | Simple Decision Table | Decision Table

Notes

- The Decision Tables facility is available in the Corporate, Unified and Ultimate Editions of Enterprise Architect

Decision Table Editor

When you select to create a new Decision Table for an Activity element, or edit an existing one, the name of the Activity element displays in the tab under the Decision Table toolbar.

Access

Ribbon	Design > Element > Decisions > Find Decision Models
Context Menu	On a diagram: Right-click on Activity element | Simple Decision Table | Decision Table or Right-click on Action element | Simple Decision Table | Decision Table

Complete the Decision Table

Task	Action
Enter a	You can enter a condition into the table in

condition	a number of ways: • Type the text of the condition into the first available row, overtyping the *Add Condition* text • Drag onto the 'Add Condition' cell the appropriate behavioral object from the Browser window, such as a model element, Activity Parameter, operation or attribute, or • Click on the 'Add Condition' cell and press Ctrl+Space; Intelli-sense is invoked to provide a list of possible objects that you can select from to complete the condition
Add another condition	You can add further condition rows to the list in any of these ways: • Click on the  icon underneath the list of conditions • Right-click anywhere on the condition panel and select the 'Add Condition' option, or • Click on the  (Add Condition) icon in the Decision Table toolbar Each condition that you add is bound to the previous conditions, so that they are evaluated in combination.

List allowable values for the condition	(Optional) You can provide a comma-separated value (CSV) list of the values that you could assign to a condition. If you do this, those values are offered as suggestions to enter in the value fields along the same row. The allowable values column is hidden by default. To display it, click on the  icon in the Decision Table toolbar. In the allowable values cell against each condition, type in all the possible values separated by commas. These values can include operators such as >, <, = and ... . Attributes dragged from the Browser window and classified by an Enumeration will complete the cell automatically and make the field read only.
Set condition values	Each of the conditions you set have defined values that act as a decision point. For example, a person under the age of 18 years old cannot hire a car, so the values of 'Yes' and 'No' are decision points for the condition '18 or over'. In each of the columns 1, 2, 3 and so on, provide a condition value by either: • Typing the value into the cell (including operators such as >, <, = and ...) or

	• Right-clicking on the cell and selecting the 'Allowable Values | <value>' option (as set in the allowable values cell)
Add further condition value columns	To add further value columns to the table, either: • Right-click anywhere on the condition panel and select the 'Add Column' option, or • Click on the  (Add Column) icon in the Decision Table toolbar These options add the column to both the 'Condition' and 'Conclusion' panels.
Set the Policy and Completeness	The 'Decision Hit Policy' defines if and how the conditions are combined in making a decision. 'Completeness' identifies whether or not the decision set is complete. To set these two flags, right-click on the 'Condition' header (top left corner of the 'Condition' panel) and select the options: • 'Hit Policy | <value>' and • 'Completeness | Complete' or 'Incomplete' If you do not want to display the completeness, select the 'Completeness | Clear' option.

Delete Column or Condition	To remove a condition that no longer applies, or a column of values that are no longer tested, right-click on the appropriate column or row and select the options: • Delete Condition or • Delete Column A prompt displays to confirm the deletion. You cannot delete the title column or condition number row, nor can you delete the allowable values column.
Enter a Conclusion	You can enter a conclusion into the table in a number of ways: • Type the text of the conclusion into the first available row, over typing the *Add Conclusion* text • Drag onto the Add Conclusion cell the appropriate behavioral object from the Browser window such as a model element, Activity Parameter, Operation or Attribute
Add Further Conclusions	You can add further conclusion rows to the list in any of these ways: • Click on the  icon underneath the list of conclusions

	• Right-click anywhere on the conclusion panel and select the 'Add Conclusion' option, or • Click on the  (Add Conclusion) icon in the Decision Table toolbar Each conclusion that you add is bound to the previous conclusions, so that the final decision takes them all into account.
List allowable values for the conclusion	(Optional) You can provide a comma-separated value (CSV) list of the values that you could assign to a conclusion. If you do this, those values are offered as suggestions to enter in the value fields along the same row. The allowable values column is hidden by default. To display it, click on the  icon in the Decision Table toolbar. In the allowable values cell against each conclusion, type in all the possible values separated by commas. These values can include operators such as >, <, = and ... . Attributes dragged from the Browser window and classified by an Enumeration will complete the cell automatically and make the field read only.
Set Conclusion	Each of the conclusions you set has values that define the outcome of the

values	conditional values in the same column, when the conditions are met. In each of the columns 1, 2, 3 and so on, provide a conclusion value by either: • Typing the value into the cell (including operators such as >, <, = and ...) or • Right-clicking on the cell and selecting the 'Allowable Values | <value>' option (as set in the allowable values cell)
Add Further Conclusion Columns	To add further value columns to the table, either: • Right-click anywhere on the conclusion panel and select the 'Add Column' option, or • Click on the  (Add Column) icon in the Decision Table toolbar These options add the column to both 'Condition' and 'Conclusion' panels.
Set theTable Aggregation Values	The 'Table Aggregation' value indicates how the conclusion values are to be combined to form the decision. Set the value by right-clicking on the 'Conclusion' column heading and selecting the'Aggregation | <value>' option.

	If you do not want to set the Table Aggregation, select the 'Aggregation | Clear' option.
Delete Column or Conclusion	To remove a conclusion that is no longer valid, or a column of result values that are no longer produced, right-click on the appropriate column or row and select the options: • Delete Conclusion or • Delete Column A prompt displays to confirm the deletion. You cannot delete the title column or conclusion number row, nor can you delete the allowable values column.
Check Object properties and location	If you have created a condition or conclusion by dragging an object from the Browser window, you can right-click on its row and: • Display the 'Properties' dialog for the object, by selecting the 'Properties' menu option • Locate the object in the model by selecting the 'Find in Project Browser' option
Save your	At regular intervals, and before you exit

changes	from the table, click on the Save icon in the Decision Table toolbar.

Notes

- The Decision Tables facility is available in the Unified and Ultimate Editions of Enterprise Architect

Code Generation from Decision Models

It is possible to generate the method code of a Class from an Activity element, based on the logic defined in its Decision Table, by placing the Activity as a child of the Class element from which the code is generated.

Set up Class and generate code

Step	Action
1	Create an Activity with a Decision Table.
2	Make the Activity element a child of a Class element.
3	Select the Class and press F11 to open the 'Generate Code' dialog, and generate the code.
4	Press F12 to view the generated source code.

Decision Table Code Templates

Code Generation from a Decision Table automatically applies these EASL code generation templates:

- Behavior Body
- Decision Table
- Decision Logic
- Decision Condition
- Decision Action

Notes

- Currently C++ is the only language implemented in the EASL templates for Decision Table code generation
- You can use ActivityParameters to define method parameters
- You can add Attributes to the Activity element to define local variables
- The Decision Tables facility is available in the Unified and Ultimate Editions of Enterprise Architect

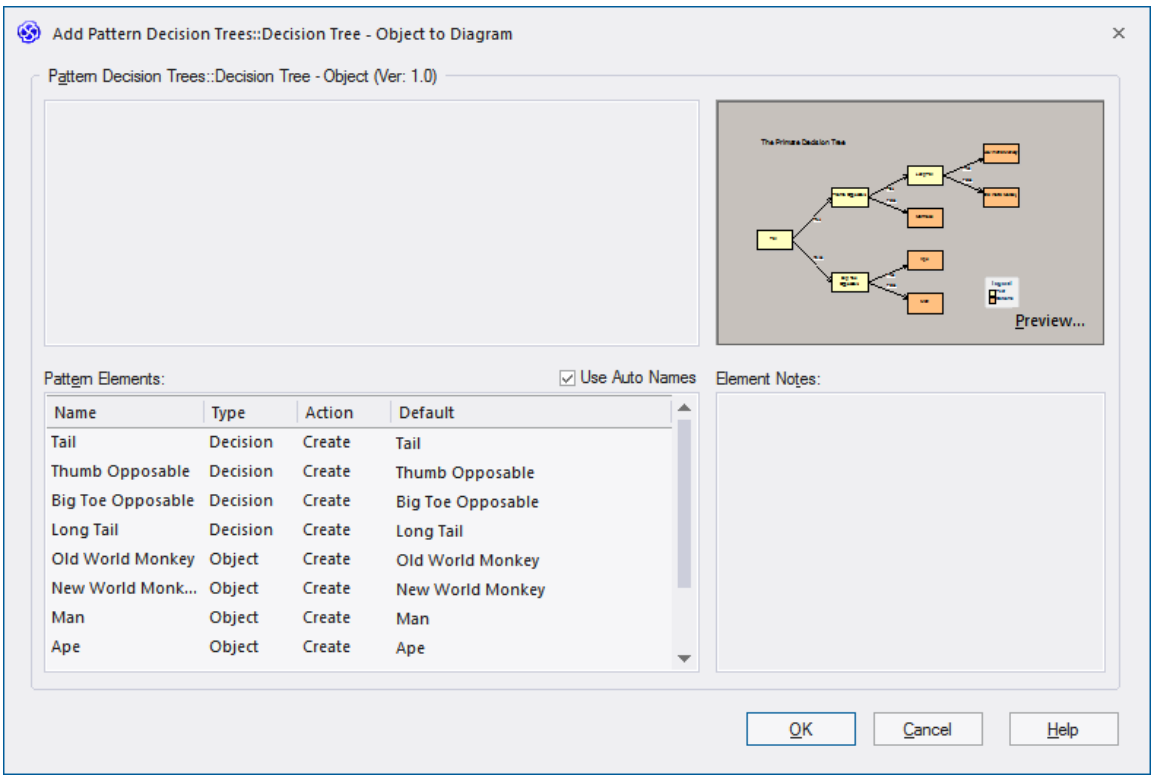
Decision Tree

Represent and Analyze Decisions in a Tree Structure

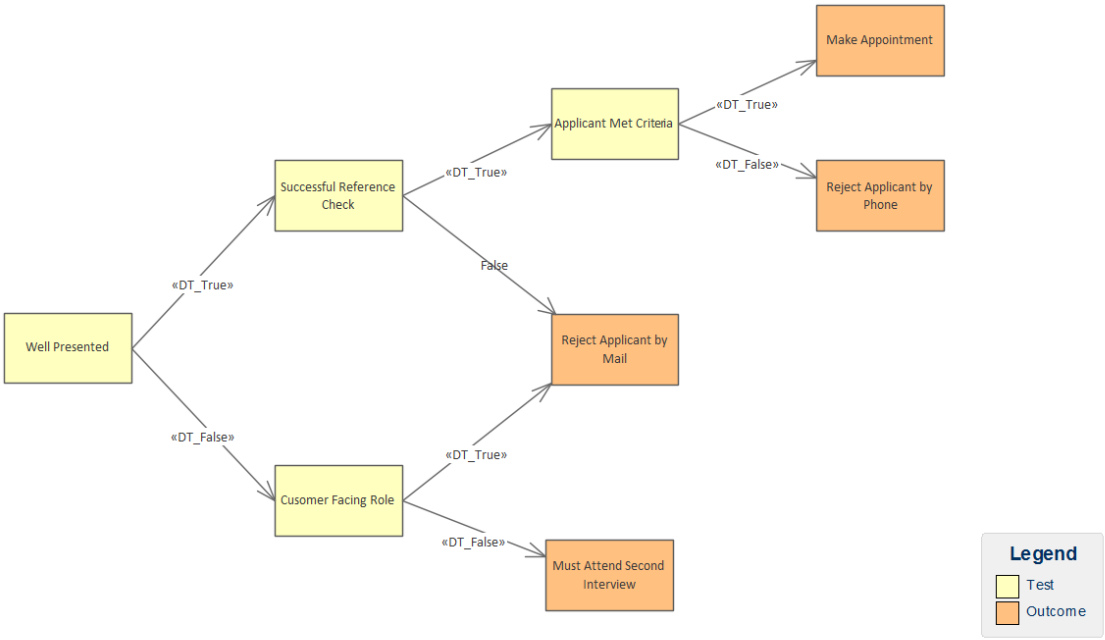
A Decision Tree uses a visual notation to represent a series of decisions and possible outcomes. It can be used in either a descriptive or predictive manner to visualize outcomes and decision points.

Creating a Decision Tree

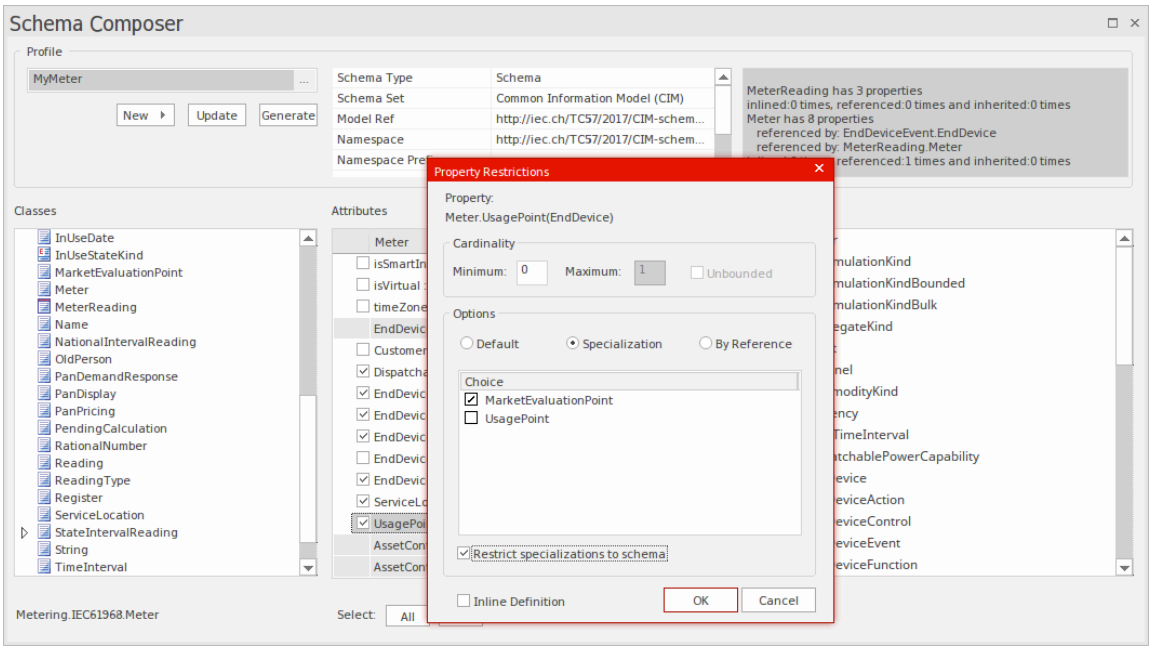
1. Create a new Package called 'Decision Tree', followed by a Decision Tree diagram called 'Key Decisions'.
2. Select the Pattern 'Decision Tree - Object' and place it on the Diagram View.



Modify the 'Primate Decision Tree' template created by the Pattern to create a Decision Tree related to the interview process for staff at a restaurant.



XML Schema (XSD)



Structural models in Enterprise Architect, especially Class models, are frequently used to define the meta-model of some domain of interest. For example a meta-model can be defined using a Class model to rigidly define the objects, data, relationships and types that make up the domain of Geospatial information. Likewise, models can be (and are) built to describe domains such as Water Management, Health, Retail, Insurance, Car Registration, Entertainment and many more.

These models are extremely valuable and frequently represent a significant investment in time and money by either commercial or standards based organizations. An important part of realizing the benefit of these models, in particular where information must be exchanged between multiple parties, is in the definition of schema (often XSD based) that codify how a message should be formed to be

conformant to the underlying meta-model. Traditionally, such message schema are written by hand, based on the meta-model. This is generally a laborious and error prone exercise.

Enterprise Architect has a long history of being associated with the development of both commercial and standards based meta-models, and there are many examples of models defined in Enterprise Architect model files that are used to specify the exact construction of an information domain of interest.

The Schema Composer in Enterprise Architect has been built to take maximum advantage of models stored in an Enterprise Architect model file or repository (or Cloud based server) by streamlining the conversion of model information into schemas that comply with the naming standards and format of a variety of popular industry meta-models. This approach drastically reduces the time taken to form a valid schema and eliminates human error in transcribing model information into schema text.

The current version of the Schema Composer supports XSD generation for a number of technologies, and in addition supports the customization of output by integrating tightly with both the Automation Interface and the Add-In framework. In this manner it is possible to use one of the schema generators supplied 'As-Is' or to write a custom generator using JavaScript, or to go further and fully customize the process by writing a suitable Add-In in a language of choice.

In addition to the new Schema Composer, Enterprise

Architect also supports the modeling of XSD and WSDL definitions using UML Profiles that support explicit modeling of the relevant types. This is sometimes necessary when building a complex XSD or WSDL from scratch and needing to have a fully worked out visual model of the final schema. Note that as Enterprise Architect also supports the import of XSD documents, it is possible to produce a schema using the Schema Composer, and then for documentation and visualization purposes (or even for further customization), import that schema back into either the current or a different model.

Additional topics included in the Schema Engineering section are devoted to the Meta Object Facility (MOF), the Ontology Definition Metamodel (ODM) and the National Information Exchange Model (NIEM). The section on NIEM is quite extensive, as Enterprise Architect includes many features necessary to model and work with NIEM domains and schema. As with some of the other technologies, there is in addition a downloadable version of the NIEM core as an Enterprise Architect model.

Getting Started

Selecting the Perspective

Enterprise Architect partitions the tool's extensive features into Perspectives, which ensures that you can focus on a specific task and work with the tools you need without the distraction of other features. To work with the Case Management Model and Notation features you first need to select one of these Perspectives:

- <perspective name> > Software Engineering > XML Schema or WSDL or XSLT
- <perspective name> > Information Exchange > NIEM

Setting the Perspective ensures that the schema modeling diagrams, their tool boxes and other features of the Perspective will be available by default.

Example Diagram

The Schema Composer

XSD Models

XSL Transforms

XML Validation

XML Service Oriented Architecture

Meta Object Facility

Example Diagram

The Schema Composer

Seamlessly Model Schema Compliant Message Definitions in a SImple and Productive Tool

The Schema Composer is a versatile tool for quickly and easily defining a variety of formal schema from a model. Due to the unique nature of the Schema Composer, it is not necessary to use a profile or stereotyped elements when building the definition of an XSD (or other) document. This greatly enhances the re-usability of the underlying model and helps alleviate the complexity that arises when dealing directly with XSD or other element types and restrictions.

Many industries have worked hard over the last decade to define shared meta-models specific to their industry, and it is these models that now form the basis for contractual information sharing across organizations and across geographic borders. A typical usage scenario of the Schema Composer is in the creation of message definitions (schema) to exchange information between organizations, ensuring that such messages comply with the underlying meta-model that has been adopted by the involved parties.

When information is shared between organizations, it is frequently the case that only a subset of the full meta-model is required, but it is essential that what is shared conforms precisely to the agreed meta-model. In this case the Schema Composer is the perfect tool for deriving contractual schema based on sub-sets and restricted data sets that take a 'slice'

through the meta-model as a whole.

The Schema Composer avoids the common 'pain points' of working with XSD and other schema languages directly:

- There is no need to create a relatively complex XSD model composed of specific XSD elements, in addition to your 'normal' business and data models, to define the required data, its associations and references, and any restrictions or conditions
- You do not need to understand how to use the XSD elements and apply the XSD naming rules and conventions to correctly construct such models; formatting and naming rules as specified by the supported standards are automatically taken care of

The Schema Composer greatly simplifies the process of creating standards compliant schema in a re-usable and accessible manner. In this illustration, you can see how a simple Class diagram is used as the source for the Schema Composer to generate XML Schema.

Schema Composer

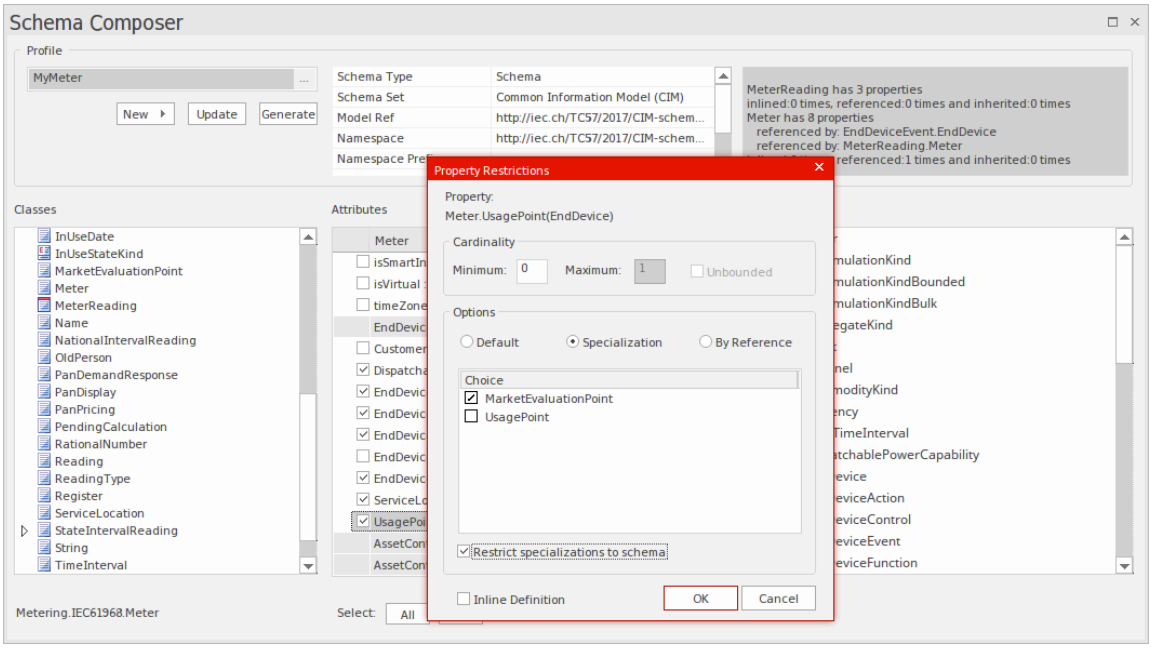


Figure shows a Schema Composition for the Process Order domain in the Example model.

Access

Ribbon	Develop > Schema Modeling > Schema Composer > Open Schema Composer

Benefits

The Schema Composer:

- Operates on a Class model rather than an XML schema profile

- Relieves you of the XSD-specific design and schema generation decisions, whilst still ensuring consistency across the profile
- Can operate on a generic Class model to provide generic XSD documents
- Is most useful when operating on industry standard Class models that have specific domain based meaning
- In most circumstances operates on a full model from which a subset of properties from selected Classes are drawn to build specific messages, to communicate only what is necessary for the information to send or request
- For standards such as NIEM, will generate a new sub-model as part of a broader NIEM compliant schema definition

Standards that the Schema Composer currently supports include:

- The Common Information Model (CIM)
- National Information Exchange Modeling (NIEM)
- United Nations Center for Trade Facilitation and Electronic Business (UN/CEFACT) Modeling Methodology (UMM), specifically the Naming and Design Rules (NDR) 2.1 and 3.0
- Universal Business Language (UBL), specifically the Naming and Design Rules (NDR) 3.0

The Schema Composer also helps you to build a definition of the same message using different formats such as:

- XSD

- RDFS
- JSON

In addition the Schema Composer

- Supports formats implemented using a custom Add-In that takes advantage of the Schema Composer automation interface
- Has built-in support for various serialization formats and styles used by different industry models

Notes

The Schema Composer is supported in the Corporate, Unified and Ultimate Editions of Enterprise Architect

Schema Composer Profiles

Schema Composer profiles are the configuration files that describe the elements and restrictions that will make up a particular schema or sub-model. Profiles are generally tied to a particular technology such as the Common Information Model (CIM) or the UML Profile for Core Components (UPCC), and the interpretation of the material within the Profile and the nature of the schema or sub-model published will be dependent on the technology-specific generator used. While Enterprise Architect supports a number of technologies 'out of the box' (and more are planned), it is also possible to customize the process by taking advantage of the extensive automation interface in Enterprise Architect to leverage the rich content of Schema Composer Profiles under your own terms, either in an Add-In or script.

Schema Profiles

A Schema Composer profile comes in two forms. Each form fulfills a particular system requirement - Schema generation (xsd, rdfs, json) and sub model creation. When you create a profile in the Schema Composer you choose which form to use based on your needs. A single profile in the Schema Composer can be used to either compose a schema, *in its common forms,* or create a UML sub-model from a core model.

Profile Types

Type	Description
Model Transform	A profile of this type is used to generate a sub-model from a core model.
Schema	A profile of this type is used to generate a schema; typically an XSD schema representing messages, but also other formats such as JSON object notation and resource descriptor formats.

Schema Composition Methodologies

National Information Exchange Model (NIEM)

Enterprise Architect provides a NIEM framework and Schema Composer for generation of sub-model and XML schemas.

Common Information Model (CIM)

Enterprise Architect Schema Composer supports provides the CIM standard out of the box, for composition of CIM compliant schema.

Universal Business Language (UBL)

Enterprise Architect provides a Universal Business Language framework, and the Schema Composer which provides the UBL standard for schema generation.

Core Component Technical Specification (CCTS) UN/CEFACT

Enterprise Architect provides a UML Profile for Core Components framework and Schema Composer. The Composer can generate business components libraries from core component libraries and simplifies the composition / publication of schema from message assemblies / business information entities.

Generic

Where a standard does not meet your requirements, the generic option provides a simpler choice for quick schema composition from your UML model. Typically you will model your own data library using UML Classes with attributes, associations, Aggregation and Inheritance. You can then use this model as the input to the Schema Composer.

EA Script Engine

Enterprise Architect provides a scripting engine that supports JavaScript, VBScript and JScript languages. The scripting engine is also integrated with the Schema Composer. When generating a schema, either for a particular standard or generic scheme, a script can be employed to perform the operation on its own or as a supplement to the options provided by the standard.

EA Add-In

Enterprise Architect provides Add-In integration with the Schema Composer. An Add-In can participate in the generation of the sub model or schema by registering its interest with Enterprise Architect. The Add-In can provide

options and alternatives to be listed in the 'Schema Generation' dialog, and will be invoked should its options be chosen. The Add-In can access the content of the profile using the Schema Composer automation interfaces.

Create a Schema Profile

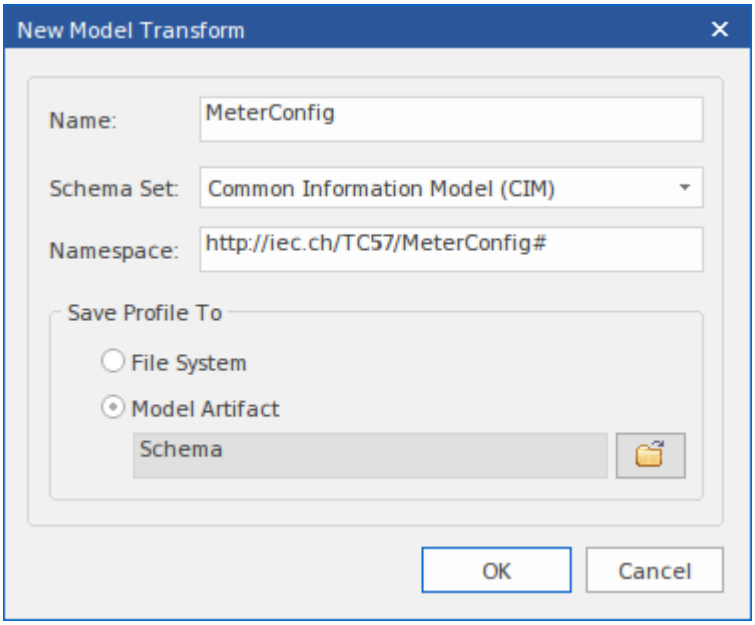
A schema profile identifies the name, technology and content of the schema as a precursor to defining how the schema is generated. You can create and edit as many schema profiles as you need. Schema profiles are bound to a single technology and will either map to a generated schema or a sub-setting transform.

Access

Ribbon	Develop > Schema Modeling > Schema Composer > Open Schema Composer

Creating a new Profile

If you are creating a schema for a particular technology, start by opening a model that has the required meta-model loaded. Sparx Systems make a number of meta-models available when using the Model Wizard and/or the Sparx RAS/Cloud services. Follow these steps to build a new Profile. With the Schema Composer displayed, click on the 'New' button and select the profile type, either Schema or Transform.



The New Profile Screen

Option	Action
Schema Set	Select the standard to use, or choose the 'Generic' option.
Namespace	Depending on the standard you have selected this field might take an automatic value or remain blank. Provide a relevant namespace if blank. Refer to the next section for a description of how namespaces are managed in the Schema Composer.
Save profile	Profiles can be stored in the file system or

To:	in the model. Profiles stored in the model can be shared with others, while file system profiles are private.
OK	Click on this button to edit the new schema in the Composer.

Namespaces

When a new profile is created, you specify the target namespace and the namespace prefix. Schemas typically involve multiple namespaces and the Schema Composer provides support for this within a single model. The scheme by which namespaces are identified is the presence of two specific properties on a Package. The properties are 'URI', which specifies the namespace, and 'Alias', which provides the namespace prefix. The properties can be present on the immediate Package or a parent Package. When present, elements of the Class will take that namespace. Where no namespace exists, Classes will take the target namespace specified when the profile was created.

Save the profile

Click the Update button to save the profile you have just

created.

Notes

- The process of creating and generating schema for NIEM has additional notes in the *NIEM* Help topic
- The Schema Composer is supported in the Corporate, Unified and Ultimate Editions of Enterprise Architect

Schema Compositions

A schema composition refers to a restricted set of elements taken from the model that together describe a unique entity that has no equivalent in the model. Commonly, schema compositions are used to generate schema files such as XSD files. In contrast, model compositions are used to configure the material as the basis of a subset 'transform' - for example when creating a NIEM model subset.

Define Schema Content

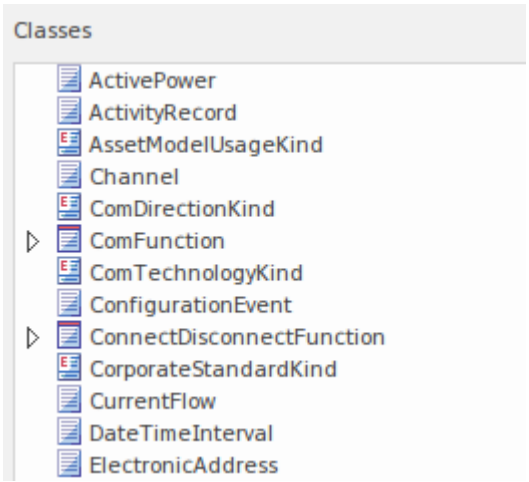
These steps walk you through the basic procedure of composing types in a Schema profile and show how you can restrict the content of elements to meet the message requirements.

Add Classes

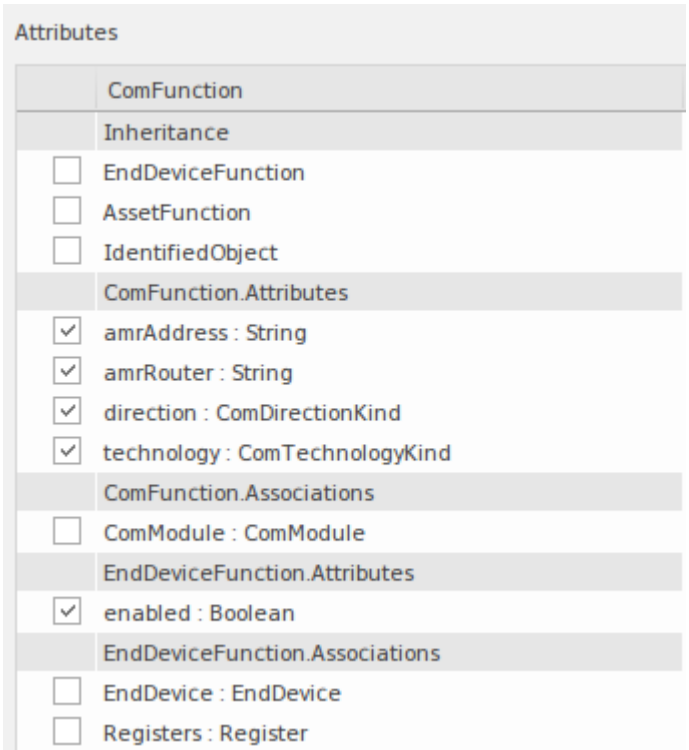
Drag the required Class elements from the Browser window into the 'Classes' panel. As you add a Class:

- Its ancestry is listed in the 'Inheritance' section in the middle panel
- Its attributes are listed under the 'Inheritance' section, with a blank checkbox against each one; Association and Aggregation entries are named according to the role name on the connector
- Its model structure path is shown underneath the 'Classes'

panel

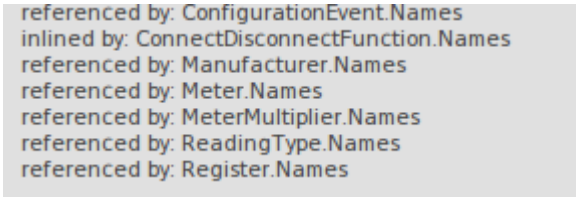


Select Properties



Any time you select a Class in the 'Classes' list, its attributes and model ancestry are listed in the 'Attributes' list. Select the checkbox against each attribute to define the elements of this type. When chosen, the attribute's type is added automatically to the schema, appearing in the 'Classes' list and the 'Schema' panel to the right.

When an attribute is unchecked, the type is not automatically removed. Types can be removed using the Class context menu. It is worth noting that each time a Class is selected, all references to the Class are displayed in the status panel, enabling you to quickly review any Class usage.



Inheritance

If you favor or foresee a need for inheritance in the schema you are preparing, it would make sense to begin the composition with ancestors first, then re-use these as child Classes are added. The method is not set in stone. You can switch from an inheritance model to an aggregated composition or vice-versa at any time. Here is a brief description of the provision of inheritance in the Schema Composer.

The Schema Composer offers flexibility in dealing with inheritance. For example, you can choose to aggregate selected attributes from the Class and its parent, while choosing to inherit the grandparent. However, when you choose to use inheritance, you choose to inherit the restricted form of that type as well. When an ancestor is selected in this list, the generated XML schema would show an extension element identifying this ancestor. Only one ancestor can be selected.

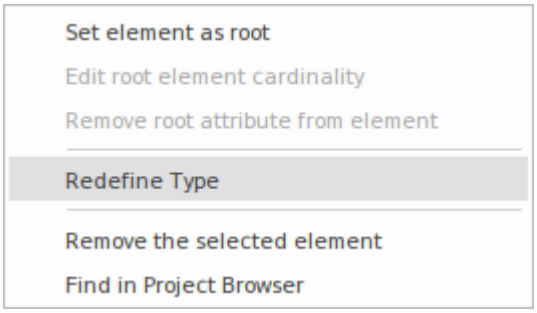
Click on the Update button to validate and save your schema profile.

If there are any problems with the profile, they are identified in the status panel in the top right of the screen.

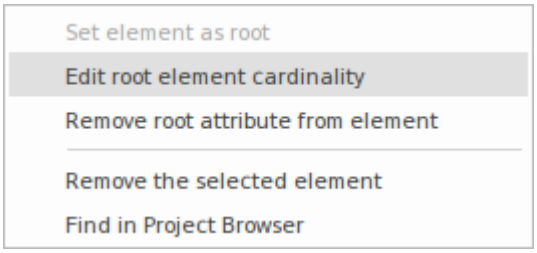
Redefined Types

One of the common issues with schema composition is the requirement to be able to vary a type description to meet various demands of the instances a schema describes. A vehicle, for example, might be described by its *brand, model and price* by an element of a *Truck* type, but by its *year, model and color* by an element of a *Sedan* type. The issue is that we might only have one actual *Vehicle* Class at our disposal. To address this the Schema Composer allows you to clone the *Vehicle* Class and give it another name. You can then assign this version of *Vehicle* to any property that has *Vehicle* as its type. The type created is only available within the domain of the schema - the model is untouched.

To create a new definition of a type, select the Class first in the 'Classes' list, then right-click on it and choose the 'Redefine Type' option. Enter a unique name for this type and press the Enter key. You can then define or restrict this type independently, the way you would for any Class.



Root elements

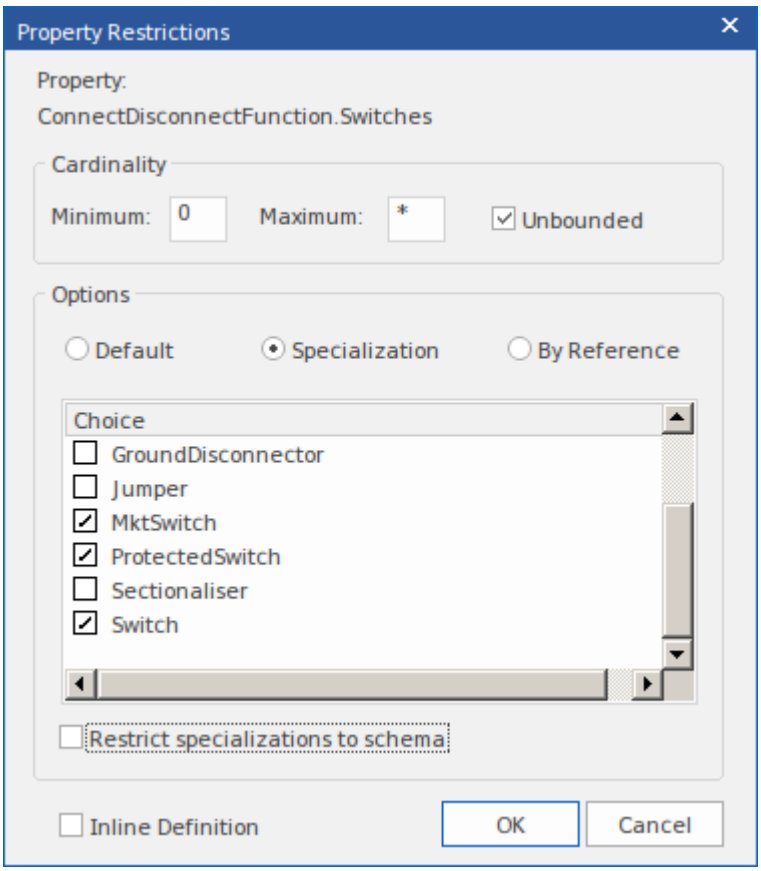


When the schema is generated, a single top level element representing the message is generated. The body or elements of this top level element are the Classes marked as root elements. The cardinality of these root elements can be adjusted. To mark a Class as a root element or restrict its cardinality, right-click on the Class in the list and use these context menu options:

- Set element as root - root elements form the body of the top level element representing the message / profile
- Edit root element cardinality - set the minimum and maximum number of instances
- Remove root attribute from element - removes the root mark from the Class
- Remove the selected element - delete the selected element from the schema

- Find in Project Browser - locate and highlight the element in the Browser window

Property Restrictions



In the 'Attributes' list, right-click on a selected property and use the context menu to add, edit or remove a property restriction. Use this feature to:

- Modify the property cardinality
- Redefine the type of the property
- Enable and limit the choices available for this property
- Mark a property to be emitted as an inline element definition

- Mark a property to be emitted 'By Reference'

Cardinality

The cardinality of a property can be further restricted from its model counterpart, but it cannot be less restrictive. The cardinality can be changed for any root element Class and any Class property.

Type redefinition

When a Class is redefined within the Schema Composer it creates a new type. The new type is a clone of the original, but has a name that is unique to the schema. A Payment enumeration type, for example, might be redefined as a CardPayment to better suit the schema purpose. The new type is a restriction of the original in that no new attributes can be added to it. Other properties that share this type and be similarly restricted by specifying the new type in their restriction dialog. Redefined types such as sub types can be offered as additional choice elements in the restriction of other properties.

Specializations

Where specializations of a property's type are present, those subtypes will be available in the 'Restriction' dialog. When more than one specialization is selected, these will appear as choice elements in the schema. When only one is chosen, the property will exhibit this subtype in the schema.

Inline Elements

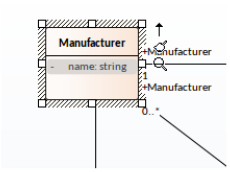
A property type will be emitted as an inline definition when this box is checked.

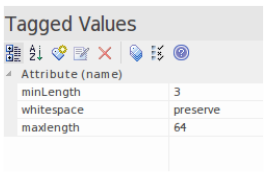
By Reference

A property will take the 'By reference' form when emitted in the schema. The 'By reference' form emits an inline complexType that defines a single attribute named 'ref' of type 'string'.

Property Constraints - Facets

Facets are supported in the Schema Composer Generic Profile. The sources of facets are the Tagged Values on a property. Tagged Values are recognized as facets if they name a constraining facet from the XML Schema specification; JSON validation keywords are also recognized.






```
<xs:complexType name="Manufacturer">
    <xs:sequence>
        <xs:element name="name" minOccurs="1" maxOccurs="1">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:maxLength value="64"/>
                    <xs:minLength value="3"/>
                    <xs:whiteSpace value="preserve"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:element>
```

Constraining Facets from XML Schema:

- length
- minLength

- maxLength
- pattern
- enumeration
- whiteSpace
- maxInclusive
- maxExclusive
- minExclusive
- minInclusive
- totalDigits
- fractionDigits

Validation keywords in JSON:

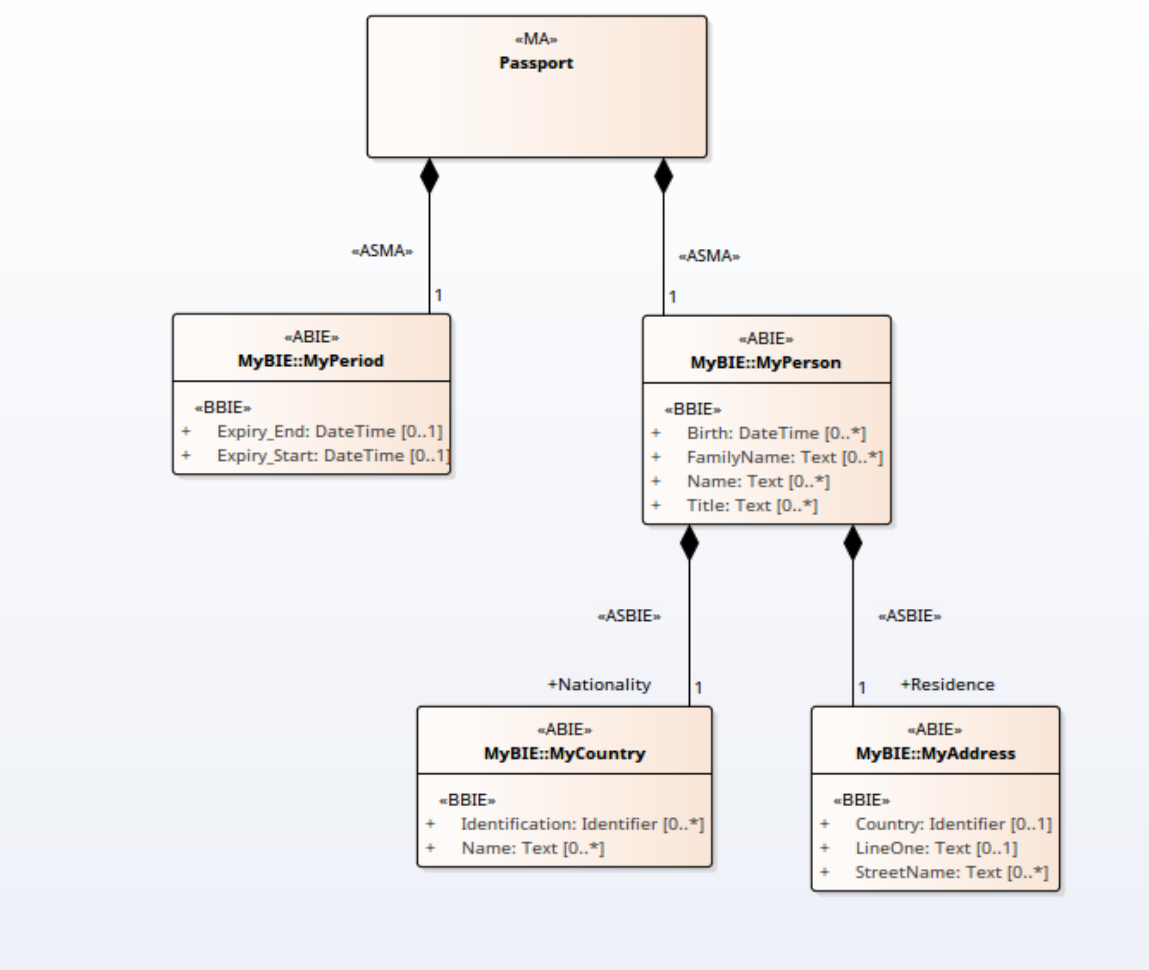
- Number and integer
  - multipleof
  - minimum
  - maximum
  - exclusiveMinimum
  - exclusiveMaximum
- strings
  - minLength
  - maxLength
  - pattern
- arrays
  - minItems
  - maxItems
  - uniqueItems

Class Diagrams

The Schema Composer also supports the creation of simple XSD and other formats from generic UML Classes. This is particularly useful when there is a need to export a Class definition in a generic manner for consumption by a script or web based tool, for example.

Generating schema from Class diagram

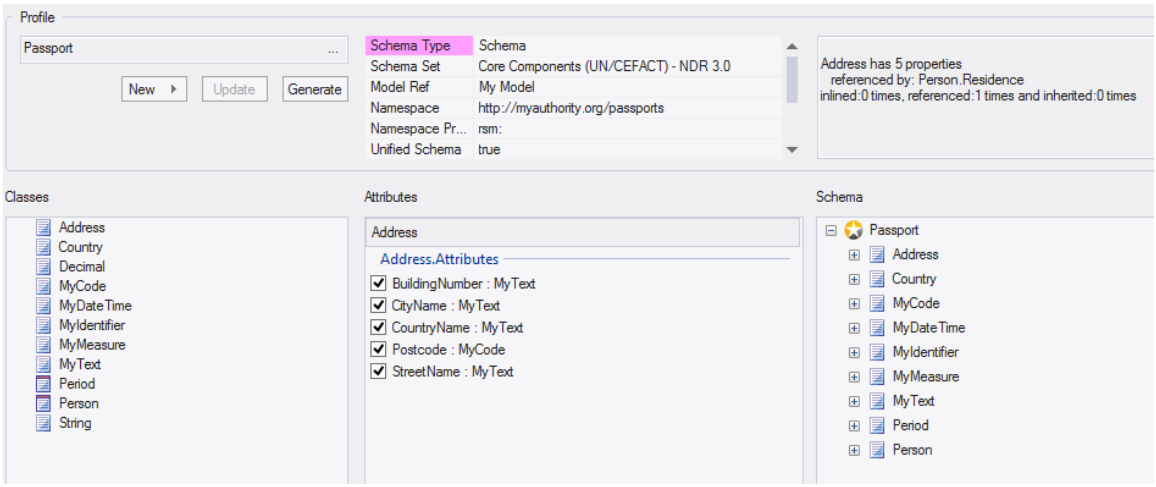
Users who prefer to use a modeling approach in composition can also use the Schema Composer for the generation of their chosen format(s). Any Class diagram can be loaded into the Schema Composer. This image illustrates a message composed using the UML Profile for Core Components, but it is not necessary for the message to be modeled according to a particular UML profile.



Loading the message into the Composer

The message is loaded into the Composer by selecting a Class on the diagram that represents the message and using its context menu to present the diagram as a schema in the Schema Composer. The selected Class will become the root element of the message and its relationships will shape the schema that is loaded.

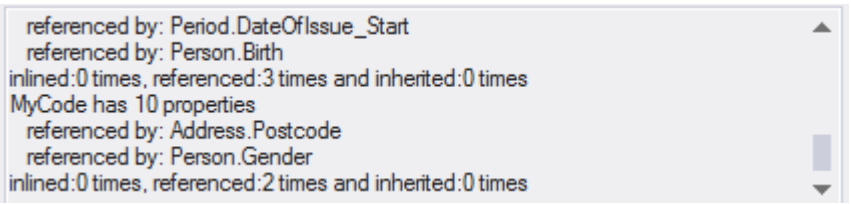
This is the Class diagram loaded into the Schema Composer


Profile
Passport
New
Update
Generate
Schema Type
Schema
Schema Set
Core Components (UN/CEFACT) - NDR 3.0
Model Ref
My Model
Namespace
http://myauthority.org/passports
Namespace Pr...
rsm:
Unified Schema
true
Address has 5 properties
referenced by: Person.Residence
inlined:0 times, referenced:1 times and inherited:0 times
Classes
Address
Country
Decimal
MyCode
MyDateTime
MyIdentifier
MyMeasure
MyText
Period
Person
String
Attributes
Address
Address.Attributes
BuildingNumber : MyText
CityName : MyText
CountryName : MyText
Postcode : MyCode
StreetName : MyText
Schema
Passport
Address
Country
MyCode
MyDateTime
MyIdentifier
MyMeasure
MyText
Period
Person

Schema Analysis

Analysis on the go

The Schema Composer performs analysis of each type as it is added to the schema, and whenever the Class is selected. The System Output window will show how many, if any, references exist for the type, the number of times it is inherited and other helpful information. This illustration shows a message detailing the elements that are referencing the selected Class.



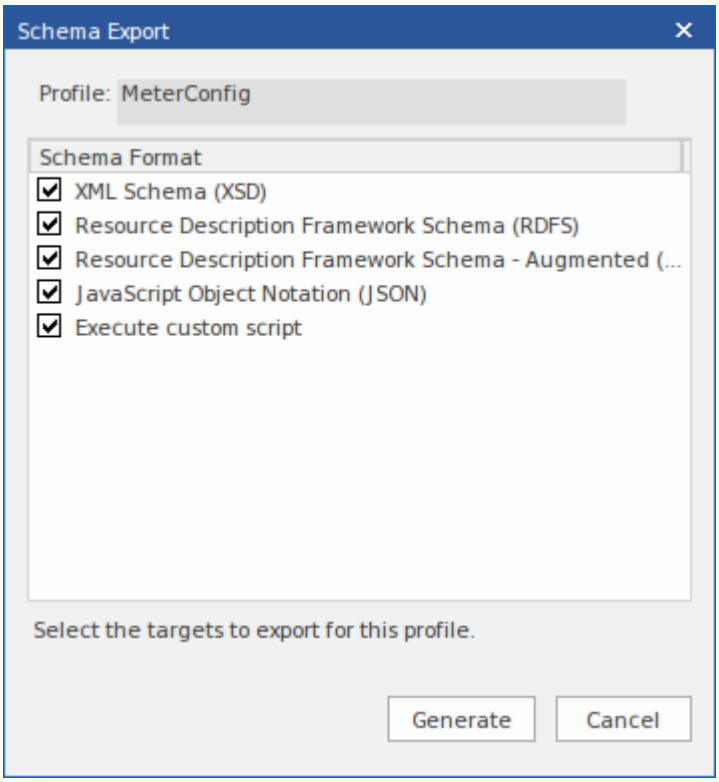
Validation on the go

The Schema Composer performs specific validation for a technology should one be assigned. This image shows warnings about missing Tagged Values for Classes in a schema built on the UN/CEFACT Core Components standard.

```
Warning: dataTypeQualifierTermName facet missing for class MyIdentifier
Warning: dataTypeQualifierTermName facet missing for class MyMeasure
Warning: dataTypeQualifierTermName facet missing for class MyText
Warning: dataTypeQualifierTermName facet missing for class MyCode
Warning: dataTypeQualifierTermName facet missing for class MyIdentifier
Warning: dataTypeQualifierTermName facet missing for class MyDateTime
Warning: dataTypeQualifierTermName facet missing for class MyMeasure
```

Generate Schema

Having designed a profile, at any stage, with a minimum of definitions and customizations, you can quickly and easily generate the schema or sub-model. Depending on the technology chosen and the profile type (schema or transform), the formats presented to you will vary. Note multiple formats can often be generated at once. And, of course, you can repeat the process easily, as the composition evolves or after design changes to the model.



Access

Ribbon	Develop > Schema Modeling > Schema

	Composer > Open Schema Composer : Generate

Notes

- The Schema Composer is supported in the Corporate, Unified and Ultimate Editions of Enterprise Architect

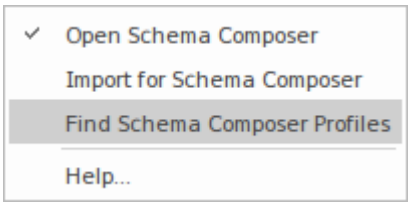
Select a Schema Profile

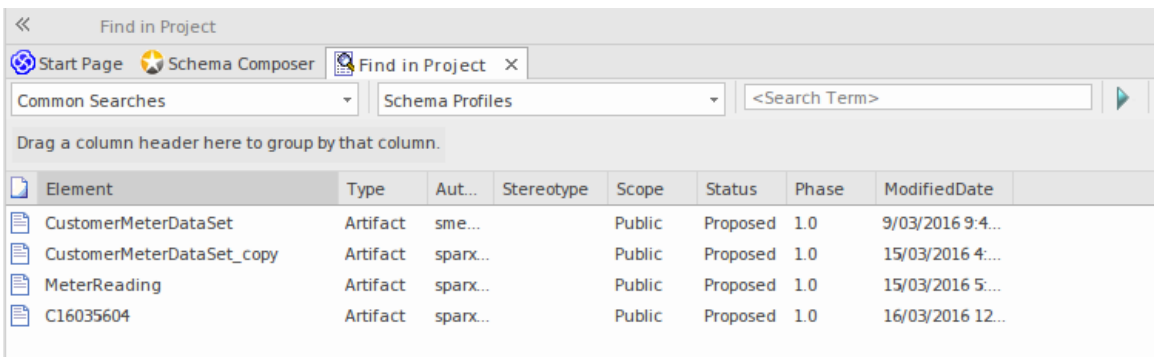
Access

Ribbon	Develop > Schema Modeling > Schema Composer > Find Schema Composer Profiles Develop > Schema Modeling > Schema Composer (icon)

Locating Schema Profiles in the Model

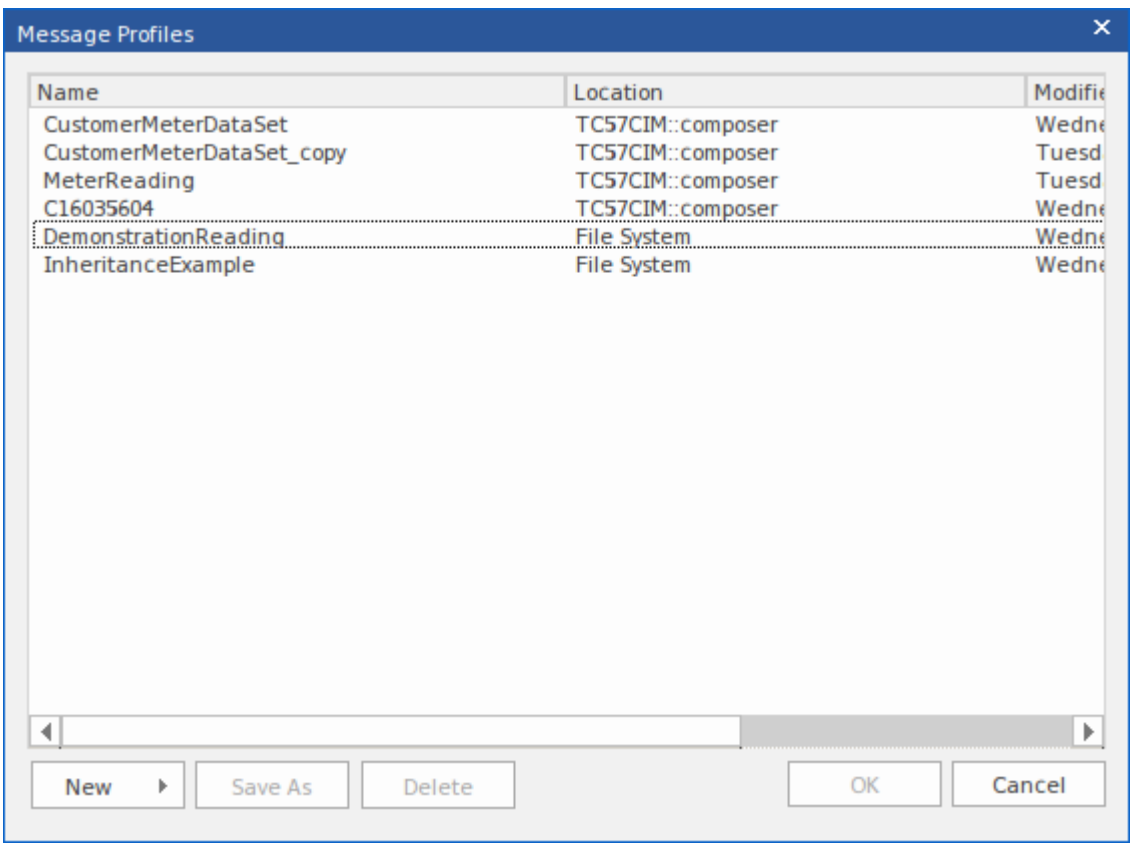
Schema profiles can be located quickly from the Schema Composer drop-down menu in the ribbon. The menu provides quick access to the existing profiles in the model.





Locating Schema Profiles in the Schema Composer

Schema profiles can stored in either the model and file system. You can easily locate every profile by authored in your model by opening the Schema Composer and clicking the select profile button (the button with the ellipsis '...') This brings up a list of all the profiles for this model and indicates where they live; the model or the file system. You might have many Enterprise Architect models that you work with, but only those file system profiles that relate to the open model will be listed.



Generate Schema File

Having defined a schema profile and added the necessary elements and restrictions, you can quickly and easily generate the schema(s). XML schema generation is available in all technologies, but each technology might support additional formats.

Access

Ribbon	Develop > Schema Modeling > Schema Composer > Open Schema Composer : Generate

Schema Formats

Select the checkbox against each schema format to export.

Schema Format	Details
CIM	• XML Schema (XSD) • Resource Description Framework Schema (RDFS)

	• Resource Description Framework Schema - Augmented (RDFS) • JavaScript Object Notation (JSON) • Execute Custom Script
UN/CEFACT NDR 3.0	• XML Schema (XSD) • Execute Custom Script
UN/CEFACT NDR 2.1	• XML Schema (XSD) • Execute Custom Script
Generic	• XML Schema (XSD) • Resource Description Framework Schema (RDFS) • JavaScript Object Notation (JSON) • Execute Custom Script
UBL 2.1	• XML Schema (XSD) • Execute Custom Script
Execute Custom Script	Although the Schema Composer can generate schema for a number of recognized standards, it also features a scripting solution for those users who want control over the format and medium of the schema. When you specify a script to the generator, it is referring to a language script such as JavaScript that

	exists in your model. How and what the script produces is pretty much up to you. How the script accesses the schema in the Schema Composer is documented in the Schema Composer Scripting Integration.

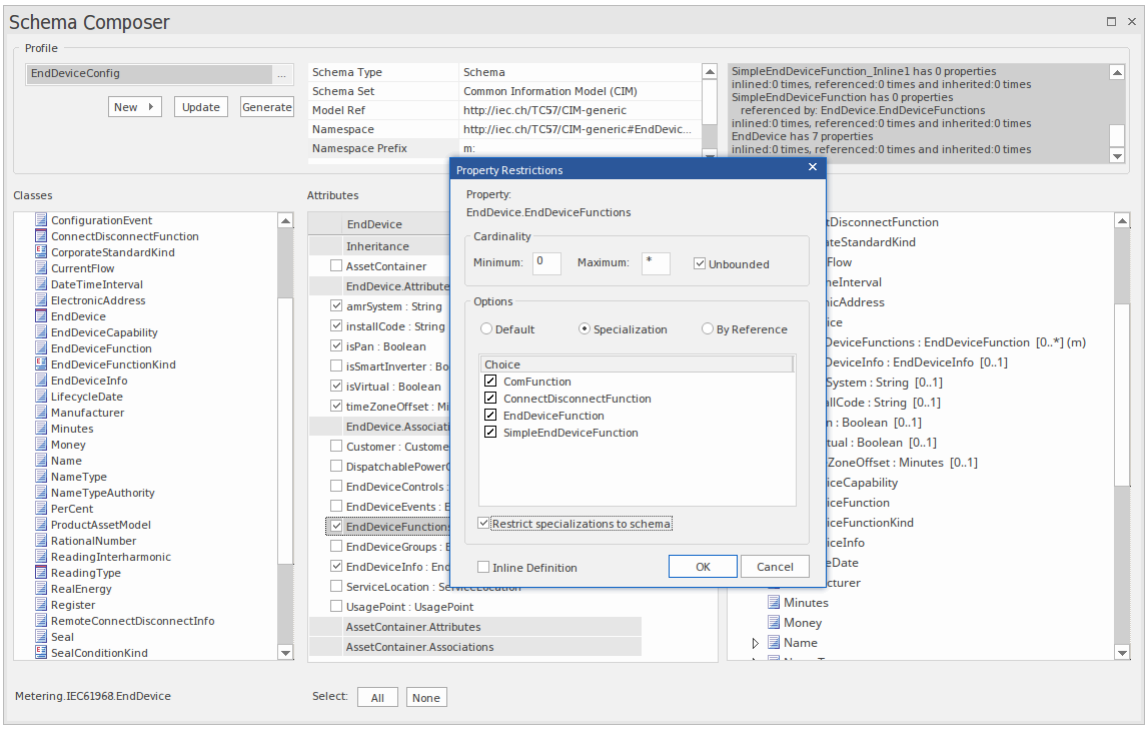
Generate

Click on this button to generate the schema.
Use a file browser to locate and open the schema files.

Notes

- You can edit and validate XML documents including XSD schema, using Enterprise Architect
- You can set Enterprise Architect as the default document handler for XML documents

CIM Schema Guide



This guide describes the creation and generation of a CIM compliant XML Schema.

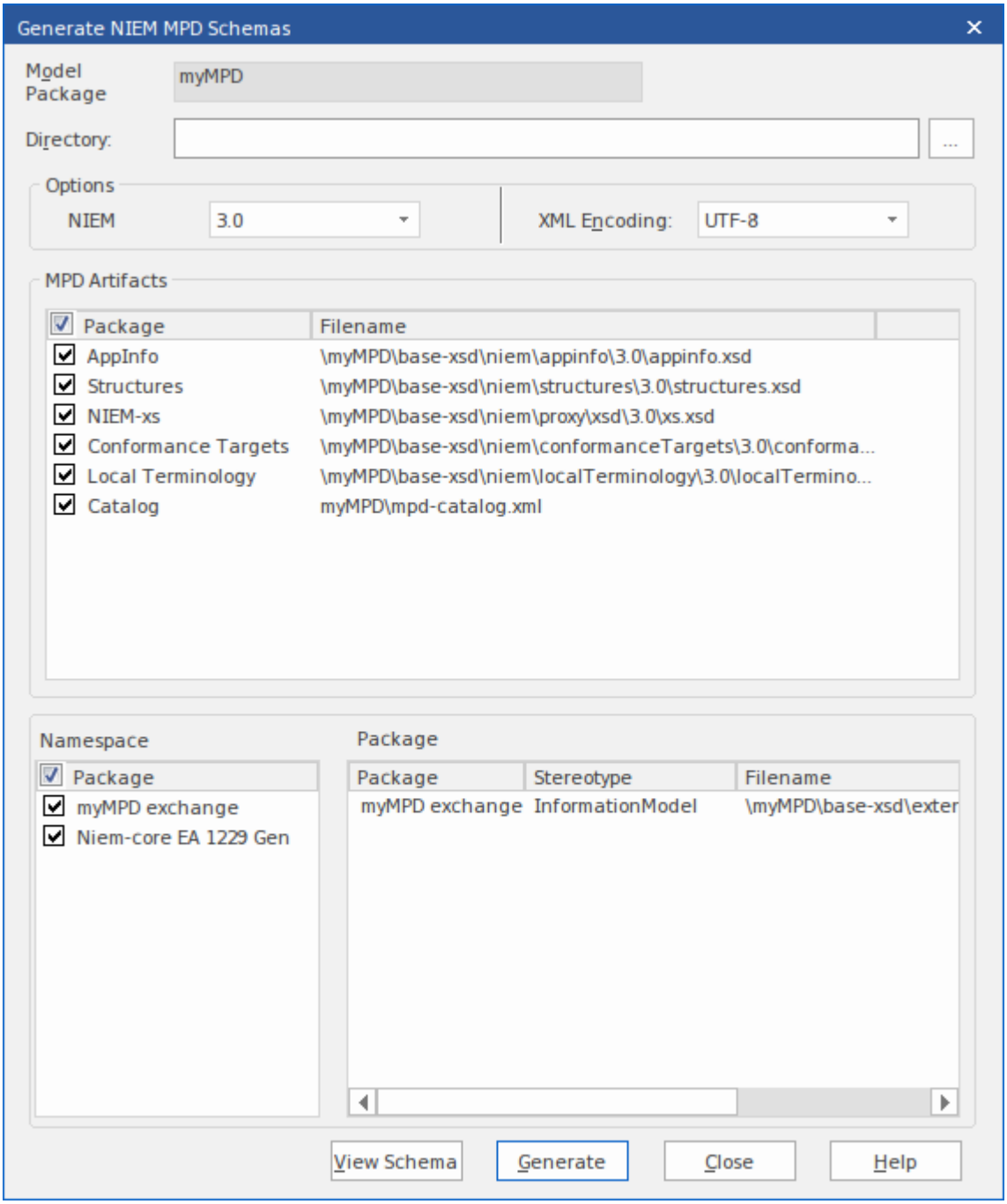
Create a CIM message

Step	Action
1	Display the Schema Composer.
2	Click 'New | Schema'.
3	Enter a unique name for this CIM schema (message).

4	Select the Common Information Model.
5	Drag the initial CIM Class(es) into the Class window that best represents the message. Set *root* elements appropriately using the context menu.
6	If you want to compose this type using inheritance, select a single ancestor from the inheritance list.
7	Use the checkboxes on the attributes of each Class to define the set of properties that will describe this message or schema.
8	Apply restrictions to elements using the context menu on the property.
9	Click update to save the message.
10	Click the Generate button and choose the schema formats to export.

NIEM Schema Guide

Generation of schema for NIEM is accomplished on either an instance of a ModelPackageDescription Class (NIEM 3.0 and above) or a «ModelPackageDescription» stereotyped component (NIEM 2.1). In either case a dialog is presented that allows you to configure the schema produced.



Generate NIEM Schemas (NIEM 2.1)

Click on a component with a «ModelPackageDescription» stereotype and select one of these options:

Ribbon	Specialize > Technologies > NIEM 2.1 >

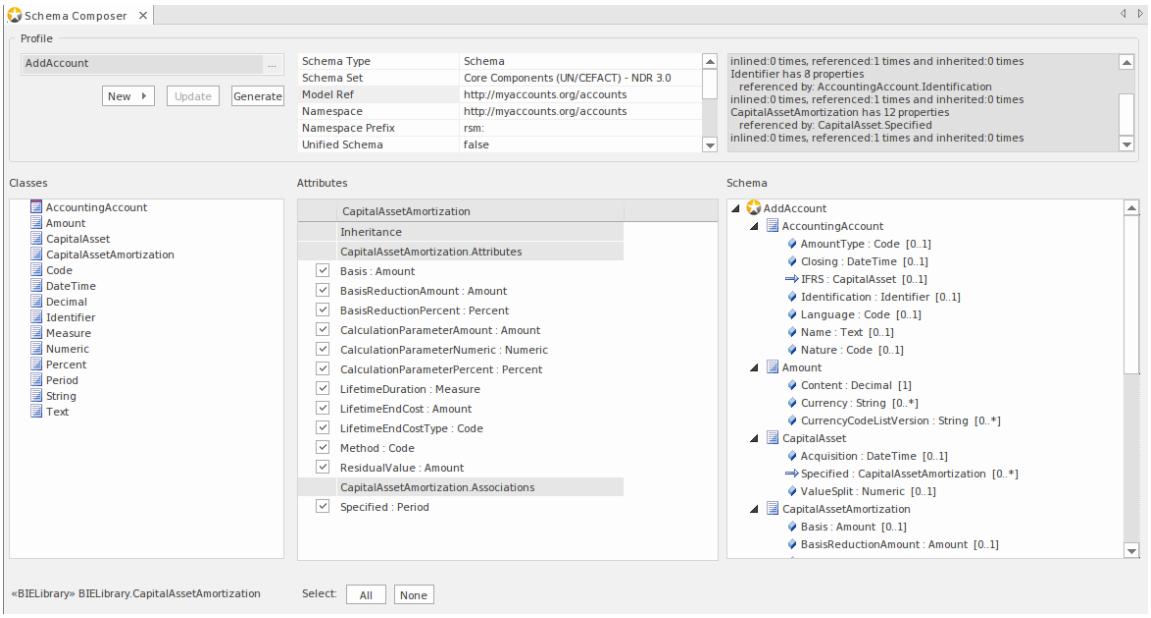
	Generate NIEM 2.1 Schema or
Context Menu	Right-click on the element | Specialize | NIEM 2.1 | Generate NIEM 2.1 Schema

Generate NIEM Schemas (NIEM 3.0 and above)

Click on any object instance of a ModelPackageDescription Class and select one of these options:

Ribbon	Specialize > Technologies > NIEM > Generate NIEM Schema
Context Menu	Right-click on the element | Specialize | NIEM | Generate NIEM Schema

UPCC Schema Guide



This guide describes the composition and generation of a UPCC compliant XML schema.

Creation of UPCC Schema

Step	Action
1	Display the Schema Composer.
2	Click 'New | Schema'.
3	Enter a unique name for the schema.

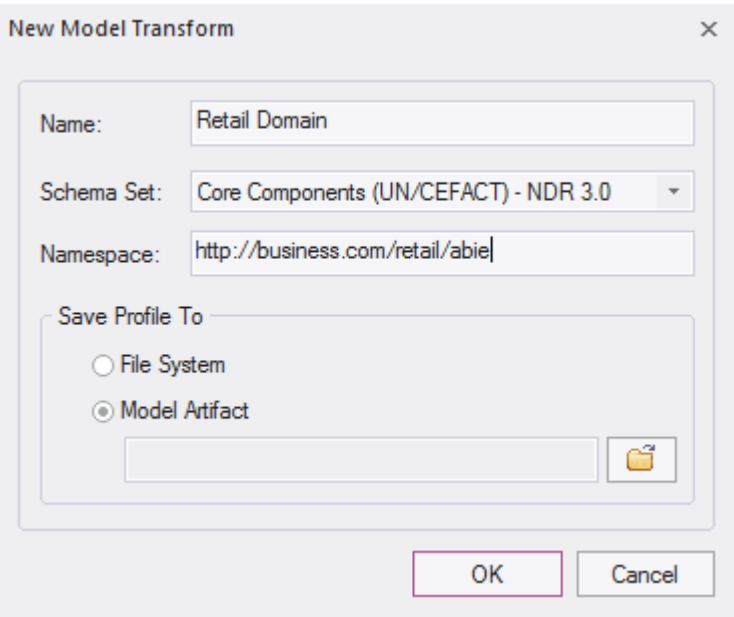
4	Select the UPCC Naming and Design rules to use, from the list of standards.
5	Drag one or more <ABIE> components from a <BIE library> into the Class list.
6	Set the Class as a root element using the context menu.
7	Select required attributes (Referenced types are added to the schema).
8	Click on the Update button to save changes.
9	Click on the Generate button and select 'XML Schema'. Click on the OK button.

Model Compositions

The model composition feature of the Schema Composer is useful for creating a sub-model from a core model. This can be as simple as generating a single business Package from a core Package (*CDT library to BDT library transform in UN/CEFACT Core Components standard)* or creating a complete sub-model from a large core model.

The enormity of such a task can be daunting and error prone; *ensuring every type that is referenced by the sub-model is included by the sub-model,* for example. The Schema Composer addresses this problem by automatically working out the dependencies and adding them to the schema for you where necessary.

Create Transform

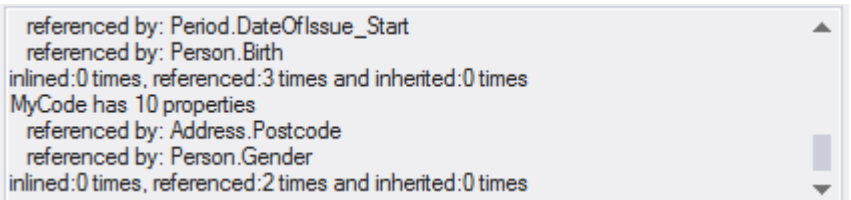


Define Model Content

Content is added to the model by dropping Classes from the model on to the Schema Composer Class window and choosing which properties to include. The resultant type can mirror the core type or a provide a simpler classification. When a property is included, the Schema Composer will check the property type and if the type is missing will add it to the schema automatically.

Reference checking

When a property is excluded that was previously included in the schema and is no longer referenced, the property type is not automatically removed. However, the Schema Composer will always show the number of references for a type if you select it in the Class window. Types that show no references at all can easily be removed.



Summary

The process of composing a sub-model is summarized here:

1. Create a Schemer Composer Transform Profile
2. Create elements by dropping Classes from the model into the schema.
3. Include / exclude required properties.
4. Generate the sub model.

Access

Ribbon	Develop > Schema Modeling > Schema Composer > Open Schema Composer : New > Transform

Generate a Model Subset (Transform)

Having defined the content of your sub-model or library and applied any restrictions, you can now generate the model. The model transforms that can be performed depend on the technology associated with the profile. Each technology and the transforms it supports are listed here:

Access

Ribbon	Develop > Schema Modeling > Schema Composer > Open Schema Composer : Generate

Model Transform

Select the model transform(s) to run.

Transform Option	Description
NIEM	**NIEM Model Subset** This option will generate a NIEM Model Subset containing the schema described

	by the profile. When you click on the OK button, you will be prompted to select the target Package for creation of the subset. The <<Namespace>> Packages that make up the subset will then be created at this location. If any of the subset Packages already exist at this location, their content will be added to. All subset Packages will have the 'defaultPurpose' Tagged Value set to 'subset'. **Execute custom script** A user defined language script such as JavaScript will be executed. The script can obtain access to the profile using the Schema Composer automation interfaces.
Generic	**Generic model Subset** You will be prompted for a target Package. This will be populated with the types from the schema. If a qualifier is entered this will be applied to the Classes generated. Any restrictions in the schema will also be applied. Types that exist in the target Package will be overwritten. New properties will be added. Types or properties that exist in the target but that no longer exist in the profile will not be

	removed by this process. **Execute custom script** A user defined language script such as JavaScript will be executed. The script can obtain access to the profile using the Schema Composer automation interfaces.
UN/CEFACT NDR 3.0	**BDT Library** A Business Datatype Library will be populated from core datatypes listed in the profile. Stereotypes will be transformed according to the CCTS specification. The types could be more restricted than their core counterparts. Properties of datatypes that exist will be overwritten. New properties and types will be added to the library. Types are matched by name and stereotype. Types or properties that exist in the target but no longer exist in the profile will not be removed by this process. **BIE Library** A Business Information Entity library will be populated from aggregated core components. listed in the profile. Stereotypes will be transformed according to the CCTS specification. Properties of datatypes that exist will be overwritten. New properties and types

	will be added to the library. Types are matched by name and stereotype. Types or properties that exist in the target but no longer exist in the profile will not be removed by this process. **Execute custom script** A user-defined language script such as JavaScript will be executed. The script can access the profile using the Schema Composer automation interfaces.
UN/CEFACT NDR 2.1	**UDT Library** Performs an unqualified copy of selected core datatypes to a UDT library. **QDT Library** A Qualified Business Datatype Library will be populated from core datatypes listed in the profile. The names of the resultant types will be qualified by the named qualifier in the profile. Stereotypes will be transformed according to the CCTS specification. Properties of datatypes that exist will be overwritten. New properties and types will be added to the library. Types are matched by name and stereotype. **BIE Library** A Business Information Entity library

	will be populated from aggregated core components listed in the profile. Stereotypes will be transformed according to the CCTS specification. Properties of datatypes that exist will be overwritten. New properties and types will be added to the library. Types are matched by name and stereotype. **Execute custom script** A user-defined language script such as JavaScript will be executed. The script can obtain access to the profile using the Schema Composer automation interfaces.

Generate

Click on the OK button to generate the schema. When the generation is complete, the message *Export of profile <name> completed* displays.

You can then expand the Package in the Browser window to see the generated UML model.

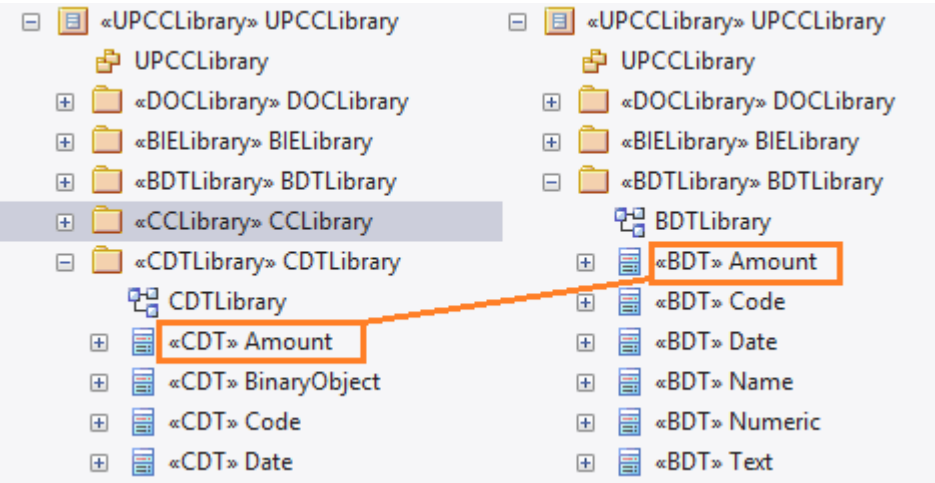
Notes

- The Schema Composer is supported in the Corporate, Unified and Ultimate Editions of Enterprise Architect

UML Profile for Core Components (UPCC)

The UPCC framework provides core component and core data type libraries and is available to Enterprise Architect users through the Model Wizard ('Create from Pattern' tab of the Start Page). Whether you model according to the UMM specification, or want to leverage the advantages this standard brings, or have a compliance requirement, to model with this technology you will require - as a minimum - a Business datatype library and a Business Information Entity library. The Schema Composer can generate these libraries for you.

This image shows a BDT library created from a UPCC Core CDT Library



Common Libraries

Libraries shared by both versions of the UML Profile for

Core Components.

Library	Description
CCLibrary	The CCTS core component library.
CDTLibrary	The CCTS core datatype library. It contains basic datatypes such as Amount, Code, Text and Graphic.
BIELibrary	A Business library containing ABIE entities based on CCLibrary components. The entities can be composed using the Schema Composer. These can also be modeled using the UML modeling tools available for the technology.
DOCLibrary	A Package typically used for the modeling of Message Assemblies. You can generate the schema for a Message Assembly by loading it into the Schema Composer.-

UPCC Libraries

The UML Profile for Core Components is available in two versions, NDR 3.0 and NDR 2.1. Both profiles describe a

common set of libraries, with some differences, as described here:

NDR 3.0

Library	Description
BDTLibrary	A Business library containing BDT types based on CDTLibrary types. The Schema Composer can be used to easily generate the content of a BDTLibrary from selected types in the core library.

NDR 2.1

Library	Description
UDTLibrary	An unqualified datatype library. Basically a mirror of the CDTLibrary for use in a business context. The Schema Composer can be used to easily generate the content of a UDTLibrary from selected types in the core library.

QDTLibrary	A qualified datatype library. The library contains restricted types based on the CDTLibrary with qualified type names. The Schema Composer can easily generate the content of a QDTLibrary from selected types in the core library.

UPCC Diagrams

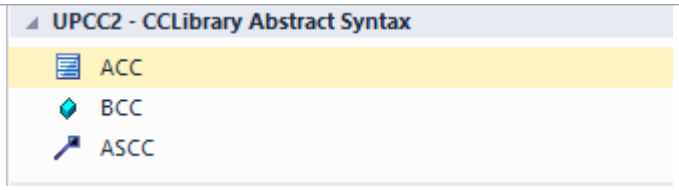
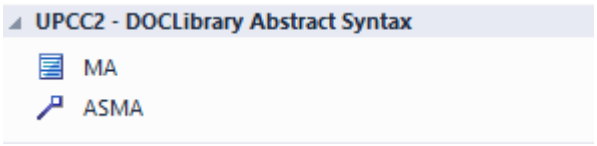
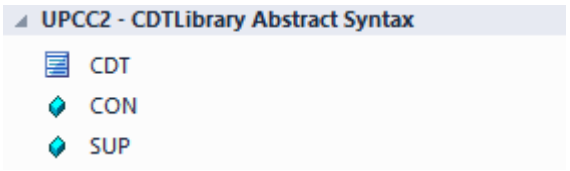
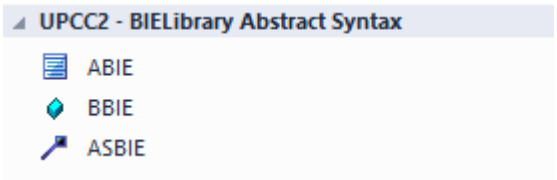
The UML profile for Core Components uses UML Class diagrams for composition of components. There are however specific toolboxes provided by the technology for each of its libraries.

UPCC Toolbox Pages

Common

In this notation, UPPCx represents the UPCC profile and x is the version of the NDR

Common	Description

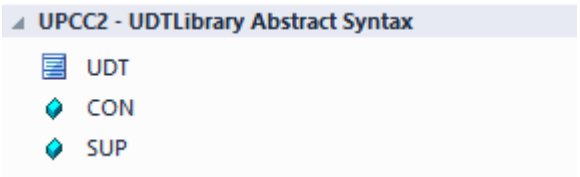
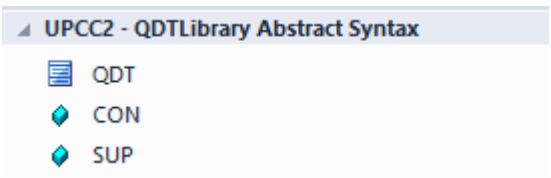
UPCCx - CCLibrary Abstract Syntax	
UPCCx - DOCLibrary Abstract Syntax	
UPCCx - CDTLibrary Abstract Syntax	
UPCCx - BIELibrary Abstract Syntax	

NDR 3.0

Library Syntax	Description
UPCC -	

BDTLibrary Abstract Syntax	

NDR 2.1

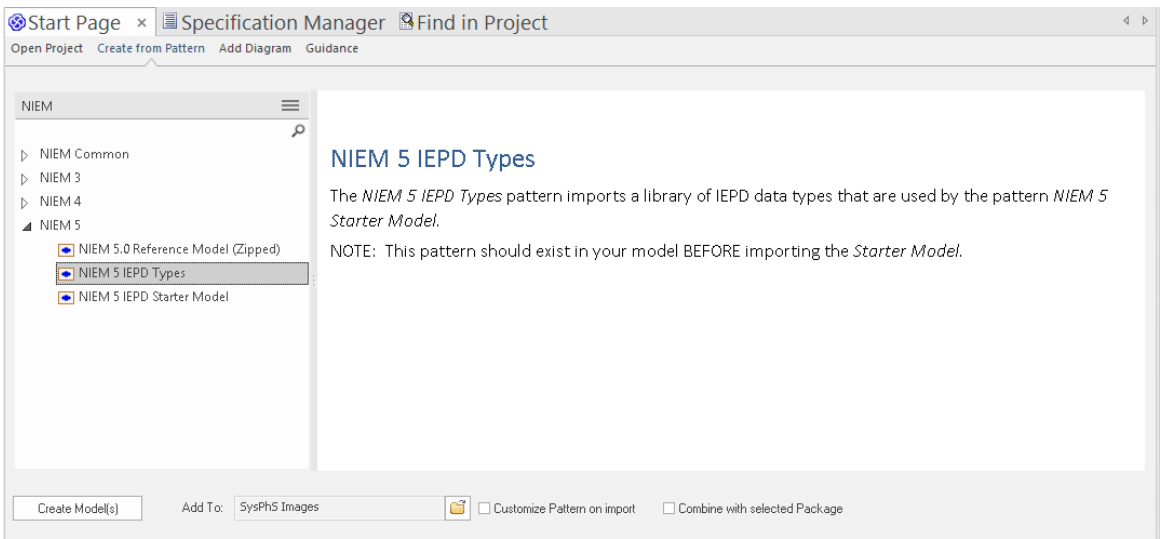
Library Syntax	Description
UPCC - UDTLibrary Abstract Syntax	
UPCC - QDTLibrary Abstract Syntax	

Available Frameworks

Using the Enterprise Architect Model Wizard (Start Page 'Create from Pattern' tab) you can deploy any of the frameworks supported by the Schema Composer - such as NIEM, CIM and CCTS - to your model in minutes, providing a multi-featured UML medium for modeling in those technologies.

The frameworks are also available directly from the Sparx Systems Reusable Asset Service (via the Cloud Server 'Cloud Connection' dialog and then 'Frameworks' on the Model Wizard).

Note: In addition to the custom frameworks such as CIM and NIEM, it is possible to use standard Class models to rapidly build generic Schemas, so if you are not targeting a particular meta-model, it might be simplest to model your data in UML and use the resultant model as input to the Schema Composer.



National Information Exchange Model (NIEM)

This is the National Information Exchange Model published by the NIEM Program Management Office (PMO).

Enterprise Architect provides these resources for modeling in NIEM:

- Provided Frameworks including NIEM core, NIEM domains, code lists and external schema adapters:
  - NIEM 2.1 modeled using NIEM-UML 1.0
  - NIEM 3.0 modeled using NIEM-UML 1.1
  - NIEM 3.1 modeled using NIEM-UML 1.1
  - NIEM 4.0 modeled using NIEM-UML 1.1
- NIEM subset creation:
  - The Schema Composer helps you create a subset of a NIEM conformant namespace
- NIEM schema generation:
  - Generation of complete NIEM IEPDs from a model Package description in either NIEM 2, NIEM 3 or NIEM 4 formats

Common Information Model (CIM)

This is the CIM specification published by International Electrotechnical Commission (IEC) Technical Committee 57.

Enterprise Architect provides these resources for modeling in CIM:

- Schema Composition
  - XML schema (XSD)
  - Resource Descriptor format (RDFS)
  - Resource Descriptor augmented format
  - JavaScript Object notation (JSON)
  - Add-In integration
  - Scripting integration

Universal Business Language (UBL)

UBL is a CCTS implementation published by OASIS that is proving popular with European governments for consolidating information exchange between agencies.

Enterprise Architect provides these resources for the composition of business documents using UBL:

- UML Framework
  - UBL 2.1 Main Document Libraries
  - UBL 2.1 Common Component Libraries
- Business Document Composition
  - Schema Composer for component composition
  - Schema Composer for document composition
  - Schema Composer for schema generation
  - Add-In integration
  - Scripting integration

Core Component Technical Specification (CCTS)

This is the CCTS specification published by UN/CEFACT.

Enterprise Architect provides these resources for modeling in CCTS:

- UML Frameworks:
  - UPCC 2.1 core component libraries
  - UPCC 3.0 core component libraries
  - UMM 2.0 business requirements, choreography and information views.
- Business Component Library Creation / Management
  - Schema Composer for ABIE and BDT composition
  - Add-In integration
  - Scripting integration
- Business Component Schema Composition
  - Schema Composer for XSD, JSON
  - Add-In integration
  - Scripting integration

Add-In Framework (Custom)

In addition to these methodologies, the Schema Composer integrates with the Enterprise Architect Automation Interface to support any individual or group in implementing

their own. An Add-In that registers its interest to Enterprise Architect in offering Schema generation capabilities will have the opportunity to offer any of its products in the Schema Composer Generation tool.

Scripting Framework (Custom)

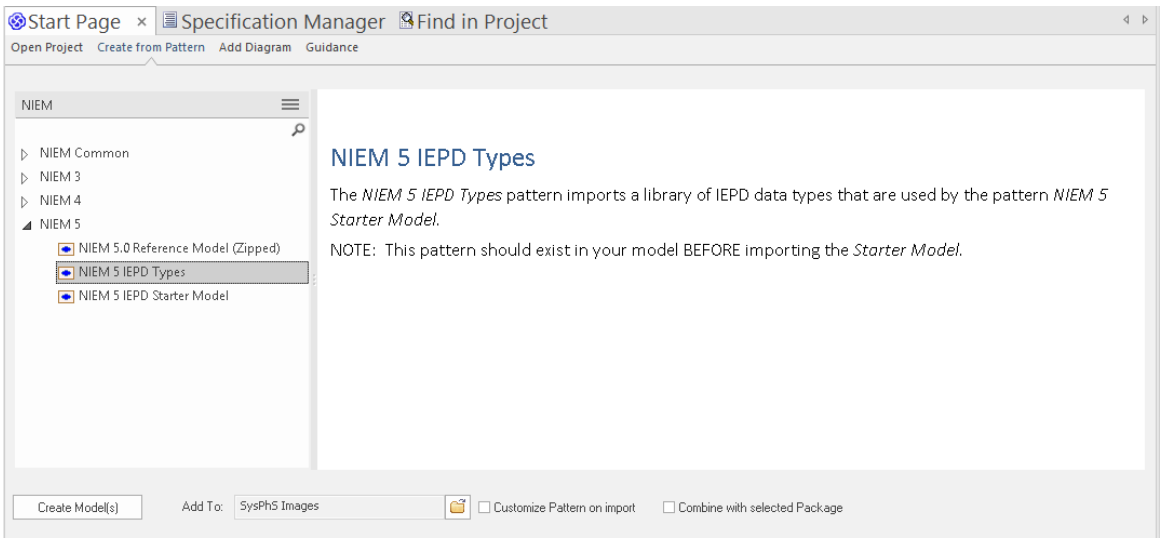
The Schema Composer also offers unconditional control over generation of schema for any profiles created with it. By writing their own script an author can access the definition of any schema and ultimately produce whatever documents they want, in a format of their choosing.

Install a Core Framework

Enterprise Architect provides a rich and diverse range of modeling technologies including every standard listed in the Schema Composer. These frameworks are available as UML models and/or MDG Technologies using Enterprise Architect's Model Wizard (Start Page 'Create from Pattern' tab). The models themselves are also directly accessible from Enterprise Architect's Reusable Asset Service.

Note: If you are modeling a generic solution and not directly using a core framework such as CIM or UBL, you do not need to install a core framework/model. In that case you are best served creating a data model using simple UML Classes with attributes.

Model Wizard



Access

Ribbon	Design > Package > Model Wizard
Context Menu	Right-click on Package | Add a Model using Wizard
Keyboard Shortcuts	Ctrl+Shift+M
Other	Browser window 'Hamburger' icon > New Model from Pattern

Note

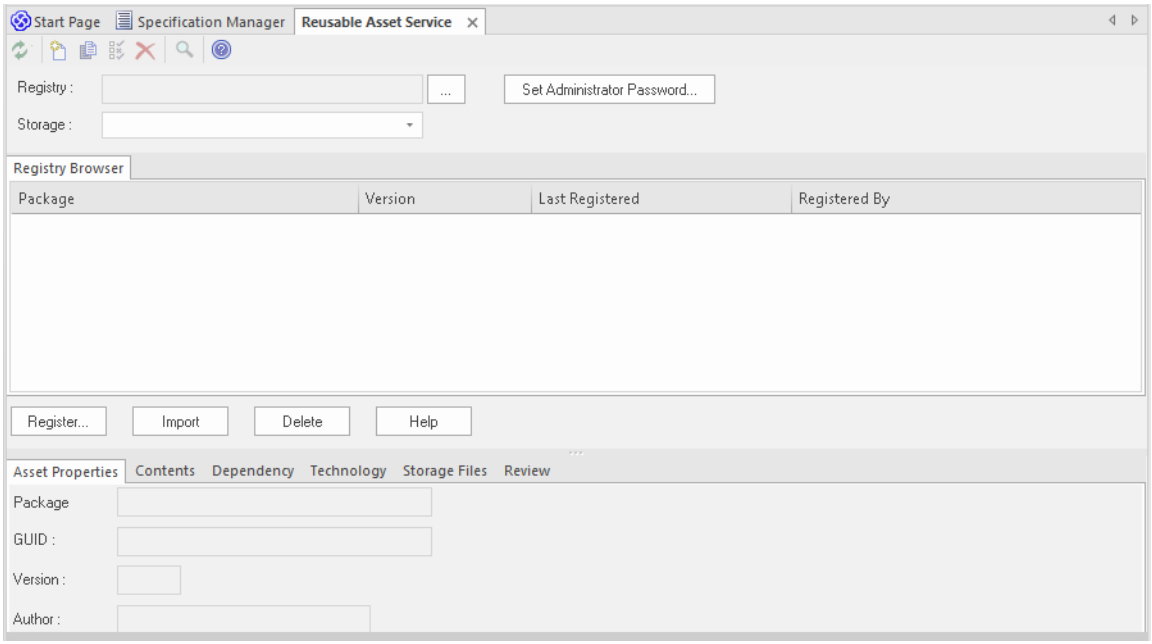
You can limit the MDG Technologies to use by selecting the ribbon option: 'Specialize > Technologies > Manage-Tech'. Here you can see which technologies are currently enabled.

Import Model

Ste	Action

p	
1	Display the Start Page 'Create from Pattern' tab (the Model Wizard).
2	Click on the <perspective name> button and select the required technology.
3	Highlight the Technology.
4	Select the Technology standards to import.
5	Clickon the Create Model(s) button to import the framework to your model.

Reusable Asset Service



Access

Ribbon	Publish > Model Exchange > Reusable Assets

Import Model

Step	Action
1	Connect to the Reusable Asset Service.

2	Choose from the available list of Repositories
3	Select the UML model Package
4	Click OK to import the selected Package to your model.

The Schema Importer

You can import Schemas compatible with the Schema Composer, into Enterprise Architect using the Schema Importer. The Schema Importer validates the Schema and creates a *Schema* type Schema Composer Profile upon successful validation, that can be viewed directly in the Schema Composer.

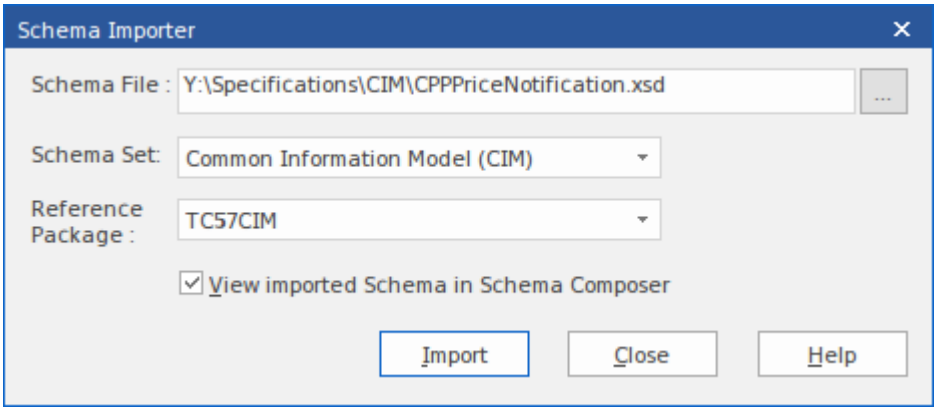
Currently, you can use the Schema Importer to import these Schemas:

- Common Information Model (CIM) specific XML Schema
- Common Information Model (CIM) specific RDFS XML

Access

Ribbon	Develop > Schema Modeling > Schema Composer > Import for Schema Composer

Import a Schema using the Schema Importer



Option	Action
Schema File	Type the directory path and filename from which to import the Schema file.
Schema Set	Select the type of Schema being imported. Currently the Schema Importer supports importing CIM specific: • XML Schema and • RDFS XML
Reference Package	Select the Package containing the common elements specific to the schema set. The Schema Importer will validate the elements in the Schema being imported against the elements in the reference Package.
View imported	Select this option to open the imported Profile in the Schema Composer.

Schema in Schema Composer	
Import	Click on this button to start the import process.
Close	Click on this button to close the 'Schema Importer' dialog.

Notes

- The progress of import will be displayed in the System Output window
- The Schema Composer will validate the Schema against the elements in the Reference Package before importing the Schema; if validation fails, the Schema elements that fail validation will be displayed in the System Output window and the import process will stop
- Double-click on a validation error entry in the System Output Window to open the Schema in Enterprise Architect's internal file editor and go to the source of the error
- If validation succeeds, the 'New Schema Definition' dialog displays, through which you can save the imported Profile in the file system or as an Artifact in the current

model

Schema Composer Automation Integration

The Schema Composer can be accessed from the Enterprise Architect Automation Interface. A client (script or Add-In) can obtain access to the interface using the 'SchemaComposer' property of the 'Repository' object. This interface is available when a Schema Composer has a profile loaded.

Schema Composer Addin Integration

Enterprise Architect Add-Ins can integrate with the Schema Composer by providing alternatives to offer users for the generation of schemas and sub models.

Schema Composer Scripting Integration

Although the Schema Composer provides out-of-the-box schema composition based on a variety of popular technologies, its scripting integration provides you with some flexibility in how you might go about implementing your own requirements. There are three ways in which you might leverage scripting within the Schema Composer:

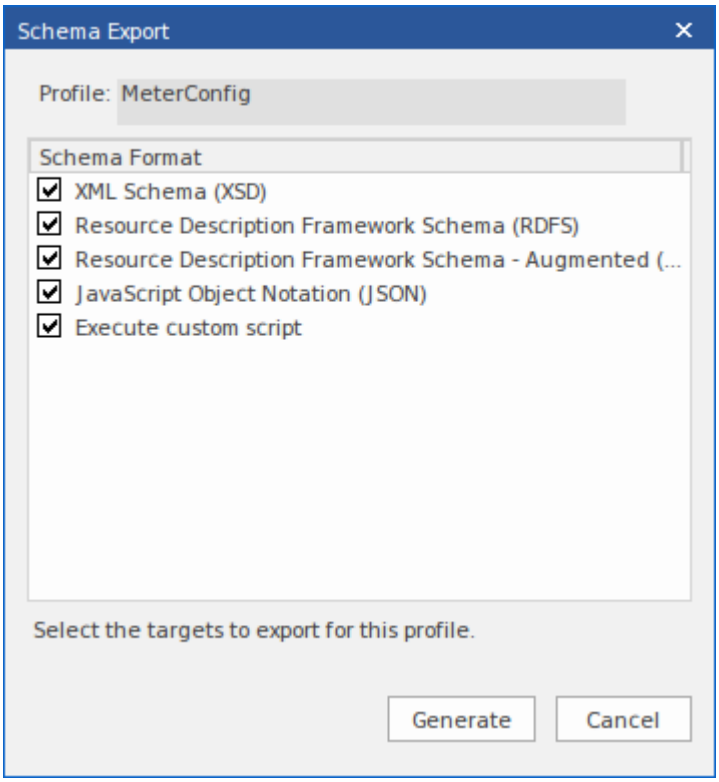
- Provide custom schema generation using a scripting language
- Provide custom model transformation using a scripting language
- Provide custom stereotype mapping to any standard model transform (such as UPCC)

Model Transformation by script

While the Schema Composer provides in-built transforms for various frameworks, you can always write your own, using the composition tools of the Composer to design the schema, then performing a custom transform with a hand crafted script.

Schema Generation by script

When you select a message in the Schema Composer and click generate, you are presented with a number of export formats. One of those choices is 'Execute custom script'



Schema Iteration Scripting Example

This example demonstrates accessing the Schema Composer in an Enterprise Architect script written in JavaScript. The script first obtains an interface to the Schema Composer and then traverses the schema, printing out the types and each of its properties.

```
/*
```

```
* Script Name: Example Schema Composer Script
* Author:  Sparx Systems
* Purpose: Demonstrate access to Schema Composer using automation and JavaScript
* Language: JavaScript
* Date:  2020
*/
function printType( xmlType, xmlns, uri)
{
	var xmlProp as EA.SchemaProperty;
	var xmlPropEnum as EA.SchemaPropEnum;
	var xmlChoiceEnum1 as EA.SchemaTypeEnum;
	var xmlChoiceEnum2 as EA.SchemaTypeEnum;

	Session.Output("Type: " + xmlType.TypeName + " in namespace: "  + xmlns + ":" + uri);
	xmlPropEnum = xmlType.Properties;
	if(xmlPropEnum)
	{
		xmlProp = xmlPropEnum.GetFirst();
		while(xmlProp)
		{
			if(xmlType.IsEnumeration())
			{
				Session.Output("    " + xmlProp.Name);
```

```
            }
            else
            {
                var sPropDesc = xmlProp.Name;
                sPropDesc += "::"
                if(xmlProp.IsPrimitive())
                    sPropDesc +=
xmlProp.PrimitiveType;
                else
                    sPropDesc += xmlProp.TypeName;

                if(xmlProp.IsByReference())
                {
                    sPropDesc += "(by reference)";
                }
                if(xmlProp.IsInline())
                {
                    sPropDesc += "(inline)";
                }
                Session.Output("   " + sPropDesc + ",
cardinality: " + xmlProp.Cardinality);

                xmlChoiceEnum1 = xmlProp.Choices;
                xmlChoiceEnum2 =
xmlProp.SchemaChoices;
```

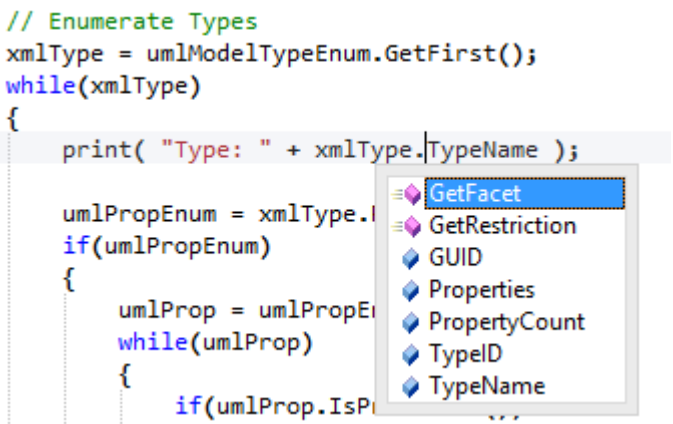
```
                var count = xmlChoiceEnum1.GetCount()
+ xmlChoiceEnum2.GetCount();
                if(count>1)
                {
                    Session.Output("   choice of: ");
                    xmlChoice =
xmlChoiceEnum1.GetFirst();
                    while(xmlChoice)
                    {
                        Session.Output("     " +
xmlChoice.TypeName);
                        xmlChoice =
xmlChoiceEnum1.GetNext();
                    }
                    xmlChoice =
xmlChoiceEnum2.GetFirst();
                    while(xmlChoice)
                    {
                        Session.Output("     " +
xmlChoice.TypeName);
                        xmlChoice =
xmlChoiceEnum2.GetNext();
                    }
                }
            }
            xmlProp = xmlPropEnum.GetNext();
```

```
            }
        }
}

function main()
{
        var schema as EA.SchemaComposer;
        var xmlType as EA.SchemaType;
        var xmlTypeEnum as EA.SchemaTypeEnum;
        var xmlNamespaceEnum as
EA.SchemaNamespaceEnum;
        var xmlNS as EA.SchemaNamespace;

        // Get SchemaComposer
        schema = Repository.SchemaComposer;

        // print the namespace references
        xmlNamespaceEnum = schema.Namespaces;
        if(xmlNamespaceEnum)
        {
            xmlNS = xmlNamespaceEnum.GetFirst();
            while(xmlNS)
            {
                Session.Output( "xmlns:" + xmlNS.Name + "
URI=" + xmlNS.URI);
```

```
                xmlNS = xmlNamespaceEnum.GetNext();
            }
        }
        // Get Schema Types Enumerator
        xmlTypeEnum = schema.SchemaTypes;
        xmlType = xmlTypeEnum.GetFirst();
        while(xmlType)
        {
            var xmlns = schema.GetNamespacePrefixForType( xmlType.TypeID );
            uri = schema.GetNamespaceForPrefix(xmlns);
            printType(xmlType, xmlns, uri);
            xmlType = xmlTypeEnum.GetNext();
        }
}

main();
```

Intelli-sense help in scripting

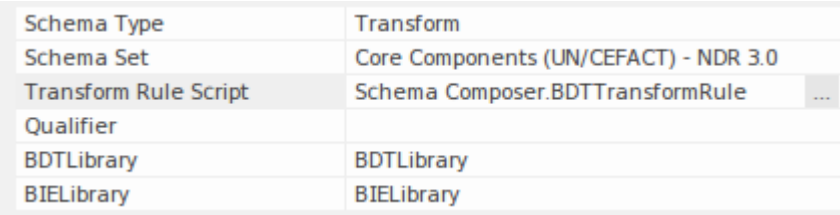
The Scripting editor in Enterprise Architect will help you write script that interacts with the Schema Composer, by providing Intelli-sense on the properties and methods of its Automation Interface.



Stereotype mapping in Model Transformation

Stereotyping forms a large part of the MDG Technology approach. Individual UML profiles for an MDG Technology define stereotypes to offer useful classifications for its elements. It is a common requirement when going from a core framework to a business model or sub-domain to reassign the stereotype. When you work with a CCTS framework the business components you generate have their stereotype automatically generated by Enterprise Architect according to a mapping defined by the CCTS specification (ACC to ABIE, for example).

When you open or create a model transform profile in the Schema Composer you can specify a script to perform this mapping for you. The script can be selected from the Properties window.



The script can be written in either JavaScript, JScript or VBScript, and only has to implement this function (described here in JavaScript notation):

function TranslateStereotype(srcStereo)

```
{
    var destStereo = srcStereo
    if (srcStereo == "BDT")
    {
        destStereo = "My_BDT"
    }
    return destStereo;
}
```

MDG Technologies - UML Profile Extensions

The Schema Composer works with MDG technologies. The standards it uses for schema generation, *other than Generic,* are only meaningful for models that adhere to that framework. However, it is quite easy to extend an existing MDG Technology. Make sure that elements authored in your business specific domain or sub-domain provide consistently named metadata or 'Tagged Values'.

The Schema Composer supports extensions to UML profiles / frameworks through its scripting integration. When a script is assigned in the Schema Composer, the transform process will invoke this script and ask it to translate keywords. These keywords are usually UML stereotypes. If a particular technology is associated with the profile, the Schema Composer will invoke this function, passing it the name of the MDG Technology.

The script can return the input name, and no mapping will take place, or it can return the name of another MDG Technology. When this occurs, the Schema Composer will again ask for the function to optionally map any UML profiles. Finally it will ask the script to translate the stereotypes from the core technology.

The result of the model transform would then be that any UML elements of the sub model will show the extended Tagged Values in addition to any core Tagged Values.

Example script that maps MDG Technology

```
function TranslateStereotype (stereo)
{
  var newStereo = stereo;
  if (stereo == "UPCC3")
  {
      newStereo = "XXX UPCC3"
  }
  return newStereo;
}
```

Example script that maps UML profile

```
function TranslateStereotype (stereo)
{
  var newStereo = stereo;
  if (stereo == "UPCC3 - BIE Library Abstract Syntax")
  {
      newStereo = "UPCC3 - BIE Library XXX Syntax"
  }
  return newStereo;
}
```

Example script that maps UML Stereotype

```
function TranslateStereotype (stereo)
{
   var newStereo = stereo;
   if (stereo == "ABIE")
   {
       newStereo = "XXX ABIE";
   }
   return newStereo;
}
```

XSD Models

XML Schema Definition (XSD), also known as XML Schema, is a World Wide Web Consortium (W3C) XML technology that is used to specify the rules to which an XML document must adhere. XSD support is critical for the development of a complete Service Oriented Architecture (SOA), and the coupling of UML 2.5 and XML provides the natural mechanism for specifying, constructing and deploying XML-based SOA artifacts within an organization.

The UML Profile for XSD specifies a set of stereotypes, Tagged Values and constraints that can be applied to the UML model in order to change specific aspects of the resulting schema. Enterprise Architect provides native support for the XSD Profile through the XML Schema page of the Diagram Toolbox. The XSD Profile supported by Enterprise Architect is an adaptation of the profile defined in the publication Modeling XML Applications with UML.

Working with the XSD Profile through Enterprise Architect, you can rapidly model, forward engineer and reverse engineer XML Schema.

You can also quickly define and generate XSD and other schema using the Enterprise Architect Schema Composer.

Modeling XSD

You can model XML schemas at two levels, using UML Class diagrams that:

- Have no XML schema-specific implementation details, to be generated directly by Enterprise Architect's Schema Generator; the generator applies a set of default mappings to convert the abstract model Package to a W3C XML Schema (XSD) file
- Are refined with XML schema-specific definitions using the 'XML Schema' pages of the Diagram Toolbox, which provides the structures of the UML profile for XSD

Model an XML Schema

Step	Action
1	In the Browser window, create the top-level project structure you need (Model and Views), and click on the appropriate View.
2	Click on the 'New Package' option in the Browser window header drop-down menu. The 'New Model Package' dialog displays.
3	In the 'Name' field type the name of the new

	Package, and select the 'Create diagram' radio button. Click on the OK button. The 'New Diagram' dialog displays.
4	In the 'Name' field type the name of the new diagram. In the 'Select From' panel select 'UML Structural', and in the 'Diagram Types' panel select 'Class'.
5	Click on the OK button. In the Browser window, double-click on the icon next to the new diagram's name; the diagram opens in the Diagram View, with the 'Class' pages displaying in the Diagram Toolbox. At this point you can either: • Create a Class diagram using the Class Toolbox icons, or • Create a tailored XML Schema diagram using the 'XML Schema' pages of the Diagram Toolbox (continue to step 6)
6	Click on  to display the 'Find Toolbox Item' dialog and specify 'XML Schema' to display the 'XML Schema' Toolbox pages.
7	Click on the 'Schema' icon from the Toolbox and drag it into the Class diagram. The 'XSD schema Properties' dialog displays.

	Complete this dialog, and click on the OK button. The 'New Diagram' dialog displays.
8	Again, in the 'Name' field type the name of the new diagram. In the 'Select From' panel select 'UML Structural', and in the 'Diagram Types' panel select 'Class'. Click on the OK button.
9	An XSDschema stereotyped Package is created in the Browser window and on the diagram, with a child Class diagram. Double-click on the Package on the diagram to open the child Class diagram, and use the constructs from the XML Schema Toolbox to model the XML Schema.

Notes

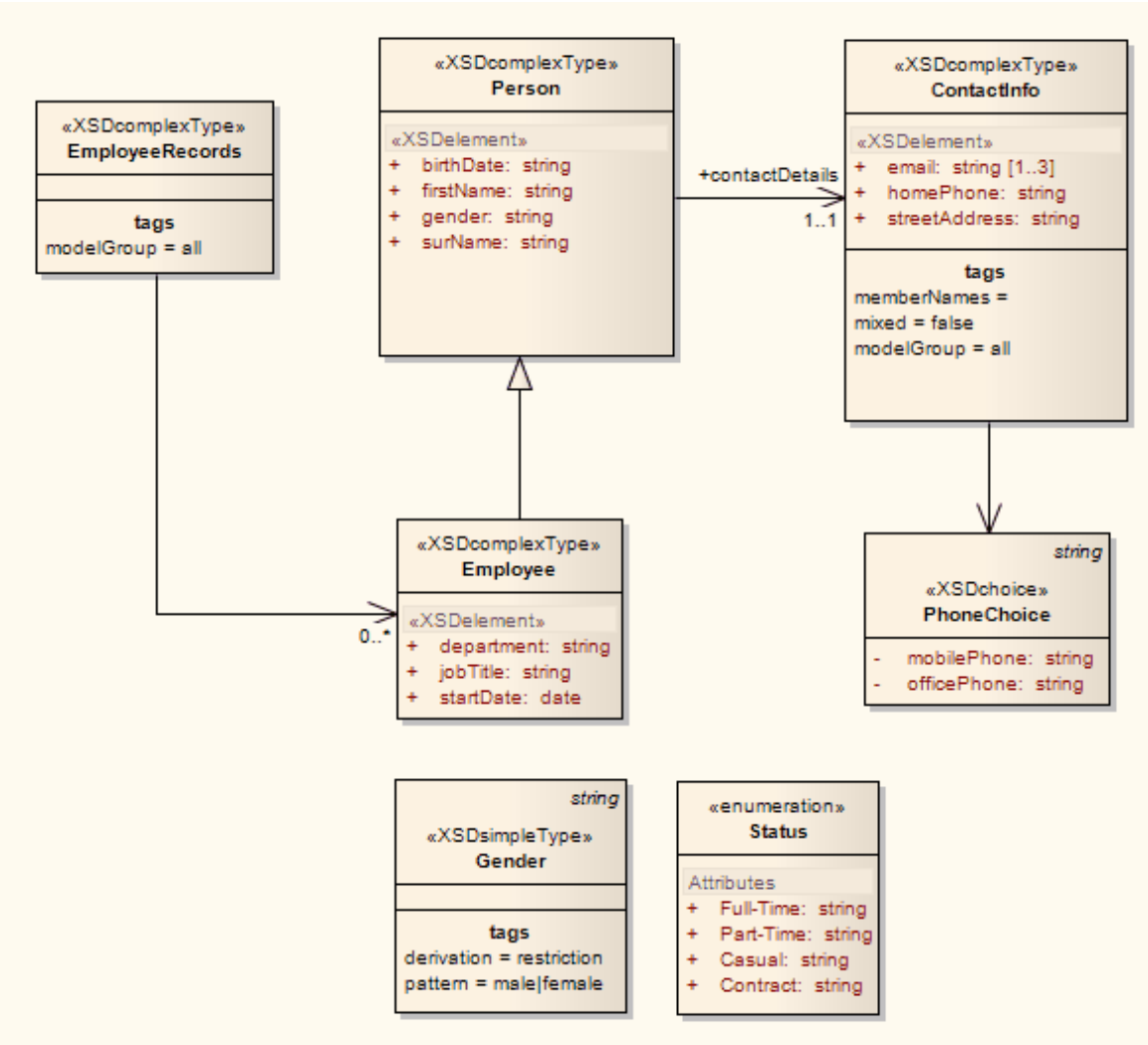
- The UML attributes of the Classes map directly to XML elements or attributes
- If you have modeled your XSD Schema as a straight Class diagram, you can define and generate schema from it using the Schema Composer
- Classes in an XML Schema model have no methods since

there is no meaningful correspondence between Class methods and XSD constructs

- Modeling Restrictions - these XML Schema constructs cannot be modeled in Enterprise Architect:
  - appinfo
  - field
  - key
  - keyref
  - notation
  - redefine
  - selector
  - substitutionGroup
  - unique

XSD Diagrams

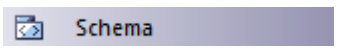
This example diagram shows a Class diagram containing XSD-specific elements created using the 'XSD Schema' pages of the Diagram Toolbox. The diagram models an employee records system.



Schema Package

An «XSDschema» stereotyped Package acts as a container for the XSD constructs, from which XML Schema can be generated. All Classes in the Package are defined within one schema; the Schema element provides the default schema-wide settings. You can create an «XSDschema» Package by dragging the Schema icon from the XML Schema Toolbox and dropping it directly onto a diagram.

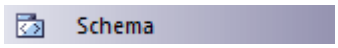
Toolbox Icon



Access

To display the XSD schema 'Properties' dialog for the selected «XSDschema» stereotyped Package, use one of the methods outlined here:

Ribbon	Design > Package > Manage > Properties
Context Menu	Right-click on «XSDschema» stereotyped Package | Properties
Other	In Browser window, double-click on

	existing «XSDschema» stereotyped Package, or Drag  icon from toolbox onto diagram (this creates a new «XSDschema» stereotyped Package)

Define Properties

Field/Button	Action
Schema Name	If you do not want to use the default name of the schema Package, overtype it with another name.
Target Namespace	(Optional) Type in the target namespace for this Schema Package.
Prefix	(Optional) Type in the abbreviated text to represent the Target Namespace.
Default Namespace	(Optional) Type in the default namespace for all non-prefixed XSDelements and XSDattributes.
Schema File	Type in or browse for (click on ...) the file path where the XML Schema file for

	this Package is to be generated.
XMLNS	Identify the additional namespace or namespace-prefix pairs used in this Schema Package. To add a namespace or namespace-prefix pair, click on the New button; to edit an existing entry, double-click on it. In either case, the 'Namespace Details' dialog displays. • Prefix - Type in the abbreviated text to represent the Namespace • Namespace - Type in the name of the Namespace • OK - Click on this button to save the new information and close the 'Namespace Details' dialog • Cancel - Click on this button to discard the new information and close the 'Namespace Details' dialog • Help - Click on this button to display this Help topic To remove an entry from the list, click on it and click on the Delete button.
OK	Click on this button to save the schema data entered and close the XSD schema 'Properties' dialog.

Cancel	Click on this button to discard the schema data entered and close the XSD schema 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing Schema Package information. Click on the button to open the UML element 'Properties' dialog for the Schema element.

Notes

- The default schema-wide settings are defined by Tagged Values, which you can review on the 'Tags' tab of the schema element 'Properties' dialog, or the Properties window for the element; you can edit the schema-wide settings if you need to, or provide element-specific overrides in the properties and Tagged Values of the individual XSD construct elements

Global Element

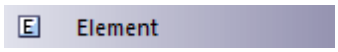
An «XSDtopLevelElement» stereotyped Class acts as a XSD global element. You can create it by dragging the 'Element' icon from the XML Schema Toolbox and dropping it directly onto a diagram.

Toolbox Icon

Access

To display the 'XSD element Properties' dialog for the selected «XSDtopLevelElement» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «XSDtopLevelElement» stereotyped Class | Properties
Keyboard Shortcuts	Alt+Enter
Other	In Browser window, double-click on

	«XSDtopLevelElement» stereotyped Class, or Drag  icon from toolbox onto diagram (this creates a new «XSDtopLevelElement» stereotyped Class)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the global element, overtype it with another name.
Type	Either: • Type the name of a data type, or • Click on the drop-down arrow and select an XSD built-in dataType from the list, or • Click on the  button and browse for an existing XSD classifier element, or • Select one of these two checkboxes
Nested	Select this checkbox to create an

complexType	XSDcomplexType as a child of this global element.
Nested simpleType	Select this checkbox to create an XSDsimpleType as a child of this global element.
Value	(Optional) If you have entered an XSD built-in data type in the 'Type' field, type in a value.
Default	Select this radio button to set the Value as a default value.
Fixed	Select this radio button to set the Value as a fixed value.
Annotation	(Optional) Type in any notes you need for this element.
OK	Click on this button to save the element data entered and close the XSD element 'Properties' dialog.
Cancel	Click on this button to discard the element data entered and close the XSD element 'Properties' dialog.
Help	Click on this button to display this Help

	topic.
UML	This button is displayed when you are editing existing XSD element information. Click on the button to open the UML element 'Properties' dialog for the global element.

Notes

- The fields 'Type', 'Nested complexType' and 'Nested simpleType' are mutually exclusive; selecting one disables the others
- The fields 'Nested complexType' and 'Nested simpleType' are available in the dialog only when creating a new global element (and not when editing the global element)
- A Global element:
  - Cannot contain any UML attributes
  - Cannot be the source of an Association connector
  - Can be the target of an Association connector from a Complex Type Class or Group Class element
  - Cannot be the target of a Generalization connector
  - Can be the source of one Generalization connector to a Complex Type Class or Simple Type Class

Local Element

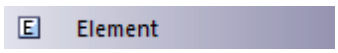
A Local element is an «XSDelement» stereotyped UML attribute that acts as a local XSD element. You can create it by dragging the 'Element' icon from the XML Schema Toolbox and dropping it onto an «XSDcomplexType» or «XSDgroup» stereotyped Class.

Toolbox Icon

Access

To display the 'XSD element Properties' dialog for the selected «XSDelement» stereotyped UML attribute, use one of the methods outlined here.

Ribbon	With a specific «XSDelement» stereotyped attribute selected in a diagram: Design > Element > Features > Attributes
Context Menu	With a specific «XSDelement» stereotyped attribute selected in a diagram :

	Right-click on attribute | View Properties
Keyboard Shortcuts	With a specific «XSDelement» stereotyped attribute selected in a diagram : F9
Other	Double-click on «XSDelement» stereotyped attribute, or Drag  icon onto «XSDcomplexType» or «XSDgroup» stereotyped Class, (this creates a new «XSDelement» within the Class)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the local element, overtype it with another name.
Type	Either: • Type the name of a data type, or • Click on the drop-down arrow and

	select an XSD built-in dataType from the list, or • Click on the ... button and browse for an existing XSDcomplexType or XSDsimpleType element as a classifier
Reference	(Optional) Specify whether to use the ref attribute (instead of the type attribute) to refer to the XSDcomplexType or XSDsimpleType element you selected in the 'Type' field, in the generated XSD.
Value	(Optional) If you have entered an XSD built-in data type in the 'Type' field, type in a value.
Default	Select this radio button to set the Value as a default value.
Fixed	Select this radio button to set the Value as a fixed value.
MinOccurs	(Optional) Type the minimum number of times this element must occur in the Class. Type '0' to indicate that the element is optional. The default value is '1'.

MaxOccurs	(Optional) Type the maximum number of times this element can occur in the Class. Type unbounded to indicate that there is no limit to the number of times the element can occur. The default value is 1.
Form	(Optional) Click on the drop-down arrow and select whether or not to qualify the element: • qualified - Use the Prefix defined in the Schema Package to qualify this element • unqualified - Do not qualify this element
Annotation	(Optional) Type in any notes you need for this local element.
OK	Click on this button to save the element data entered and close the XSD element 'Properties' dialog.
Cancel	Click on this button to discard the element data entered and close the XSD element 'Properties' dialog.
Help	Click on this button to display this Help

	topic.
UML	This button is displayed when you are editing existing XSD element information. Click on the button to open the attribute properties for the local element.

Notes

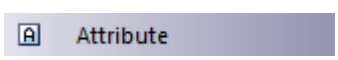
- Only «Complex Type», «Group» and «Model Group» stereotyped elements can have this UML Attribute

Global Attribute

A Global Attribute is an «XSDtopLevelAttribute» stereotyped Class. You can create it by dragging the 'Attribute' icon from the XML Schema Toolbox and dropping it directly onto a diagram.

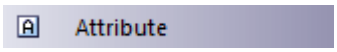
Access

To display the 'XSD attribute Properties' dialog for the selected «XSDtopLevelAttribute» stereotyped element, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «XSDtopLevelAttribute» stereotyped element| Properties
Keyboard Shortcuts	Alt+Enter
Other	Double-click on «XSDtopLevelAttribute» stereotyped Class, or Drag  icon from toolbox and drop directly onto the diagram (this creates a new «XSDtopLevelAttribute»

	stereotyped Class)

Toolbox Icon



Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the global attribute, overtype it with another name.
Type	Either: • Type the name of a data type, or • Click on the drop-down arrow and select an XSD built-in dataType from the list, or • Click on the ... button and browse for an existing XSDsimpleType element as a classifier Alternatively, select the 'Nested simpleType' checkbox.

Nested simpleType	Select this checkbox to create an XSDsimpleType element as a child of this global attribute element.
Value	(Optional) If you have selected an XSD built-in dataType in the 'Type' field, type in a value.
Default	Select this radio button to set the 'Value' field as a default value.
Fixed	Select this radio button to set the 'Value' field as a fixed value.
Form	(Optional) Click on the drop-down arrow and select: • qualified to use any Prefix supplied on the Schema Package to qualify this attribute, or • unqualified to show no qualifying prefix on the attribute name
Annotation	(Optional) Type in any notes you need for this attribute.
OK	Click on this button to save the attribute data entered and close the XSD attribute 'Properties' dialog.

Cancel	Click on this button to discard the attribute data entered and close the XSD attribute 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing XSD element information. Click on the button to open the UML element 'Properties' dialog for the global attribute Class.

Notes

- The field 'Nested simpleType' is available in the dialog only when creating a new global attribute (and not when editing the global attribute)
- The fields 'Type' and 'Nested simpleType' are mutually exclusive; selecting one disables the other
- A Global attribute:
  - Cannot contain any UML attributes
  - Cannot be the source of an Association connector
  - Can be the target of an Association connector from a

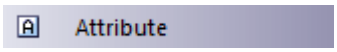
Complex Type Class

- Cannot be the target of a Generalization connector
- Can be the source of one Generalization connector to a Simple Type Class

Local Attribute

A local attribute is an «XSDattribute» stereotyped UML attribute. You can create it by dragging the 'Attribute' icon from the XML Schema Toolbox and dropping it onto an «XSDcomplexType» or «XSDattributeGroup» stereotyped Class.

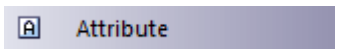
Toolbox Icon



Access

To display the 'XSD attribute Properties' dialog for the selected «XSDattribute» stereotyped UML attribute, use one of the methods outlined here.

Ribbon	With a specific «XSDattribute» stereotyped UML attribute selected on a diagram: Design > Element > Features > Attributes
Context Menu	With a specific «XSDattribute» stereotyped UML attribute selected on a diagram:

	Right-click on attribute | View Properties
Keyboard Shortcuts	With a specific «XSDattribute» stereotyped UML attribute selected on a diagram: F9
Other	Double-click on the «XSDattribute» stereotyped UML attribute, or Drag  icon from the Toolbox and drop it onto an «XSDcomplexType» or «XSDattributeGroup» stereotyped Class (this creates a new «XSDattribute» stereotyped UML attribute)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the local attribute, overtype it with another name.
Type	Either:

	• Type the name of a data type, or • Click on the drop-down arrow and select an XSD built-in dataType from the list, or • Click on the ... button and browse for an existing XSDsimpleType element as a classifier
Reference	(Optional) Specify whether to use the ref attribute (instead of the type attribute) to refer to the XSDsimpleType element you selected in the 'Type' field, in the generated XSD.
Value	(Optional) If you have entered an XSD built-in data type in the 'Type' field, type in a value.
Default	Select this radio button to set the Value as a default value.
Fixed	Select this radio button to set the Value as a fixed value.
Form	(Optional) Click on the drop-down arrow and select whether or not to qualify the attribute: • qualified - Use the Prefix defined in the Schema Package to qualify this

	attribute • unqualified - Do not qualify this attribute
Annotation	(Optional) Type in any notes you need for this local attribute.
OK	Click on this button to save the attribute data entered and close the XSD attribute 'Properties' dialog.
Cancel	Click on this button to discard the element data entered and close the XSD attribute 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing XSD attribute information. Click on the button to open the attribute properties for the local attribute.

Notes

- Only Complex Types and Attribute Groups can have this UML attribute

Attribute Group

An Attribute Group Class is used to group a set of «XSDattribute» stereotyped UML attributes and Simple Type Classes that can be referenced from an «XSDcomplexType» stereotyped Class. You can create it by dragging the 'Attribute Group' icon from the XML Schema Toolbox and dropping it directly onto a diagram.

Toolbox Icon

Access

To display the 'XSD Attribute Group Properties' dialog for the selected «XSDattributeGroup» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «XSDattributeGroup» stereotyped Class | Properties
Keyboard Shortcuts	Alt+Enter

Other	Double-click on «XSDattributeGroup» stereotyped Class, or Drag  icon from toolbox onto diagram (this creates a new «XSDattributeGroup» stereotyped Class)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the Attribute Group, overtype it with another name.
Annotation	(Optional) Type in any notes you need for this Attribute Group.
OK	Click on this button to save the attribute group data entered and close the XSD Attribute Group 'Properties' dialog.
Cancel	Click on this button to discard the attribute group data entered and close the XSD Attribute Group 'Properties' dialog.

Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing XSD attribute group information. Click on the button to open the UML element 'Properties' dialog for the attribute group.

Notes

- An Attribute Group element:
  - Cannot be the child of any other XSD Class
  - Can contain only «XSDattribute» stereotyped UML attributes and Simple Type Classes
  - Can be the source of an Association connector to another Attribute Group
  - Can be the target of an Association connector from a Complex Type Class
  - Cannot be the source or target of a Generalization connector

Complex Type

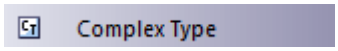
An «XSDcomplexType» stereotype is applied to a generic UML Class, to tailor the generation of a complexType definition in the Schema. You can create an «XSDcomplexType» stereotyped Class by dragging the Complex Type icon from the XML Schema Toolbox and dropping it directly onto a diagram.

Toolbox Icon

Access

To display the 'XSD complexType Properties' dialog for the selected «XSDcomplexType» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «XSDcomplexType» stereotyped Class | Properties
Keyboard Shortcuts	Alt+Enter

Other	Double-click on «XSDcomplexType» stereotyped Class, or Drag  icon from toolbox onto diagram (this creates a new «XSDcomplexType» stereotyped Class)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the complexType Class, overtype it with another name.
Model Group	Click on the down-arrow and select the option that defines how the child elements of this complexType should occur in the Schema. • 'sequence' - the child elements must occur in the specified order • 'choice' - only one of the child elements can occur • 'all' - the child elements can occur in any order

MinOccurs	(Optional) Type the minimum number of times this element must occur in the Class. Type 0 to indicate that the element is optional. The default value is 1.
MaxOccurs	(Optional) Type the maximum number of times this element can occur in the Class. Type unbounded to indicate that there is no limit to the number of times the element can occur. The default value is 1.
Annotation	(Optional) Type any notes you need for this element.
Abstract	(Optional) Select this checkbox to use this complexType in an instance XML file.
Mixed	(Optional) Select this checkbox to allow character data to display among the child elements.
OK	Click on this button to save the complexType data entered and close the XSD complexType 'Properties' dialog.

Cancel	Click on this button to discard the complexType data entered and close the XSD complexType 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing XSD complexType information. Click on the button to open the UML element 'Properties' dialog for the complexType Class.

Notes

- A complexType can:
  - Contain both XSDelement and XSDattribute stereotyped UML attributes
  - Contain other complexTypes as child elements
  - Be a child of a Global Element
  - Be the source of Association connectors to other complexTypes, Simple Types, Attribute Groups, Groups and Model Groups
  - Be the source of a maximum of one Generalization connector to either another complexType or a Simple

Type Class

Simple Type

An «XSDsimpleType» stereotype is applied to a generic UML Class, to tailor the generation of a simpleType definition in the Schema. You can create an «XSDsimpleType» Class by dragging the Simple Type icon from the XML Schema Toolbox and dropping it directly onto a diagram.

Toolbox Icon

Access

To display the 'XSD simpleType Properties' dialog for the selected «XSDsimpleType» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «XSDsimpleType» stereotyped Class | Properties
Keyboard Shortcuts	Alt+Enter

Other	Double-click on «XSDsimpleType» stereotyped Class, or Drag  icon from toolbox onto diagram (this creates a new «XSDsimpleType» stereotyped Class)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the simpleType element, overtype it with another name.
Type	Either: • Type the name of a data type, or • Click on the drop-down arrow and select an XSD built-in dataType from the list, or • Click on the ... button and browse for an existing «XSDsimpleType» element as a classifier
Restriction	Select this radio button to restrict the value of this simpleType to that of the

	selected Type. The various restrictions (facets) on the simpleType are available as Tagged Values on this Class.
List	Select this radio button to specify this simpleType as a list of values of the selected Type.
Annotation	(Optional) Type any notes you need for this element.
OK	Click on this button to save the simpleType data entered and close the XSD simpleType 'Properties' dialog.
Cancel	Click on this button to discard the simpleType data entered and close the XSD simpleType 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing XSD simpleType information. Click on the button to open the UML element 'Properties' dialog for the simpleType Class.

Notes

- A simpleType:
- Cannot contain any «XSDelement» or «XSDattribute» stereotyped UML attributes
- Cannot contain any child Classes
- Cannot be the source of an Association connector
- Can be the target of a Generalization connector
- Can have at the most one Generalization connector to another simpleType Class

Group

The Group Class is used to group a set of «XSDelement» stereotyped UML attributes, Complex Type Classes and Simple Type Classes that can be referenced from an «XSDcomplexType» Class. You can create this type of element by dragging the Group icon from the XML Schema Toolbox and dropping it directly onto a diagram.

Toolbox Icon

Access

To display the 'XSD group Properties' dialog for the selected «XSDgroup» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «XSDgroup» stereotyped Class | Properties
Keyboard Shortcuts	Alt+Enter

Other	Double-click on «XSDgroup» stereotyped Class, or Drag  icon from toolbox onto diagram (this creates a new «XSDgroup» stereotyped Class)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the Group element, overtype it with another name.
Model Group	Click on the drop-down arrow and select the value that defines how the child elements of this group should occur in the Complex Type Class: • sequence - to specify that the child elements must occur in the specified order • choice - to specify that only one of the child elements can occur • all - to specify that the child elements

	can occur in any order
Annotation	(Optional) Type any notes you need for this element.
OK	Click on this button to save the Group data entered and close the XSD group 'Properties' dialog.
Cancel	Click on this button to discard the Group data entered and close the XSD group 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing XSD Group information. Click on the button to open the UML element 'Properties' dialog for the Group Class.

Notes

- A Group element can:
- Contain only «XSDelement» stereotyped UML attributes

- Contain Complex Types and Simple Types as child elements
- Be the source of Association connectors to other Complex Types, Simple Types and Groups
- Be the target of an Association connector from a Complex Type element
- Not be the source or target of a Generalization connector

Any

An «XSDany» stereotyped Class allows a Complex Type Class to contain elements that are not specified in the Schema Package. You can create it by dragging the Any icon from the XML Schema Toolbox and dropping it directly onto a diagram.

Toolbox Icon

Access

To display the 'XSD any Properties' dialog for the selected «XSDany» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «XSDany» stereotyped Class | Properties
Keyboard Shortcuts	Alt+Enter

Other	Double-click on «XSDany» stereotyped Class, or Drag  icon from toolbox onto diagram (this creates a new «XSDany» stereotyped Class)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the Any element, overtype it with another name.
Namespace	(Optional) Type the namespace to contain the elements that can be used in the Complex Type.
ProcessContents	(Optional) Click on the drop-down arrow and select the value that defines how the XML Parser should validate these elements: • lax - to attempt to validate the elements against their Schema; no error is flagged when the Schema cannot be obtained

	• skip - to skip validating the elements • strict - to validate the elements against their Schema and flag an error if the Schema is unobtainable
Annotation	(Optional) Type any notes you need for this element.
OK	Click on this button to save the 'Any' element data entered and close the XSD Any 'Properties' dialog.
Cancel	Click on this button to discard the Any element data entered and close the XSD group 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing «XSDany» element information. Click on the button to open the UML element 'Properties' dialog for the Any Class.

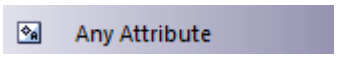
Notes

- An Any Class:
- Cannot contain any UML Attributes or child XSD Classes
- Cannot be the child of any XSD Class
- Cannot be the target of a Generalization connector
- Cannot be the source of an Association or Generalization connector
- Can be the target of Association connectors from Complex Types, Groups and Model Groups
- Must be the target of at least one incoming Association connector

Any Attribute

The «XSDany» stereotyped UML attribute allows a Complex Type element or an Attribute Group element to contain attributes that are not specified in the Schema Package. You can create it by dragging the 'Any Attribute' icon from the XML Schema Toolbox and dropping it onto an «XSDcomplexType» or «XSDattributeGroup» stereotyped Class.

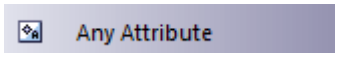
Toolbox Icon



Access

To display the 'XSD anyAttribute Properties' dialog for the selected «XSDany» stereotyped UML attribute, use one of the methods outlined here.

Ribbon	With a specific «XSDany» stereotyped UML attribute selected on a diagram: Design > Element > Features > Attributes
Context Menu	With a specific «XSDany» stereotyped UML attribute selected on a diagram:

	Right-click on attribute | View Properties
Keyboard Shortcuts	With a specific «XSDany» stereotyped UML attribute selected on a diagram: F9
Other	Double-click on the «XSDany» stereotyped UML attribute, or Drag  icon from the Toolbox and drop it onto an «XSDcomplexType» or «XSDattributeGroup» stereotyped Class (this creates a new «XSDany» stereotyped UML attribute)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the attribute, overtype it with another name.
Namespace	(Optional) Type the namespace to contain the attributes that can be used in the Complex Type or Attribute Group

	elements.
ProcessContents	(Optional) Click on the drop-down arrow and select the value that defines how the XML Parser should validate these attributes: • lax - to attempt to validate the attributes against their Schema; no error is flagged when the Schema cannot be obtained • skip - to skip validating the attributes • strict - to validate the attributes against their Schema and flag an error if the Schema is unobtainable
Annotation	(Optional) Type any notes you need for this attribute.
OK	Click on this button to save the attribute data entered and close the XSD anyAttribute 'Properties' dialog.
Cancel	Click on this button to discard the attribute data entered and close the XSD anyAttribute 'Properties' dialog.
Help	Click on this button to display this Help topic.

UML	This button is displayed when you are editing existing «XSDany» attribute information. Click on the button to open the attribute properties for the «XSDany» attribute.

Notes

- Only Complex Type and Attribute Group elements can have this UML attribute

Union

A Union Class is a Simple Type element that defines a collection of Simple Types. You can create it by dragging the Union icon from the XML Schema Toolbox and dropping it directly on a diagram.

Toolbox Icon

Access

To display the 'XSD union Properties' dialog for the selected «XSDunion» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «XSDunion» stereotyped Class | Properties
Keyboard Shortcuts	Alt+Enter
Other	Double-click on «XSDunion»

	stereotyped Class, or Drag  icon from toolbox onto diagram (this creates a new «XSDunion» stereotyped Class)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the Union, overtype it with another name.
Member Types	Click on the ... button to display the 'XSD Union Members' dialog, and select built-in XSD datatypes and Simple Type elements to be members of the collection. • Choose - Instead of typing or selecting values in the 'Class Name' field, click on this button to display the 'Select Classifier' browser and locate and select a Simple Type element; click on the OK button to close the browser and immediately add the selected element to the 'Type Details' list This option is generally used to specify

	objects that are in the same Package as the Union element, but you can select objects in any other Package also • Add - Click on this button to add the data type or element specified in the 'Class Name' field to the 'Type Details' list • Accept classifier even if not in model - Select this checkbox to include elements or data types that have been named but that are not present in the same model Package as the Union element • Type Details - Review the list of selected elements or data types; if you intend to remove an object from the list, highlight it and click on the Delete Selected button • Delete Selected - Click on this button to remove the currently-selected Classifier from the 'Type Details' list • Close - Click on this button to close the 'XSD Union Members' dialog and to list the selected elements and data types in the 'Member Types' field
Annotation	(Optional) Type any notes you need for this element.

OK	Click on this button to save the attribute data entered and close the XSD union 'Properties' dialog.
Cancel	Click on this button to discard the attribute data entered and close the XSD union 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing «XSDunion» element information. Click on the button to open the UML element 'Properties' dialog for the «XSDunion» element.

Notes

- When you click on the Close button on the XSD union 'Properties' dialog, a Generalization connector is added to the diagram from the XSD Union element to each of the member elements on the same diagram; any elements not on the same diagram are listed in the top right corner of the XSD Union element

- If the Member Types that are not on the same diagram as the Union element are not listed, select 'Start > Appearance > Preferences > Preferences > Diagram > Behavior' and select the 'Show Hidden Parents' checkbox
- A Union element:
  - Cannot contain any child Classes
  - Cannot contain any «XSDelement» or «XSDattribute» stereotyped UML attributes
  - Cannot be the source of an Association connector
  - Can be the target of an Association connector from a Complex Type element
  - Can be the target of a Generalization connector from a Simple Type element

Model Group

You can create an «XSDsequence», «XSDchoice» or «XSDall» stereotyped Class by dragging the Model Group icon from the XML Schema Toolbox and dropping it directly onto a diagram.

An «XSDsequence» model group (the default model group type) is a container for the attributes and associations owned by the Class. The model group is in turn added to the model groups of the Class's respective owners. Tagged Values specified by owners of the Class persist through to the child elements of the model group; if memberNames are unqualified for a complexType, so are the children of this model group when added to that complexType.

Toolbox Icon

Access

To display the 'XSD Model Group Properties' dialog for the selected «XSDsequence», «XSDchoice» or «XSDall» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties

Context Menu	Right-click on «XSDsequence», «XSDchoice» or «XSDall» stereotyped Class | Properties
Keyboard Shortcuts	Alt+Enter
Other	• Double-click on «XSDsequence», «XSDchoice» or «XSDall» stereotyped Class, or • Drag the  icon from the Toolbox onto the diagram (this creates a new Model Group element; you can choose from the «XSDsequence», «XSDchoice» or «XSDall» stereotypes, of which «XSDsequence» is the default)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the Model Group, overtype it with another name.

Model Group	Click on the drop-down arrow and select the value that defines how the child elements of this group should occur in the Complex Type Class: • sequence - to specify that the child elements must occur in the specified order; creates an «XSDsequence» stereotyped Class • choice - to specify that only one of the child elements can occur; creates an «XSDchoice» stereotyped Class • all - to specify that the child elements can occur in any order; creates an «XSDall» stereotyped Class
MinOccurs	(Optional) Type the minimum number of times this element must occur in the Class. Type 0 to indicate that the element is optional. The default value is 1.
MaxOccurs	(Optional) Type the maximum number of times this element can occur in the Class. Type unbounded to indicate that there is no limit to the number of times the element can occur.

	The default value is 1.
Annotation	(Optional) Type any notes you need for this element.
OK	Click on this button to save the Model Group data entered and close the XSD element 'Properties' dialog.
Cancel	Click on this button to discard the Model Group data entered and close the XSD element 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing Model Group element information. Click on the button to open the UML element 'Properties' dialog for the Model Group Class.

Notes

- A Model Group:

- Can contain only «XSDelement» stereotyped UML attributes
- Can contain Complex Types and Simple Types as child elements
- Can be the source of Association connectors to Complex Type, Simple Type, Group and Model Group elements
- Must be the target of least one incoming Association connector from a Complex Type
- Cannot be the source or target of a Generalization connector

Enumeration

An Enumeration defines a list of acceptable values for the Class. You can create an Enumeration element by dragging the Enum icon from the XML Schema Toolbox and dropping it directly onto a diagram.

Toolbox Icon

Access

To display the 'XSD enumeration Properties' dialog for the selected «enumeration» stereotyped element, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «enumeration» stereotyped element | Properties
Keyboard Shortcuts	Alt+Enter
Other	Double-click on «enumeration»

	stereotyped element, or Drag  icon from toolbox and drop directly onto the diagram (this creates a new «enumeration» stereotyped element)

Define Properties

Field/Button	Action
Name	If you do not want to use the default name of the Enumeration, overtype it with another name.
Type	Either: • Type the name of a data type, or • Click on the drop-down arrow and select an XSD built-in dataType from the list, or • Click on the  button and browse for an existing XSDsimpleType element
Values	Type each of the values, separated by commas, for the selected Type. These values are listed on the element as attributes.

Annotation	(Optional) Type any notes you need for this element.
OK	Click on this button to save the Enumeration element data entered and close the XSD enumeration 'Properties' dialog.
Cancel	Click on this button to discard the Enumeration element data entered and close the XSD enumeration 'Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing Enumeration element information. Click on the button to open the UML element 'Properties' dialog for the Enumeration Class.

Notes

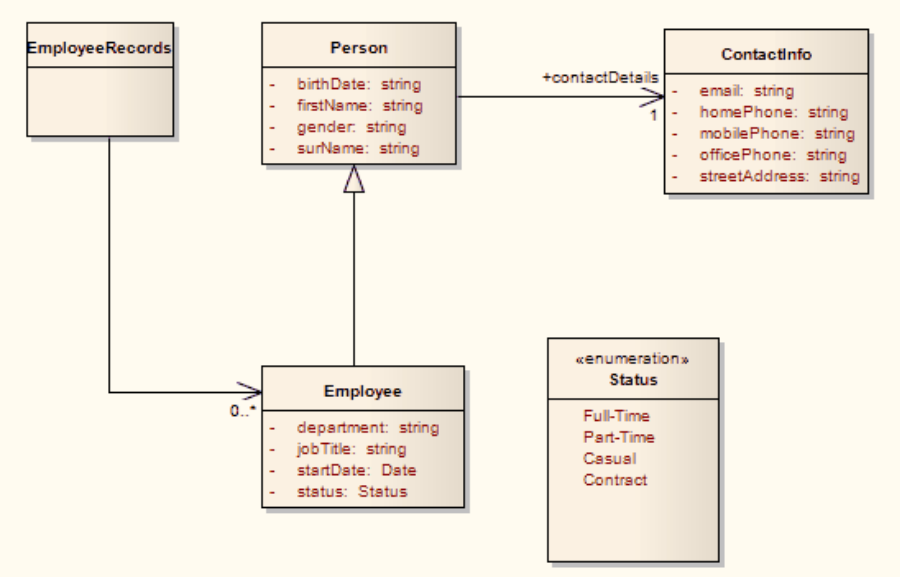
- An Enumeration:
- Cannot contain any «XSDelement» or «XSDattribute» stereotyped UML attributes
- Cannot contain any child Classes
- Cannot be the source of an Association connector
- Can be the target of a Generalization connector
- Can have at most one Generalization connector to a Simple Type Class

XML from Abstract Class Models

You can model XML schemas using only simple, abstract Class models. This makes it possible for an architect, for example, to start working at a higher level of abstraction without concern for the implementation details of a Schema. Whilst such an abstract model can subsequently be refined using the 'XML Schema' pages of the Toolbox, it can be also be generated directly by Enterprise Architect's Schema Generator, in which case the Schema Generator applies a set of default mappings to convert the abstract model to an XSD file.

Example

Structure	Detail
Diagram	This is a simple Class element version of the earlier Employee Details example model. It does not use XSD-specific stereotypes or Tagged Values.



Schema

This schema fragment can be generated from the example model:

```
<?xml version="1.0"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema">
<xs:simpleType name="Status">
<xs:restriction base="xs:string">
<xs:enumeration value="Full-Time"/>
<xs:enumeration value="Part-Time"/>
<xs:enumeration value="Casual"/>
<xs:enumeration value="Contract"/>
</xs:restriction>
</xs:simpleType>
<xs:element name="Person" type="Person"/>
```

```
<xs:complexType name="Person">
<xs:sequence>
<xs:element name="firstName"
type="xs:string"/>
<xs:element name="surName"
type="xs:string"/>
<xs:element name="birthDate"
type="xs:string"/>
<xs:element name="gender"
type="xs:string"/>
<xs:element name="contactDetails"
type="ContactInfo"/>
</xs:sequence>
</xs:complexType>
<xs:element name="Employee"
type="Employee"/>
<xs:complexType name="Employee">
<xs:complexContent>
<xs:extension base="Person">
<xs:sequence>
<xs:element name="status"
type="Status"/>
<xs:element name="jobTitle"
type="xs:string"/>
<xs:element name="startDate"
type="xs:date"/>
<xs:element name="department"
```

	type="xs:string"/> </xs:sequence> </xs:extension> </xs:complexContent> </xs:complexType> <xs:element name="EmployeeRecords" type="EmployeeRecords"/> <xs:complexType name="EmployeeRecords"> <xs:sequence> <xs:element name="Employee" type="Employee" minOccurs="0" maxOccurs="unbounded"/> </xs:sequence> </xs:complexType> <xs:element name="ContactInfo" type="ContactInfo"/> <xs:complexType name="ContactInfo"> <xs:sequence> <xs:element name="homePhone" type="xs:string"/> <xs:element name="mobilePhone" type="xs:string"/> <xs:element name="officePhone" type="xs:string"/> <xs:element name="email" type="xs:string"/>

	<xs:element name="streetAddress" type="xs:string"/> </xs:sequence> </xs:complexType> </xs:schema>

Default UML to XSD Mappings

When you are defining simple schemas using abstract Class models, the Enterprise Architect Schema Generator translates the UML information to XSD using a default mapping of UML to XSD constructs. These defaults are also used by the Schema Generator to generate unstereotyped elements in an abstract model.

When you model XML Schema using the 'XML Schema' pages of the Diagram Toolbox, the stereotypes and Tagged Values of the Toolbox elements override the default mappings.

Constructs

UML Construct	Default XSD Production Rules
Package	A Schema element is generated for the target Package. If the target Package includes Classes from another Package, which has the Tagged Values targetNamespace and targetNamespacePrefix set, these are included as attributes of the Schema element. In addition, an import or include element

	is created for each referenced Package: • An include element is used if the external Package shares the same targetNamespace Tagged Value as the target Package • An import element is used where the targetNamespaces differ
Class	A root-level element declaration and complexType definition are generated. The element name and type are the same as the Class name. An XSD sequence Model Group is also generated, to contain UML attributes generated as elements.
Attribute	An element is declared for each Class attribute. The element name is set to that of the UML attribute name. This is prefixed with the Class name to make the element unique. The minOccurs and maxOccurs attributes are set to reflect the attribute cardinality. If the attribute refers to another Class, the element declaration is followed by a complexType definition, which contains a reference to the appropriate complexType.
Association	An element is declared for each

	Association owned by a Class. The element name is set to that of the Association role. The minOccurs and maxOccurs attributes reflect the cardinality of the Association.
Generalization (Inheritance)	For single inheritances, an extension element is generated with the base attribute set to the base Class name. The UML attributes of the child Class are then appended to an XSDall Model Group within the extension element.
Enumeration	A simpleType element is declared for the Enumeration with the name attribute set to the Enumeration name. A Restriction element is generated with base set to string. Each of the Enumeration attributes is appended to the Restriction element as XSD Enumeration elements with value set to the UML attribute name. Any type specification for the UML attributes is ignored by the schema generator.

Notes

- If left unspecified, the minOccurs and maxOccurs

attributes default to 1

- If the direction of the Association is unspecified, the owner is assumed to be the source

Generate XSD

When you have developed your XML Schema model, either as an abstract Class model or a tailored XSD Class model, you can forward-engineer it into W3C XML Schema (XSD) files using the Generate XML Schema feature. As an XML Schema corresponds to a UML Package in Enterprise Architect, XML Schema generation is a Package-level operation.

You define the location of the file into which the XML Schema is to be generated, in the Schema Package element in your model.

Access

Ribbon	Develop > Schema Modeling > Export XSD

Generate Schema files

Option	Action
Encoding	Either:

	• Click on the drop-down arrow and select the XML encoding scheme to use, or • Click on the Default button to apply the default encoding scheme (UTF-8)
Generate global element for all global ComplexTypes ('Garden of Eden' style)	Selected by default to generate Schema in the Garden of Eden style, containing a global element. Clear the checkbox if you want to omit the global element.
Generate XSD for Referenced Packages	Select the checkbox to generate Schema for Packages that are referenced by any of the Packages selected on this dialog.
Prompt when missing Filename	Select the checkbox to prompt, during Schema generation, for a filename for a referenced Package if the path into which to generate the Schema file is missing. This option is not available if the 'Generate XSD for Referenced Packages' option is not selected.
Use	Select the checkbox to use a relative-path

relative-path to reference XSDs (if 'schemaLocation' tag is empty)	in the XSD import (or XSD include) statement when referencing external Packages, provided that the schemaLocation tag is empty on the referenced Packages. You set the 'Schema File' field on the 'XSD Schema Properties' dialog (the element 'Properties' dialog for a Schema element) for the referenced and referencing XSDschema stereotyped Packages, so that the relative-path is correctly determined.
Generate XSD for Child packages	Select the checkbox to generate schema for child Packages of the selected Package, and then select either: • Include all packages - to list all child Packages under the parent Package in the list box, or • Include <XSDschema> packages - to list only those Packages that have the stereotype «XSDschema» The list-box shows, for each Package, the Package name and the file path into which the schema file can be generated (if set). To change the file path for a Package, double-click on the entry in the list-box and type in or browse for the new file

	path in the prompt field. If the Package has a filepath already set, its checkbox is selected by default, to generate an XSD schema; if you do not want to generate an XSD schema from that Package, you can deselect the checkbox. If you select the checkbox against a Package that does not have a filepath set, the prompt automatically displays for the filepath.
Generate	Click on this button to generate the Schema for each of the Packages selected in the list-box.
Close	Click on this button to close the dialog, without saving your option selections.
View Schema	Click on this button to view the generated Schema for a Package highlighted in the list-box.
Progress	Check the progress of Schema generation.

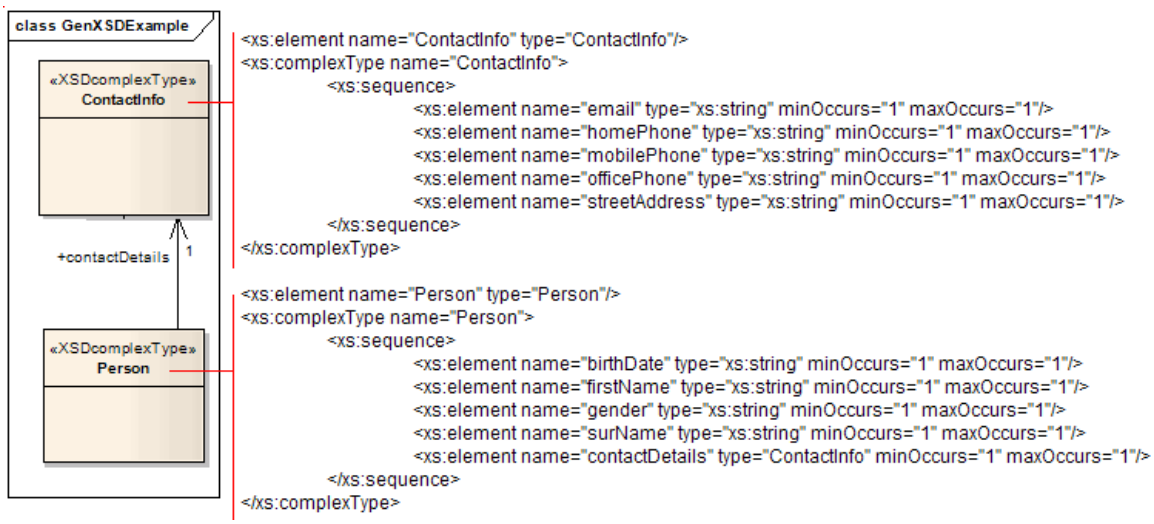
Generate Global Element

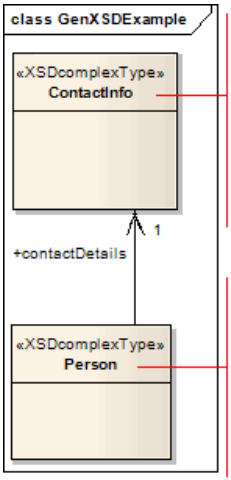
Enterprise Architect, by default, generates XML Schema in the Garden of Eden style. For every global XSDcomplexType stereotyped Class, the system generates a global element.

Example

You can change the specified default behavior by de-selecting the 'Generate global element for all global ComplexTypes' checkbox on the 'Generate XML Schema' dialog. Then, the generated XSD no longer contains the global element; that is, it no longer has the lines:

- <xs:element name="ContactInfo" type="ContactInfo"/> and
- <xs:element name="Person" type="Person"/>






```
<xs:complexType name="ContactInfo">
        <xs:sequence>
                <xs:element name="email" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="homePhone" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="mobilePhone" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="officePhone" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="streetAddress" type="xs:string" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
</xs:complexType>

<xs:complexType name="Person">
        <xs:sequence>
                <xs:element name="birthDate" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="firstName" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="gender" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="surName" type="xs:string" minOccurs="1" maxOccurs="1"/>
                <xs:element name="contactDetails" type="ContactInfo" minOccurs="1" maxOccurs="1"/>
        </xs:sequence>
</xs:complexType>
```

Import XSD

To reverse engineer a W3C XML Schema (XSD) file to create or overwrite a Package of your UML Class model, you can use the XML Schema Import facility.

Access

Ribbon	Develop > Schema Modeling > Import XSD

Import Schema files

Option	Action
Package	Displays the name of the selected target Package.
Directory	Type in or browse for (click on ...) the directory containing the source XSD file(s).
Selected	Lists the XML Schema(s) currently

File(s)	available for import. • To select a single file, click on it • To select several individual files Ctrl+click on each file • To select a range of files, press Shift and select the first and last file in the range
Import global elements with "Type" postfix	Select this checkbox to treat the global element and the ComplexType it is referring to as two separate entities.
Import referenced XML Schema(s)	Select this checkbox to import any XML Schema that is being referenced by any of the files selected in the 'Selected File(s)' field.
Create Diagram for XML Schema(s)	Select this checkbox to create a Class diagram under each imported XSDschema Package.
Import XSD Elements/Attributes as	Select the appropriate radio button to indicate how the inline XSDelements and XSDattributes are to be imported into a Class, either as: • UML Associations or

	• UML attributes
Import	Click on this button to begin the XSD import.
Close	Click on this button to close the dialog, without saving your option selections.
Progress	Displays system messages indicating the progress of the Schema import. On imports containing a large number of external references, it can be useful to capture the progress messages to check exactly what has been imported. To do this, right-click on the messages and: • Copy the selected messages to the clipboard (select the 'Copy Selected to Clipboard' menu option) • Copy all the messages to the clipboard (select the 'Copy All to Clipboard' menu option), or • Save all the messages to a file (select the 'Save to File' menu option)

Notes

- If an XML Schema file being imported already exists in the model, Enterprise Architect skips importing the file
- References to XSD Primitive Types are always imported as UML attributes
- References to XSD constructs in external Schema files are always imported as UML attributes
- Enterprise Architect uses the schemaLocation attribute in the XSD Import and XSD Include elements of an XML Schema to determine the dependencies between the files; this attribute must be set to a valid file path (and not a URL) for the dependent XML Schema(s) to be imported correctly

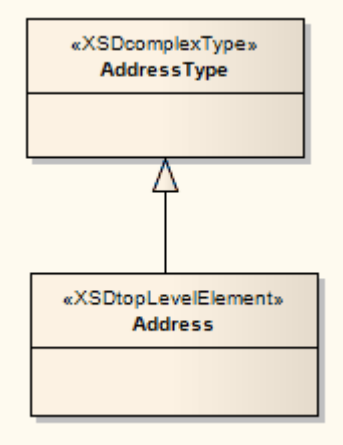
Global Element and ComplexType

Some XML Schemas have ComplexType elements with the same name as the referring global elements, but with the suffix 'Type', as shown:

```
<xs:element name="Address" type="AddressType"/>
<xs:complexType name="AddressType">
        <xs:sequence/>
</xs:complexType>
```

On XSD import, by default, Enterprise Architect treats this global element and its bounding ComplexType as a single entity, and creates a single XSDcomplexType stereotyped Class with the same name as the global element, as shown:

You can change this default behavior by selecting the 'Import global elements with "Type" postfix' checkbox on the 'Import XML Schema' dialog. When you select this option, the system treats the global element and the ComplexType it is referring to as two separate entities. For the example, the system creates an «XSDtopLevelElement» stereotyped Class for the global element and an «XSDcomplexType» stereotyped Class for the ComplexType, connected as shown:



Notes

- Enterprise Architect treats these two definitions as separate entities irrespective of whether the 'Import global elements with "Type" postfix' checkbox is selected or unselected:

```
    <xs:element name="HomeAddress"
type="AddressType"/>
    <xs:complexType name="AddressType">
        <xs:sequence/>
    </xs:complexType>
```

XSL Transforms

Model, Author and Execute XSLT Transforms and Stylesheets with XML Documents

Enterprise Architect provides facilities for modeling and executing XSL Transformations. XSLT is a technology that can be used to convert XML input documents into other types of document. Stylesheets are the XSL components used to transform the content. Facilities include:

- Specialized diagram and toolbox for modeling XSLT transformations
- Specialized editor for Stylesheet authoring, debugging and execution.
- XML document validation
- XML Schema validation

You model a transformation using the XML Transform diagram. On this diagram you can create xml documents and stylesheets, link them to a transformation (Activity) and then execute or debug the transformation. Inputs to the XSL Transform model are the XSLT and XML File Artifacts, which can be selected from the toolbox. These artifacts are most commonly created by dragging appropriate xml and xsl files on to the diagram. Output from the Transformation is described using the Output Artifact. The progress and success/failure of the transformation is shown on the 'XSLT' tab of the System Output window.

Create the XML Transform Diagram

Step	Action
1	In the Browser window, right-click on the appropriate Package and select the 'Add Diagram' option.

Artifact Elements in the XML Transformation Toolbox

Artifact	Description
XML Transform	The model reference for the transformation, providing inputs and optional outputs. Used to run or debug the transformation. Inputs: XML File, XSLT Outputs: Output Artifact (Optional)
2	In the 'New Diagram' dialog, type an appropriate diagram name in the 'Name' field (if necessary) and click on

	'Extended' in the 'Select From' list and 'XML Transform' in the 'Diagram Types' list. Click on the OK button. The new diagram opens, with the Diagram Toolbox showing the 'XML Transformation' page.
XSLT	Identifies the stylesheet to execute. Inputs: N/A Outputs: N/A
XML File	Identifies the input document to transform. Inputs: N/A Outputs: N/A
XSD	Identifies the schema that can be used, optionally, to automatically perform XML validation on the output document. Inputs: Output Artifact, XML File, (or optional both) Outputs: N/A
Output Artifact	Use this Artifact to define the output of an XSLT operation. The Artifact provides the file path to use when output is created by the transform. To select or name the output file, double-click on the Artifact to

	display its properties and enter the file path under the 'Files' tab. To make use of the Artifact, draw a trace connector to it from the transform element.

Manually Validate documents

Using Enterprise Architect, you can perform XML validation both of documents to be transformed and of XSLT stylesheets.

To run the validation, right-click in the XML document or stylesheet in the XSL Debugger and select 'XML Validation'. A prompt displays to confirm if you are validating against a document type definition or an XML Schema.

- For a document type definition, simply click on the OK button; the validation proceeds
- For an XML schema, select the appropriate radio button to identify if the validation grammar is defined within the document or elsewhere; if elsewhere, enter the namespace and URL or file path for the grammar

If errors are found during a debugging run, they will be output to the Debug window (press Alt+8 to display this window).

If errors are found during a normal validation run, they will be output to the 'XSL' tab of the System Output window

(press Alt+1 and select 'System Output' if this window does not display automatically). To locate the error in the document, double-click on the error message.

XSLT Processor and Version

The XSL Processor used in these features is built from the Apache Xalan Project (C++ version 1.11)

Model an XSL Transformation

When you model an XSL Transformation, you can either draw on files that already exist in your file system, or you can create the contents of the stylesheet and source within the model elements.

Model elements from existing files

This is the simplest and most common method for modeling transformations. When you drag a file on to the XML Transform diagram, the appropriate Artifact element is generated for you. You can then use the Quick Linker to link the file Artifact elements as inputs to the XML Transform element, using Trace connectors.

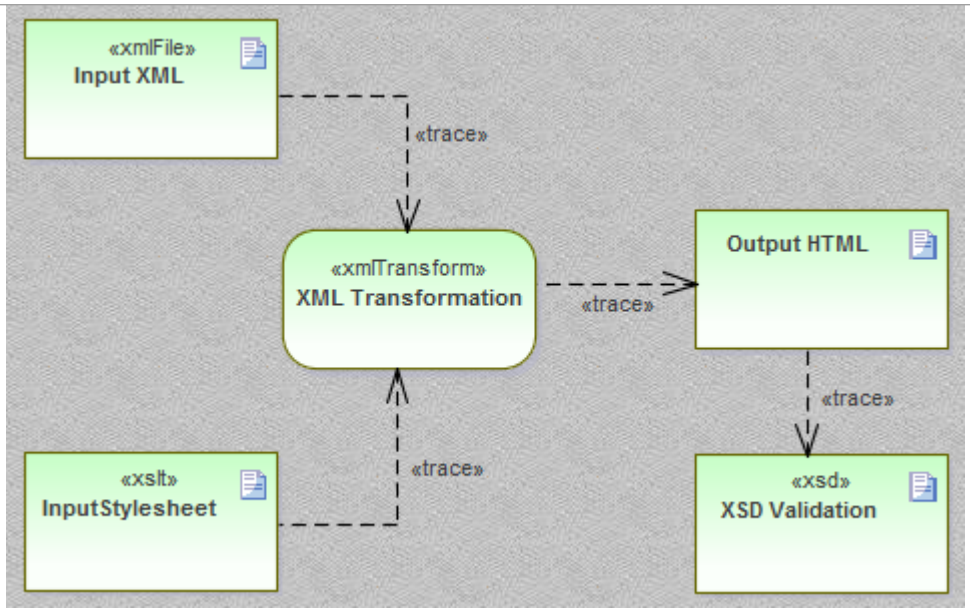
Optionally, you can:

- Specify an alternative output location (file) by linking an XML File or Output Artifact to a Trace connector from the XML Transform Artifact
- Validate the output document by dragging an XSD schema file on to the diagram and connecting the resulting XSD element to any Output Artifact of the XML Transform element

Step	Action
1	Open your file browser and the XML Transform

	diagram.
2	In the file browser, click on the input file and drag it onto the XML Transform diagram. A prompt displays to save the file as an: • External Artifact, where the XML File Artifact serves as a shortcut to the file in the file system • Internal Artifact, where the file content is read into the XML File Artifact and stored inside the model; you would select this option to make the source file contents available to other users of the model
3	Select the 'External Artifact' option. An XML File Artifact element is generated for the input file.
4	In the file browser, click on the XSL stylesheet file and drag it onto the XML Transform diagram. In response to the prompt, select the 'External Artifact' option. An XSLT element is generated for the stylesheet file.
5	Drag the XML Transform icon from the Toolbox onto the diagram, to create an XML Transform Activity element. If you prefer, give this element a new name.

6	Dragging the Trace icon from the 'Common' Toolbox page, create relationships between the: • Input XML file element and the transformation Activity element • XSLT Stylesheet file element and the transformation Activity element
7	(Optional.) • If you want to capture the output in a file, locate the appropriate file in the file browser and drag it onto the diagram to generate another File Artifact; link it to the XML Transform element with a Trace connector • If you want to validate the output document, locate the XSD schema file in the file browser and drag it on to the diagram to generate an XSD element; link this to the output File Artifact (or any Output Artifact) of the Transform element
8	Press Ctrl+S to save the diagram contents. If the output is intended to be HTML, your diagram might resemble this:



Modeling elements from scratch

When you use the 'XML Transformation' Toolbox to create XSLT and XML File elements, the system stores these as model documents. You double-click on the elements on the XML Transform diagram to open the model documents in Enterprise Architect's XSLT Debugger, where you can write and edit the file contents. When the document is saved, the contents will be saved back to the model.

Otherwise, the process of modeling a transformation is the same as described in *Modeling elements from existing files*.

Edit Documents with the XML Editor

Enterprise Architect provides a robust and useful XML editor with many features including:

- Intelli-sense
- Contextual structure tree providing quick alternative navigation (tip: press Ctrl+1 to toggle document tree view)
- Custom icons for XSL and XSD document elements
- Code completion and
- Validation of document and referenced schemas

The XML Editor will open when any document with an XML declaration is opened within Enterprise Architect. (Alternatively, press Ctrl+Shift+O.) The XSLT Debugger uses two XML editors side-by-side, to display both the stylesheet and the document being transformed.

Execute an XSL Transformation

After you have modeled an XSL Transformation, you can execute it directly from the model diagram. You can also perform the transformation directly from the XSL Stylesheet and input files.

Execute the Transformation From the Diagram

Step	Action
1	On the XML Transform diagram, right-click on the XML Transform Activity element and select the 'Run XSL Transformation' option. The XSLT Debugger view displays, showing the stylesheet (.xsl) file and XML document used in the transformation. The System Output window also displays, showing the error or success messages in the 'XSL' tab. (Press Ctrl+Shift+8 if the System Output window does not display.) If you have set up validation of the output, the System Output window also shows the validation comments.

2	If you have directed the output to a file via an Output or File Artifact, press F12 to view the output.

Debug an XSL Transformation

When you use the XSLT debugger to run a transformation you can control the process and inspect the state of the transformation using Enterprise Architect's debugger in combination with breakpoints. The XSLT Debugger provides a Run button and various Step buttons. You set breakpoints by clicking in the left margin of the stylesheet.

When a step completes or a breakpoint is encountered, the context of the transformation - including any parameters to template calls - can be viewed in the Locals window ('Execute > Windows > Local Variables'). You can also display the Call Stack ('Execute > Windows > Call Stack) to see how the current state of the transformation was reached.

Debug the Transformation

Step	Action
1	On the XML Transform diagram, right-click on the XML Transform Activity element and select the 'Debug XSL Transformation' option. The XSLT Debugger view displays, showing the stylesheet (.xsl) file and XML document used in the transformation, which is automatically initiated. The currently executing statement in the stylesheet is

	highlighted. Across the top of the view is a debugger toolbar, providing the normal debugging options to Start, Pause, Step Over, Step In, Step Out and Stop the debugging process. The final icon in the toolbar provides the option of hiding or showing the '.xml source' tab in the view. You can use these buttons to repeat and control the debugging process. The System Output window also displays, showing the debugger progress messages on the 'XSLT Transformation' tab. (Press Alt+1 and select 'System Output' if the System Output window does not display.) Error messages are directed to the Debug window (press Alt+8). You can also use the Debug window toolbar buttons and options to control debugging of the XSL Transformation.
2	If necessary, select to display the Locals window and the Call Stack. Click on the left margin of the XSLT Debugger stylesheet panel and set any Breakpoints you want to use to check processing.
3	Run the debugger again and examine the execution as indicated by the System Output window, Call Stack, Locals window, and any other Debugger or Execution Analysis tools you want to apply.

XML Validation

Enterprise Architect provides validation of XML documents. Documents can be verified against XML schema or Data Type Definitions (DTD). Validation is performed from within an Enterprise Architect editor using its context menu. Often an XML document will contain information relating to the schema that it conforms to. You can, however, choose to override this, validating the document against any schema, either at a path on your local machine or at a URL. This example demonstrates the use of the feature for a document that contains an incorrect attribute.

Access

Context Menu	Accessible from context menu of any editor window displaying xml content. Right-click in editor window and choose 'XML Validation'

XML Document Validation

Step	Action
1	Open the XML document to be validated.
2	Use the editor context menu and select the 'XML Validation' option.
3	Select the grammar of choice from the available options: • XML Schema (default) • Data Type Definition
4	Select the schema location. 'Defined in document' is selected by default. It is usual for an XML document to specify the schemas that govern its content. To choose a different schema from that defined in the document, select 'External' and provide either a URL or a file path. Examples: • http://mydomain/myschema.xsd • c:\mydomain\myschema.xsd
5	Click OK. The output of the validation will be displayed in the 'XML Validation' tab of the System Output window.

XML Document Validation Example

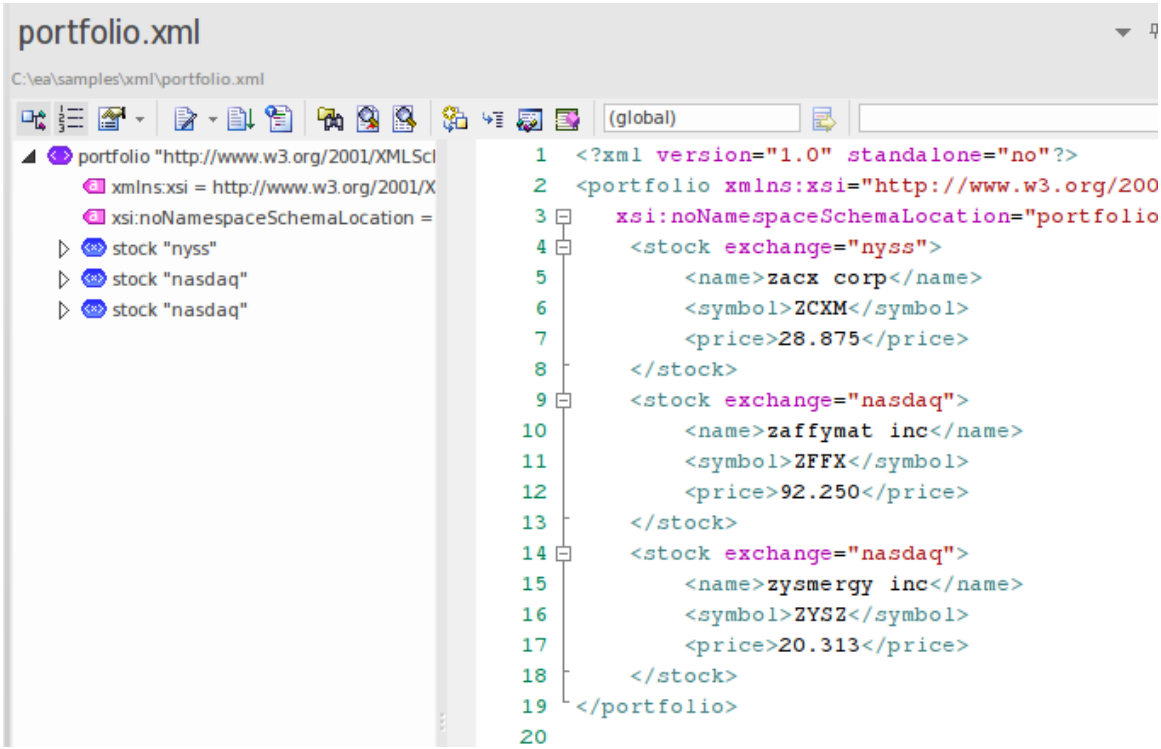


Figure 1: The XML document with an invalid attribute value 'nyss'

In this example, the document describes a stock item that has an invalid exchange code 'nyss'. As can be seen from this schema, the only valid values for the 'exchange' attribute are 'nyse', 'nasdaq' or 'ftsi'.

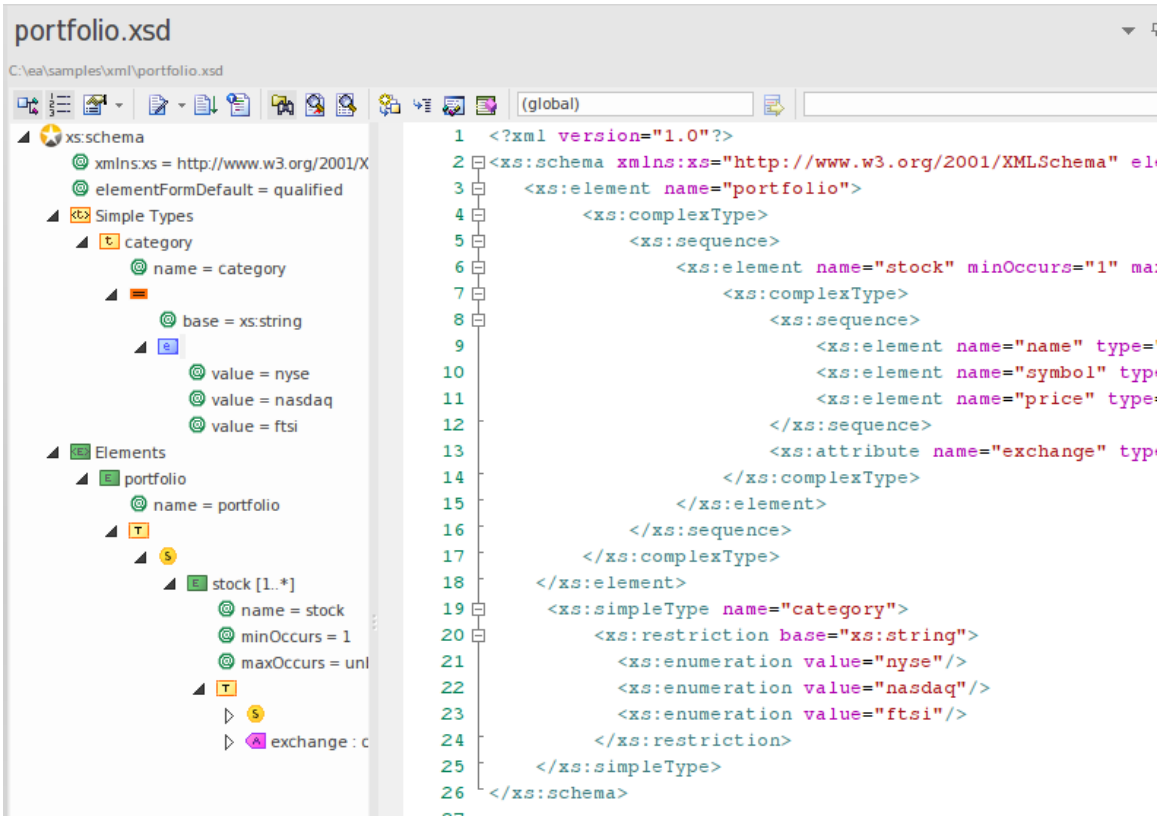


Figure 2: The XML Schema describing permitted stock exchange codes

This image shows the schema used in the validation. The declaration of a 'portfolio' element can be seen here to be made up of one or more 'stock' elements. Each stock element in turn, requires an 'exchange' attribute naming a code for the stock exchange in question.

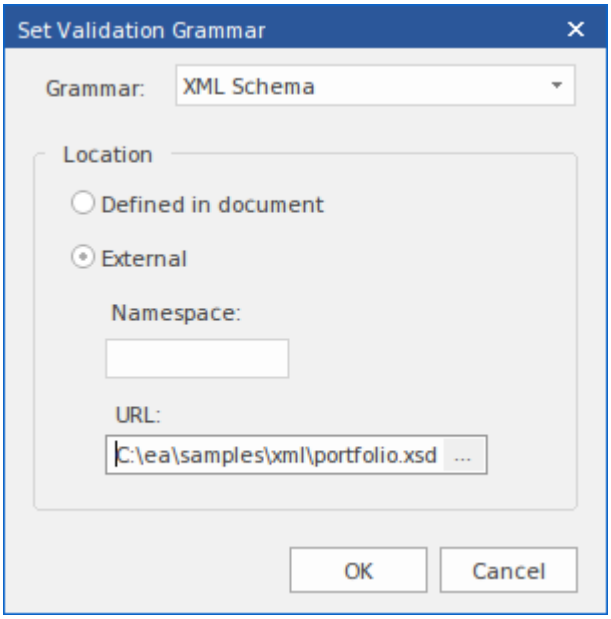


Figure 3: The 'XML validation' dialog naming a local schema file

This is the 'XML Validation' dialog. It is accessible from the context menu of any editor in Enterprise Architect that holds XML content. Here you can select the schema to use in the validation. In the example the processor will validate the document using a local schema file. This just happens to be the same schema named by the document, but it could be any schema (a development or later version of the schema for example).

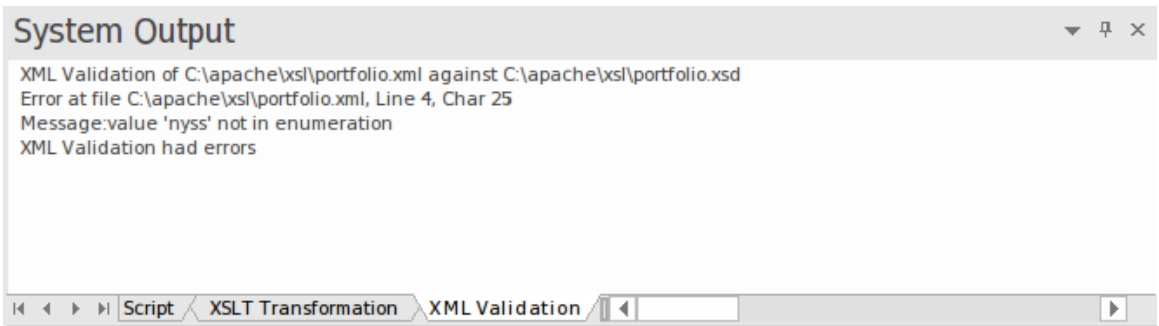


Figure 4: The System Output window showing validation error

This image shows the results of the validation. The attribute value 'nyss' has been identified as being incorrect according to the enumeration described by the schema.
Double-clicking the error will display the line of code in the editor where it can easily be correc^{ted.}

XML Service Oriented Architecture

Model Organizations, Communities and Systems to Maximize Agility, Scale and Interoperability

Service Oriented Architecture (SOA) is an architectural paradigm for defining how people, organizations and systems provide and use services to achieve results.

A service is an offer of value to another through a well-defined interface, available to a community (which could be the general public). A service results in work provided to one by another.

Service Oriented Architecture (SOA) is a way of organizing and understanding (representations of) organizations, communities and systems to maximize agility, scale and interoperability. The SOA approach is simple - people, organizations and systems provide services to each other. These services allow us to get something done without doing it ourselves or even without knowing how to do it - enabling us to be more efficient and agile. Services also enable us to offer our capabilities to others in exchange for some value - thus establishing a community, process or marketplace. The SOA paradigm works equally well for integrating existing capabilities as for creating and integrating new capabilities.

(Derived from *Service oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS)* (OMG document

ad/2008-11-01); pp. 25-26.)

In modeling and developing a complete Service Oriented Architecture in Enterprise Architect, you can work with any or all of:

- XML Schema Definition (XSD), also known as XML Schema - an XML technology that is used to specify the rules to which an XML document must adhere; Enterprise Architect provides a Schema Composer interface to help you model and generate XML schema
- XSL Transformations to convert input documents into XML or other types of document using XSL stylesheets, for which you use the XSLT Editor and Debugger for modeling and executing the transformations
- Web Services Description Language 1.1 (WSDL) - a key XML-based language for describing web services
- Service oriented architecture Modeling Language (SoaML) - a standard method of designing and modeling SOA solutions using the Unified Modeling Language (UML)
- Service-Oriented Modeling Framework (SOMF) - a service-oriented development life cycle methodology, offering a number of modeling practices and disciplines that contribute to successful service-oriented life cycle management and modeling
- National Information Exchange Modeling (NIEM) - a common framework that is used to define how information can be shared between systems, government agencies and departments

- Meta-Object Facility (MOF) - an Object Management Group (OMG) standard developed as a meta-modeling architecture to define the UML, and so provide a means to define the structure or abstract syntax of a language or of data

WSDL

Web Services Description Language 1.1 (WSDL) is a key XML-based, World Wide Web Consortium (W3C) language for describing web services. WSDL support is critical for the development of a complete Service Oriented Architecture (SOA), and the coupling of UML 2.5 and XML provides the natural mechanism for specifying, constructing and deploying XML-based SOA artifacts within an organization.

Using Enterprise Architect, you can rapidly model, forward engineer and reverse engineer WSDL files.

WSDL 1.1 Model Structure

A Web Service Description Language (WSDL), under specification 1.1, is defined within a «WSDLnamespace» stereotyped Package, which represents the top-level container for the WSDL elements. Conceptually it maps to the targetNamespace in a WSDL definition element.

When you create a WSDL model, Enterprise Architect creates the Namespace and provides a set of sub-Packages, each containing a diagram on which to define the constituent elements of the model, with an Overview diagram to navigate between the sub-Packages. You work through the sub-Packages in sequence, to define the objects that are used by later objects, themselves called into still later objects.

WSDL Structure Development

WSDL Element Type	Description
Types	Defined in an XSD Schema, these are the XSD data types used by the web service and communicated by WSDL Messages; you drag «XSDelement», «XSDsimpleType» and «XSDcomplexType» stereotyped

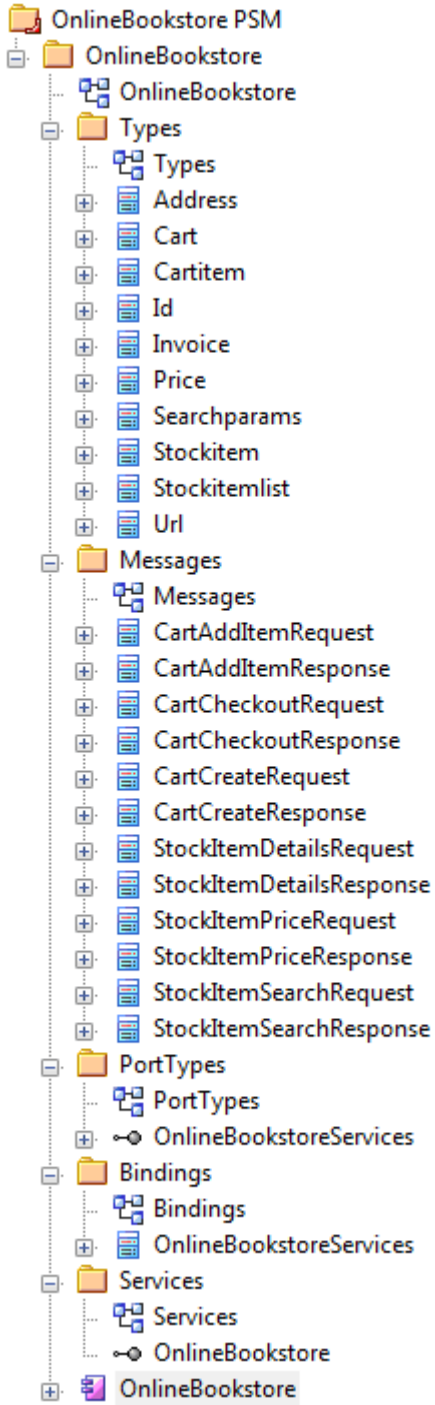
	elements onto the Types diagram from the 'XML Schema' page of the Diagram Toolbox.
Messages	WSDL Messages identify the data being communicated by a web service. Each Message element contains one or more Message Parts, which are attributes that each identify an XSD data type being communicated.
Port Types	WSDL Port Types are the essential core of the web service, defining the interfaces of the service. Each Port Type consists of a set of Port Type Operations, each of which identifies an exchange of Messages (data input to and output from the interface as that operation). The Port Type Operation can also identify Messages acting as Fault indicators.
Bindings	A Binding specifies the protocol and data format for the operations and messages defined for of a particular Port Type. Each «WSDLbinding » Class implements (realizes) the operations specified by the «WSDLportType» Interface - the Port Type Operations in the Port Type element are automatically copied into the Binding

	element as Binding operations.
Services	A WSDL Service defines a formal interface of the web service. It describes the collection of Port Types that expose a particular Binding, having an Association to each exposed Binding. It therefore encapsulates a set of the other data structures - if not all the data structures - defined in the model.
Documents	WSDL Documents are represented by Components having the stereotype «WSDL». This is the element from which you generate the WSDL file. You can create more than one Document to re-use the schema Types, Messages, Port Types, Bindings and Services of a Namespace across multiple physical WSDL documents, either in the same configuration or in different configurations.

Example

This figure shows an example WSDL namespace,

OnlineBookstore PSM, which includes a single WSDL document, OnlineBookstore (at the bottom of the hierarchy).



Notes

- You can also generate a WSDL Package structure from a UML Interface using the WSDL Model Transformation

Model WSDL

You can quickly and easily model the elements in a Web Service Definition using the WSDL page of the Diagram Toolbox. As a first step, you can create an example WSDL Package structure in the Browser window, using the Namespace icon from the WSDL page. You can use this example Package structure as a template for developing your WSDL.

Create a new WSDL Package structure

Step	Action
1	In the Browser window, create the top-level project structure you need (Model and Views), and click on the appropriate View.
2	Click on the 'New Package' option in the Browser window header drop-down list. The 'New Model Package' dialog displays.
3	In the 'Name' field type the name of the new Package, and select the 'Create Diagram' radio button.

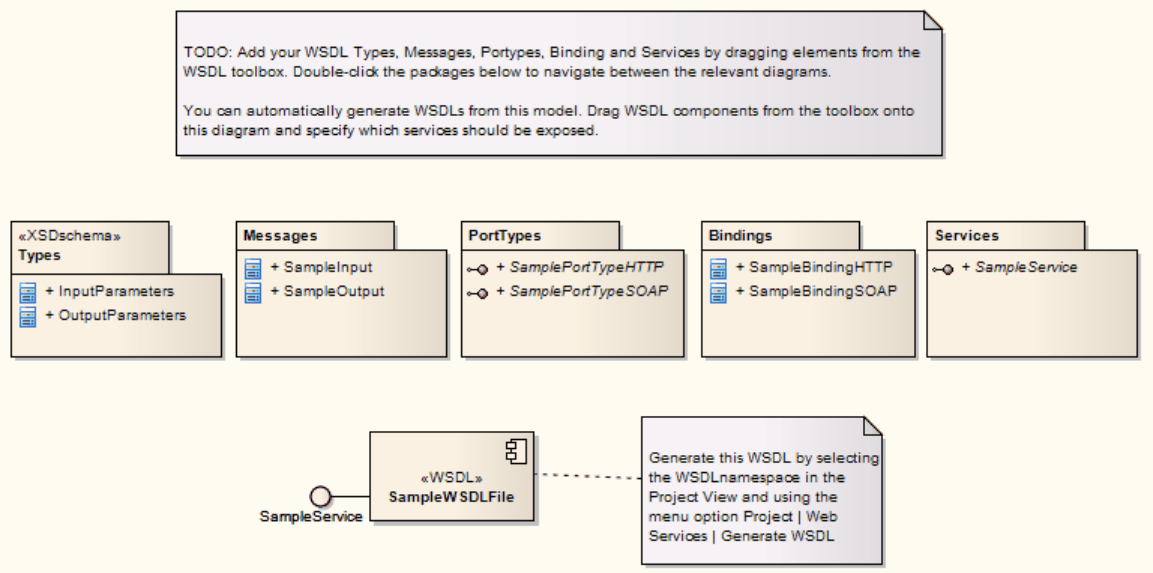
4	Click on the OK button. The 'New Diagram' dialog displays.
5	In the 'Name' field type the name of the new diagram. In the 'Select From' panel select 'UML Structural', and in the 'Diagram Types' panel select 'Class'.
6	Click on the OK button. In the Browser window, double-click on the icon next to the new diagram's name; the diagram opens in the Diagram View, with the Class pages displaying in the Diagram Toolbox.
7	In the Toolbox, click on  to display the 'Find Toolbox Item' dialog and specify 'WSDL', then select the Toolbox page from the results. The 'WSDL' Toolbox page displays.
8	Click on the 'Namespace' icon from the Toolbox and drag it into the Class diagram. The 'WSDL Namespace Properties' dialog displays. Type in a WSDL Package name and the URL of the Target Namespace. You can edit these values later.
9	Click on the OK button. The sample «WSDLnamespace» stereotyped Package structure is created on the diagram, and the

	full model structure is displayed, expanded, in the Browser window. The model structure consists of these sub-Packages, with an Overview diagram to navigate between them: • Types: Contains the XSD types for the data communicated by the web service, on a Types diagram • Messages: Contains the WSDL Messages, modeled as UML Classes marked with the stereotype «WSDLmessage» • PortTypes: Contains the WSDL Port Types, modeled as UML interfaces marked with the stereotype «WSDLportType» • Bindings: Contains the WSDL Bindings, modeled as UML Classes that realize the PortTypes • Services: Contains the WSDL Services, modeled as UML interfaces with Associations to each exposed Binding
10	Model each of the WSDL constructs in their corresponding Packages.

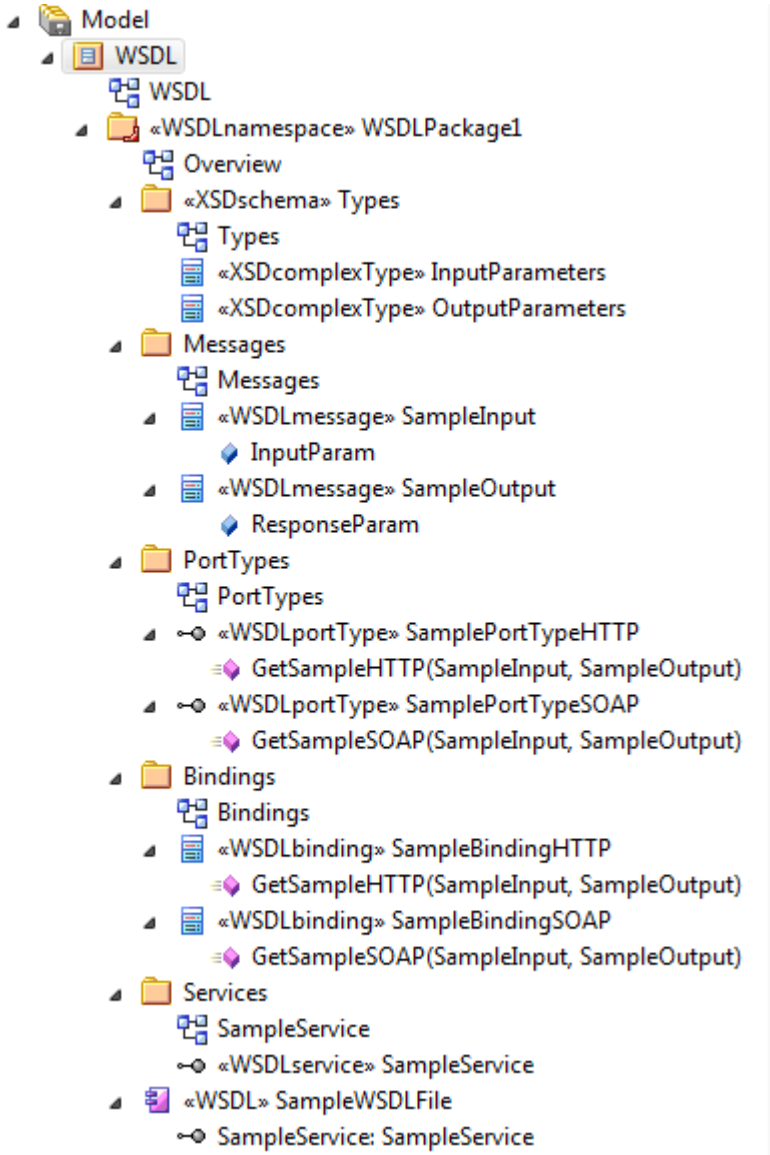
Template WSDL Model - Diagram

The WSDLnamespace Package acts as a container for the

WSDL structure.



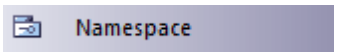
Template WSDL Model - Browser Window Hierarchy


Model
WSDL
WSDL
«WSDLnamespace» WSDLPackage1
Overview
«XSDschema» Types
Types
«XSDcomplexType» InputParameters
«XSDcomplexType» OutputParameters
Messages
Messages
«WSDLmessage» SampleInput
InputParam
«WSDLmessage» SampleOutput
ResponseParam
PortTypes
PortTypes
«WSDLportType» SamplePortTypeHTTP
GetSampleHTTP(SampleInput, SampleOutput)
«WSDLportType» SamplePortTypeSOAP
GetSampleSOAP(SampleInput, SampleOutput)
Bindings
Bindings
«WSDLbinding» SampleBindingHTTP
GetSampleHTTP(SampleInput, SampleOutput)
«WSDLbinding» SampleBindingSOAP
GetSampleSOAP(SampleInput, SampleOutput)
Services
SampleService
«WSDLservice» SampleService
«WSDL» SampleWSDLFile
SampleService: SampleService

WSDL Namespace

A «WSDLnamespace» stereotyped Package represents the top-level container for the WSDL constructs in Enterprise Architect. You can create the Namespace Package by dragging the Namespace icon from the WSDL Toolbox page and dropping it directly onto a diagram.

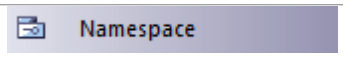
Toolbox Icon



Access

To display the 'WSDL Namepace Properties' dialog for the selected «WSDLnamespace» stereotyped Package, use one of the methods outlined here.

Ribbon	Design > Package > Manage > Properties
Context Menu	Right-click on «WSDLnamespace» stereotyped Package | Properties
Other	In Browser window, double-click on «WSDLnamespace» stereotyped Package, or

	Drag  icon from toolbox onto a diagram (this creates a new «WSDLnamespace» stereotyped Package)

Define Properties

Option	Action
WSDL Package Name	Type in the name of the WSDL Namespace Package element.
Target Namespace	(Optional) Type in the URL for the WSDL Namespace Package.
OK	Click on this button to save the values entered and close the WSDL Namespace 'Properties' dialog. If you have just created the Namespace, a new Package diagram opens containing the sample template WSDL model.
Cancel	Click on this button to discard the data entered and close the 'WSDL Namespace Properties' dialog.

Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing WSDL Namespace element information. Click on the button to open the UML element 'Properties' dialog for the Namespace Package element.

WSDL Message

A «WSDLmessage» stereotyped Class represents a WSDL Message and acts as a container for one or more WSDL Message Parts. You can create WSDL Messages by dragging the Message icon from the WSDL Toolbox and dropping it directly onto the Messages diagram (under the Messages Package in the WSDL Package structure).

Toolbox Icon

Access

To display the 'WSDL Message' dialog for the selected «WSDLmessage» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «WSDLmessage» stereotyped Class| Properties
Keyboard Shortcuts	Alt+Enter

Other	Double-click on a «WSDLmessage» stereotyped Class, or Drag  icon from the toolbox and drop directly onto the Messages diagram, under the Messages Package in the WSDL Package structure. (This creates a new «WSDLmessage» stereotyped Class.)

Define Properties

Option	Action
Name	Type in the name of the WSDL Message.
Documentation	(Optional) Type in any notes you need for this element.
OK	Click on this button to save the data entered and close the WSDL Message dialog.
Cancel	Click on this button to discard the data entered and close the 'WSDL Message' dialog.

Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing WSDL Message element information. Click on the button to open the UML Class 'Properties' dialog for the element.

Notes

- WSDL Messages can only be created under the Messages Package in the WSDL Package structure
- The name of the WSDL Message should be unique amongst all WSDL Messages within the WSDL

WSDL Message Part

A WSDL Message Part is the segment of a WSDL Message that identifies the XSD data type of the data communicated by the Message. If a Message communicates data of more than one data type, each data type is identified in a separate Message Part.

In Enterprise Architect, a WSDL Message Part is represented by a UML attribute of the WSDL Message Class. You can create the Message Part attribute by dragging the 'Message Part' icon from the WSDL Toolbox and dropping it onto a «WSDLmessage» stereotyped Class.

Toolbox Icon

Access

To display the 'WSDL Message Part' dialog for the selected Message Part, use one of the methods outlined here.

Ribbon	With a specific Message Part (attribute) selected within a WSDL Message on a diagram: Design > Element > Features > Attributes

Context Menu	With a specific Message Part (attribute) selected within a WSDL Message on a diagram: Right-click on attribute | View Properties
Keyboard Shortcuts	With a specific Message Part (attribute) selected within a WSDL Message on a diagram: F9
Other	Double-click on the Message Part (attribute) within the «WSDLmessage» stereotyped Class, or Drag  icon from toolbox and drop onto a «WSDLmessage» stereotyped Class (This creates a new Message Part (as an attribute) within the «WSDLmessage» stereotyped Class.)

Define Properties

Option	Action
Name	Type in the name of the WSDL Message

	Part attribute.
Type	Either: • Type the name of a data type, or • Click on the drop-down arrow and select an XSD built-in dataType from the list, or • Click on the ... button and browse for an existing «XSDelement», «XSDcomplexType» or «XSDsimpleType» element as a classifier The XSD Types can be defined in: • The Types Package under the WSDL Package Structure or • Any other Package in the model
OK	Click on this button to save the data entered and close the 'WSDL Message Part' dialog.
Cancel	Click on this button to discard the data entered and close the 'WSDL Message Part' dialog.
Help	Click on this button to display this Help topic.

UML	This button is displayed when you are editing existing WSDL Message Part attribute information. Click on the button to open the attribute properties for the Message Part.

Notes

- WSDLmessage stereotyped Classes can effectively contain Message Part attributes only; if you add other attributes to the Class element, they are re-cast as Message Parts

WSDL Port Type

A «WSDLportType» stereotyped Interface represents a WSDL PortType. It describes the operations exposed by the WSDL, acting as a container for one or more WSDL Port Type Operations. You can create a WSDL PortType element by dragging the Port Type icon from the WSDL Toolbox and dropping it directly onto the PortTypes diagram (under the PortTypes Package in the WSDL Package structure).

Toolbox Icon

Access

To display the 'WSDL PortType' dialog for the selected «WSDLportType» stereotyped Interface, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «WSDLportType» stereotyped Interface| Properties

Keyboard Shortcuts	Alt+Enter
Other	• Double-click on a «WSDLportType» stereotyped Interface, or • Drag  icon from the toolbox and drop directly onto the PortTypes diagram, under the PortTypes Package in the WSDL Package structure (This creates a new «WSDLportType» stereotyped Interface.)

Define Properties

Option	Action
Name	Type in the name of the WSDL PortType.
Documentation	(Optional) Type in any notes you need for this element.
OK	Click on this button to save the data entered and close the WSDL PortType dialog.

Cancel	Click on this button to discard the data entered and close the 'WSDL PortType' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing WSDL PortType element information. Click on the button to open the UML element 'Properties' dialog for the element.

Notes

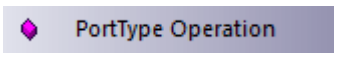
- WSDL PortTypes can only be created under the PortTypes Package in the WSDL Package structure
- The name of the WSDL PortType should be unique amongst all the WSDL PortTypes within the WSDL

WSDL Port Type Operation

A Port Type Operation identifies an exchange of Messages (data input to and output from the interface as an operation). The Port Type Operation can also identify Messages acting as Fault indicators.

In Enterprise Architect, a WSDL PortType Operation is represented by a UML Operation of the WSDL PortType Interface. You can create a PortType Operation by dragging the PortType Operation icon from the WSDL Toolbox and dropping it onto a «WSDLportType» stereotyped Interface.

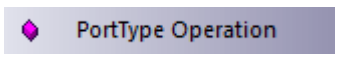
Toolbox Icon



Access

To display the 'WSDL PortType Operation' dialog for the selected PortType Operation, use one of the methods outlined here.

Ribbon	With a specific PortType Operation selected within a «WSDLportType» stereotyped Interface on a diagram: Design > Element > Features >

	Operations
Context Menu	With a specific PortType Operation selected within a «WSDLportType» stereotyped Interface on a diagram: Right-click on attribute | View Properties
Keyboard Shortcuts	With a specific PortType Operation selected within a «WSDLportType» stereotyped Interface on a diagram: F10
Other	Double-click on the PortType Operation within the «WSDLportType» stereotyped Interface, or Drag  icon from toolbox and drop onto a «WSDLportType» stereotyped Interface. (This creates a new PortType Operation (as a UML operation) within the «WSDLportType» stereotyped Interface.)

Define Properties

Option	Action

Name	Type in the name of the WSDL PortType Operation.
Documentation	(Optional) Type in any notes you need for this operation.
Operation Type	Click on the drop-down arrow and select one of the supported PortType Operation types: • OneWay • Request-Response • Solicit-Response • Notification
Input	This section is grayed out if you have selected Notification as the operation type. • Name - Defaults to a name that parallels theOperation Type. If you do not want to use the default, type an alternative name for the input Message. • Message - Click on the drop-down arrow and select one of the WSDL Messages previously created in theMessagePackage. • Documentation - (Optional) Type in any notes you need for this input Message.

Output	This section is grayed out if you have selected OneWay as the operation type. • Name - Defaults to a name that parallels theOperation Type. If you do not want to use the default, type an alternative name for the output Message. • Message - Click on the drop-down arrow and select one of the WSDL Messages previously created in theMessagePackage. • Documentation - (Optional) Type in any notes you need for this output Message.
Faults	Review the details of the WSDL Messages that can act as Faults. Faults display in this list with the most recently-created at the top and the oldest at the end. If more than four Fault Messages are defined, use the vertical scroll bar to display the rest of the list. To add a Message, click on the New button. The 'WSDL PortType Operation Fault' dialog displays. • Name - Defaults to 'Fault<n>'; if you do not want to use the default, type an alternative name for the fault Message

	• Message - Click on the drop-down arrow and select one of the WSDL Messages previously created in the Message Package • Documentation - (Optional) Type in any notes you need for this fault Message • OK - Click on this button to save the data entered and close the 'WSDL PortType Operation Fault' dialog • Cancel - Click on this button to discard the data entered and close the 'WSDL PortType Operation Fault' dialog • Help - Click on this button to display this Help topic To remove a Message from the list, click on it and click on the Delete button.
OK	Click on this button to save the data entered and close the 'WSDL PortType Operation' dialog.
Cancel	Click on this button to discard the data entered and close the 'WSDL PortType Operation' dialog.
Help	Click on this button to display this Help topic.

UML	This button is displayed when you are editing existing WSDL Port Type Operation information. Click on the button to open the UML operation 'Properties' dialog for the element.

Notes

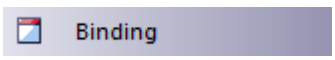
- WSDL PortType Operations can only be contained by WSDL PortTypes
- The name provided for an Input, Output or Fault Message in a PortType Operation must be unique amongst the Input, Output and Fault Messages, respectively, across the WSDL PortType
- In the UML operation 'Properties' dialog, the Messages identified as Input, Output and Fault can be examined as the parameters of the operation

WSDL Binding

A WSDL Binding element implements the operations specified by a particular «WSDLportType» stereotyped Interface and describes the message format and protocol details for the operations and messages defined by this WSDL PortType. A WSDL Binding element is represented by a «WSDLbinding» stereotyped Class.

You create a WSDL Binding element by dragging the Binding icon from the WSDL Toolbox directly onto the Bindings diagram under the Bindings Package in the WSDL Package structure.

Toolbox Icon



Access

To display the 'WSDL Binding' dialog for the selected «WSDLbinding» stereotyped Class, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context	Right-click on «WSDLbinding»

Menu	stereotyped Class | Properties
Keyboard Shortcuts	Alt+Enter
Other	Double-click on a «WSDLbinding» stereotyped Class, or Drag  icon from the toolbox and drop directly onto the Bindings diagram, under the Bindings Package in the WSDL Package structure. (This creates a new «WSDLbinding» stereotyped Class.)

Define Properties

Option	Action
Name	Type in the name of the WSDL Binding element.
PortType	Click on the drop-down arrow and select the WSDL PortType to be implemented by this WSDL Binding.
	Click on the drop-down arrow and select

Protocol	the protocol for the transmission of the selected WSDL PortType's operations. The supported protocols are: • SOAP • HTTP
Transport	This field is disabled if you have selected the HTTP protocol. Defaults to http://schemas.xmlsoap.org/soap/http. If necessary, type in an alternative URL for the SOAP protocol.
Style	This field is disabled if you have selected the HTTP protocol. Click on the drop-down arrow and select the style of SOAP protocol.
Verb	This field is disabled if you have selected the SOAP protocol. Click on the drop-down arrow and select the appropriate HTTP verb. The supported verbs are: • GET • POST
Documentation	(Optional) Type in any notes you need for this element.

OK	Click on this button to save the data entered and close the 'WSDL Binding' dialog.
Cancel	Click on this button to discard the data entered and close the 'WSDL Binding' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing WSDL Binding element information. Click on the button to open the UML element 'Properties' dialog for the element.

Notes

- A WSDL Binding must implement a WSDL PortType; therefore, WSDL PortTypes should be defined before you create WSDL Bindings
- WSDL Bindings can only be created under the Bindings Package in the WSDL Package structure

- The name of the WSDL Binding should be unique amongst all the WSDL Bindings within the WSDL

WSDL Binding Operation

When you save a newly-created «WSDLbinding» stereotyped Class, the system:

1. Adds to the Binding diagram, the WSDL Port Type element implemented by the WSDL Binding.
2. Draws a Realization connector from the Binding to the PortType.
3. Automatically populates the Binding with all the UML operations from the PortType, as the WSDL Binding Operations.

Access

To display the 'WSDL Binding Operation Details' dialog for the selected Binding Operation, use one of the methods outlined here.

Ribbon	With a specific Binding Operation selected within a «WSDLbinding» stereotyped Class on a diagram: Design > Element > Features > Operations
Context Menu	With a specific Binding Operation selected within a «WSDLbinding» stereotyped Class on a diagram:

	Right-click on attribute | View Properties
Keyboard Shortcuts	With a specific Binding Operation selected within a «WSDLbinding» stereotyped Class on a diagram: F10
Other	Double-click on the Binding Operation within the «WSDLbinding» stereotyped Class

Define Properties

Option	Action
Operation Name	Displays the name of the Operation copied from the WSDL PortType element. The value in this field cannot be edited.
Action	If the protocol of the parent WSDL Binding element was defined as HTTP, this field is grayed out. Type in the SOAP Action header (URL) for this operation.

Style	If the protocol of the parent WSDL Binding element was defined as HTTP, this field is grayed out. Click on the drop-down arrow and select the SOAP style of the operation.
Location	If the protocol of the parent WSDL Binding element was defined as SOAP, this field is grayed out. Type in the relative URL of this Operation.
Documentation	(Optional) Type in any notes you need for this operation.
Parameters	Click on this button to define the parameters for this operation. The 'WSDL Binding Operation Parameters' dialog displays, showing the names of the operation Input, Output and Faults. You cannot change these names. Click on the Details button to specify the details for Input, Output and Fault operation (Message) parameters. Note that the Details button in the: • Input section is disabled for the Notification Operation Type

	• Output section is disabled for the One-way Operation Type • Fault section is disabled if there are no Fault Messages • Use - If the protocol of the parent WSDL Binding element was defined as HTTP, this field is grayed out; click on the drop-down arrow and select the encoding that is to be used • Encoding Style - If the protocol of the parent WSDL Binding element was defined as HTTP, this field is grayed out; if 'Use' is set to 'encoded', type in the style (URL) to apply • Namespace - If the protocol of the parent WSDL Binding element was defined as HTTP, this field is grayed out; (Optional) type in the namespace • Parts - If the protocol of the parent WSDL Binding element was defined as HTTP, this field is grayed out; this field is also not applicable to Faults - (Optional) type in the Message Part attributes that appear within the SOAP Body portion • Header - This field is not applicable to Faults; (Optional) type in the text of the

	SOAP/HTTP Header • Documentation - (Optional) Type in any notes you need for this message • OK - Click on this button to save the data entered and close the 'WSDL Binding Parameter Details' dialog • Cancel - Click on this button to discard the data entered and close the 'WSDL Binding Parameter Details' dialog • Help - Click on this button to display this Help topic
OK	Click on this button to save the data entered and close the 'WSDL Binding Operation Details' dialog.
Cancel	Click on this button to discard the data entered and close the 'WSDL Binding Operation Details' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing WSDL Binding Operation information. Click on the button to open the UML operation 'Properties' dialog for the

	element.

Notes

- If you subsequently change the WSDL Port Type operations, you can refresh the Binding Operations by deleting the Realization connector and re-establishing it; the 'Overrides & Implementations' dialog displays, on which you select the updated operations to establish
- You can review the parameters of a Binding Operation by highlighting the operation in the diagram or Browser window and expanding the entries in the Properties window

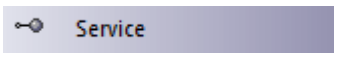
WSDL Service

A WSDL Service is represented by a «WSDLservice» stereotyped Interface; it describes a collection of Ports that expose a particular Binding. You can create a WSDL Service element by dragging the Service icon from the WSDL Toolbox and dropping it directly onto a diagram in the Services Package of your WSDL model.

When you save a newly-created «WSDLservice» stereotyped Interface, the system:

1. Adds the WSDL Binding elements exposed by the WSDL Service to the Service diagram.
2. Draws an Association connector from the Service element to each Binding element.
3. Labels each connector with the corresponding Port name.

Toolbox Icon



Access

To display the 'WSDL Service' dialog for the selected «WSDLservice» stereotyped Interface, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «WSDLservice» stereotyped Interface| Properties
Keyboard Shortcuts	Alt+Enter
Other	Double-click on a «WSDLservice» stereotyped Interface, or Drag the  icon from the toolbox and drop directly onto the SampleService diagram, under the Services Package in the WSDL Package structure. (This creates a new «WSDLservice» stereotyped Interface.)

Define Properties

Option	Action
Name	Type in the name of the WSDL Service.
Documentation	(Optional) Type in any notes you need for this element.

Ports	Identify the Ports (or endpoints) for this WSDL Service. To add a Port to the list, click on the New button. The 'WSDL Port' dialog displays. • Port Name - Type in the name of the Port • Binding - Click on the drop-down arrow and select a Binding element from the list of all the WSDL Bindings created in theBindingsPackage • Location - Type in the URL for the Port • Documentation - (Optional) Type in any notes you need for this Port • OK - Click on this button to save the values entered and close the 'WSDL Port' dialog • Cancel - Click on this button to discard the values entered and close the 'WSDL Port' dialog • Help - Click on this button to display this Help topic The Ports are organized in the list with the most recent at the top and the oldest at the end. To remove an entry from the list, click on it and click on the Delete button.

OK	Click on this button to save the data entered and close the WSDL Service dialog.
Cancel	Click on this button to discard the data entered and close the 'WSDL Service' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing WSDL Service element information. Click on the button to open the UML element 'Properties' dialog for the element.

Notes

- WSDL Services can only be created under the Service Package in the WSDL Package structure
- The name of the WSDL Service should be unique amongst all the WSDL Services within the WSDL

WSDL Document

A WSDL Document encapsulates a Web Service defined within the «WSDLnamespace» stereotyped Package, and is the source from which the WSDL file is generated. It is represented by a «WSDL» stereotyped Component element as a direct child element of the «WSDLnamespace» stereotyped Package. You can have multiple WSDL Documents under a single WSDL Namespace, to reuse and expose the WSDL Services for that namespace across multiple WSDLs.

One «WSDL» stereotyped Component element is automatically created when you create the Namespace Package structure. You can add further WSDL elements by dragging the WSDL icon from the WSDL Toolbox and dropping it directly onto the namespace Overview diagram.

Toolbox Icon

Access

To display the 'WSDL Document Properties' dialog for the selected «WSDL» stereotyped Component, use one of the methods outlined here.

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on «WSDL» stereotyped Component | Properties
Keyboard Shortcuts	Alt+Enter
Other	Double-click on a «WSDL» stereotyped Component, or Drag  icon from the toolbox and drop directly onto the Overview diagram, under the «WSDLnamespace» stereotyped Package in the WSDL Package structure. (This creates a new WSDL Document, represented by a «WSDL» stereotyped Component.)

Define Properties

Option	Action
Name	Type in the name of the WSDL document.

File Name	Type the file path into which the WSDL 1.1 file is to be generated.
Documentation	(Optional) Type in any notes you need for this element.
XMLNS	Identify the additional namespace or namespace-prefix pairs used in this WSDL Document. To add a namespace or namespace-prefix pair, click on the New button; to edit an existing entry, double-click on it. In either case, the 'Namespace Details' dialog displays. • Prefix - Type in the abbreviated text to represent the Namespace • Namespace - Type in the name of the Namespace • OK - Click on this button to save the new information and close the 'Namespace Details' dialog • Cancel - Click on this button to discard the new information and close the 'Namespace Details' dialog • Help - Click on this button to display this Help topic To remove an entry from the list, click on

	it and click on the Delete button.
Services	Review the WSDL Services that exist in the Services Package. Select the checkbox against the services to be included in the current WSDL file.
OK	Click on this button to save the data entered and close the WSDL Document 'Properties' dialog.
Cancel	Click on this button to discard the data entered and close the 'WSDL Document Properties' dialog.
Help	Click on this button to display this Help topic.
UML	This button is displayed when you are editing existing WSDL Document element information. Click on the button to open the UML element 'Properties' dialog for the element.

Generate WSDL

If you have developed a WSDL model in UML, you can forward-engineer it into WSDL 1.1 files using the Generate WSDL feature. This feature acts on either a «WSDLnamespace» stereotyped Package or a «WSDL» stereotyped Component (Document), and generates any or all of the WSDL Components owned by the target «WSDLnamespace» structure.

Access

Ribbon	Develop > Schema Modeling > Export WSDL

Generate WSDL 1.1 files

Option	Action
WSDL Package	Displays the name of the WSDL Namespace containing the source Component(s) from which the WSDL file is to be generated.

Encoding	Either: • Click on the drop-down arrow and select the XML encoding scheme you need, or • Click on the Default button to apply the default encoding scheme (UTF-8)
Select Components To Generate	Click on the «WSDL» stereotyped Component(s) in the list for which the WSDL file is to be generated. To: • Select multiple individual Components use Ctrl+click • Select a range use Shift+click • Select all entries in the list click on the Select All button • Clear all entries in the list click on the Select None button • Provide a file path and name into which to generate the WSDL file for a component, double-click on the component name; the 'Component File Name' dialog is displayed, see the table for a description
Generate	Click on this button to generate the WSDL files for the selected «WSDL»

	stereotyped Components. A message displays when the generation is complete; click on the OK button on the message to close it.
View WSDL	Click on this button to display the most recently generated WSDL.
Close	Click on this button to close this dialog.
Help	Click on this button to display this Help topic.
Progress	Monitor the progress of the WSDL file generation.

Component File Name dialog

Field/Button	Description
Name	Displays the name of the selected «WSDL» stereotyped Component.
Prefix	If multiple prefixes have been specified, click on the drop-down arrow and select the appropriate prefix for the WSDL

	Namespace.
File Name	Type in or browse for (click on ...) the file path and name into which the WSDL file is to be generated.
OK	Click on this button to save the data entered and close the 'Component File Name' dialog.
Cancel	Click on this button to discard the data entered and close the 'Component File Name' dialog.
Help	Click on this button to display this Help topic.

Notes

- You can also generate WSDL files through the Automation Interface

Import WSDL

If you have WSDL 1.1 files external to Enterprise Architect that you want to reverse engineer into UML Class models, you can import them into the system using the WSDL Import facility.

Access

Browser window | Click on root Package to contain imported file, then:

Ribbon	Develop > Schema Modeling > Import WSDL

Import a WSDL File

Option	Action
Package	Displays the name of the root Package under which the WSDL file is to be imported.
Filename	Type in or browse for (click on ...) the

	name and path of the WSDL file to import.
Target Package	Defaults to the name of the WSDL file being imported, as the name of the Package to represent the imported file. If you do not want to use the default name, type in a different name.
Import	Click on this button to start the WSDL Import. A message displays when the import is complete; click on the OK button on the message to close it.
Close	Click on this button to close this dialog.
Progress	Monitor the progress of the WSDL Import.

Notes

- Enterprise Architect cannot import a WSDL file that references WSDL constructs existing outside that file; if there are referenced constructs in other files, combine all files into a single file and import that combined file

- Example of an importable file: http://www.w3.org/TR/wsdl.html#_wsdl
- Example of a non-importable file: http://www.w3.org/TR/wsdl.html#_style; attempts to import this file result in the error message Cannot Import Split Files

SoaML

Service oriented architecture Modeling Language (SoaML) is a standard method of designing and modeling SOA solutions using the Unified Modeling Language (UML).

This text is derived from Service oriented architecture Modeling Language (SoaML) - Specification for the UML Profile and Metamodel for Services (UPMS) (OMG document ad/2008-11-01); pp. 25-26:

"A service is an offer of value to another through a well-defined interface and available to a community (which may be the general public). A service results in work provided to one by another."

"Service Oriented Architecture (SOA) is a way of organizing and understanding (representations of) organizations, communities and systems to maximize agility, scale and interoperability. The SOA approach is simple - people, organizations and systems provide services to each other. These services allow us to get something done without doing it ourselves or even without knowing how to do it - enabling us to be more efficient and agile. Services also enable us to offer our capabilities to others in exchange for some value - thus establishing a community, process or marketplace. The SOA paradigm works equally well for integrating existing capabilities as for creating and integrating new capabilities."

"SOA ... is an architectural paradigm for defining how people, organizations and systems provide and use services

to achieve results. SoaML ... provides a standard way to architect and model SOA solutions using the Unified Modeling Language (UML). The profile uses the built-in extension mechanisms of UML to define SOA concepts in terms of existing UML concepts."

"... the highest leverage of employing SOA comes from understanding a community, process or enterprise as a set of interrelated services and ... supporting that service oriented enterprise with service-enabled systems. SoaML enables business oriented and systems oriented services architectures to mutually and collaboratively support the enterprise mission. ... SoaML depends on Model Driven Architecture® (MDA®) to help map business and systems architectures, the design of the enterprise, to the technologies that support SOA, such as web services and CORBA®."

"For further information on the concepts of SoaML, see the specification document on the OMG website SoaML document page."

SoaML in Enterprise Architect

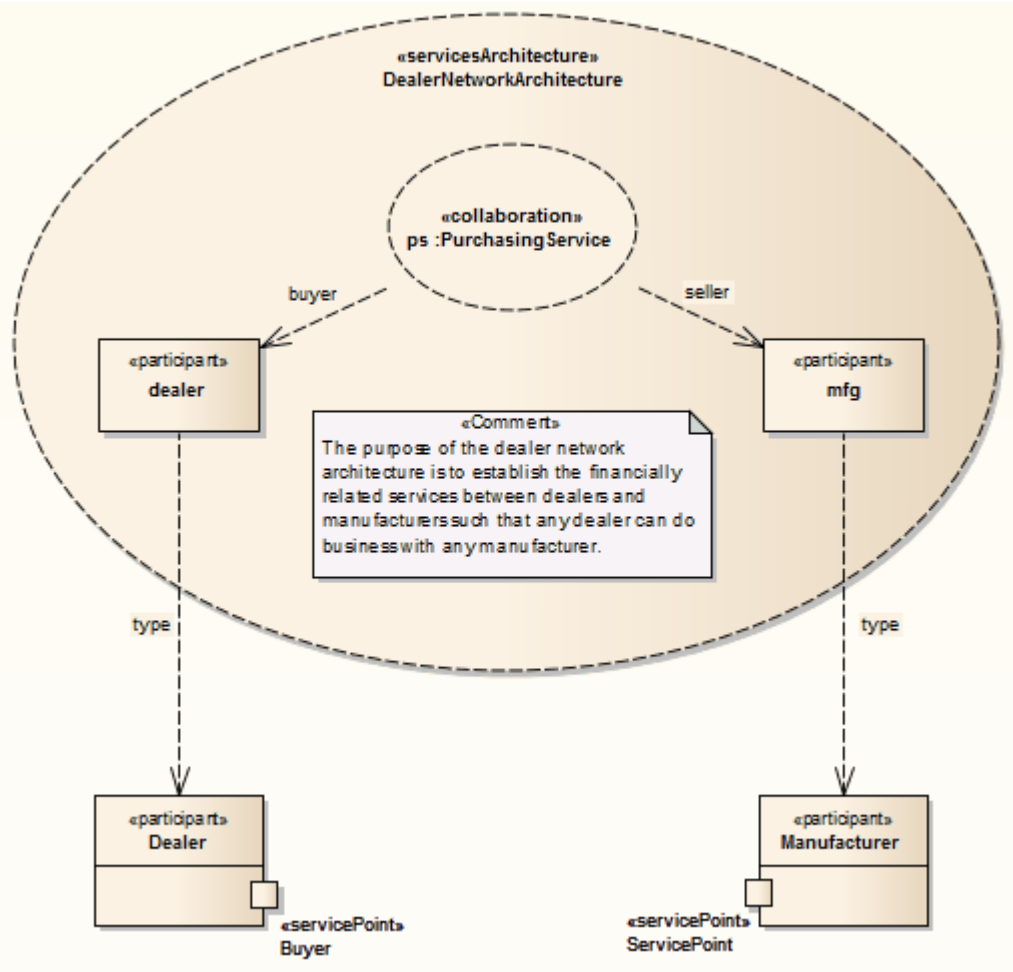
In Enterprise Architect you can model services architectures quickly and simply through use of an MDG Technology integrated with the Enterprise Architect installer. The SoaML facilities are provided in the form of:

- Two SoaML diagram types - SoaML Component diagram and SoaML Sequence diagram - accessed through the

'New Diagram' dialog

- SoaML pages in the Diagram Toolbox
- SoaML element and relationship entries in the 'Toolbox Shortcut' menu and Quick Linker

Example SoaML Diagram



Notes

- Service Oriented Architecture Modeling Language (SoaML) is supported in the Corporate, Unified and Ultimate Editions of Enterprise Architect

SoaML Toolbox Pages

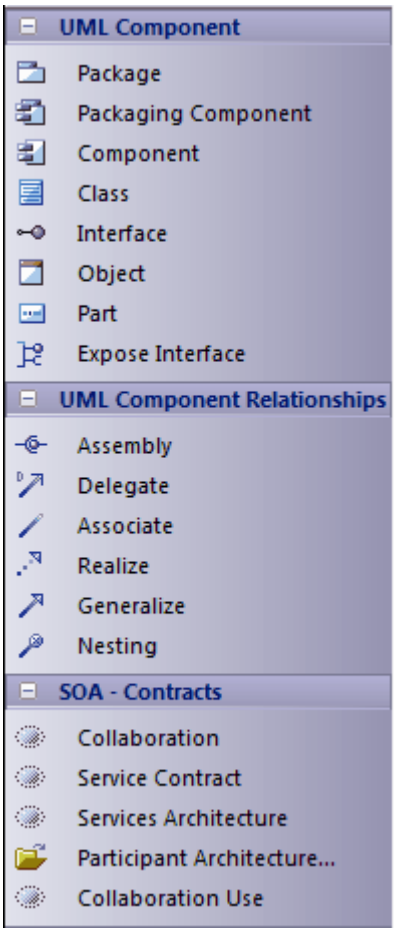
You can create the elements and relationships of the SoaML model using the 'SoaML' pages of the Diagram Toolbox. Each of the two SoaML diagram types has a separate set of pages, although the last five (SOA-specific) pages in the two sets are identical.

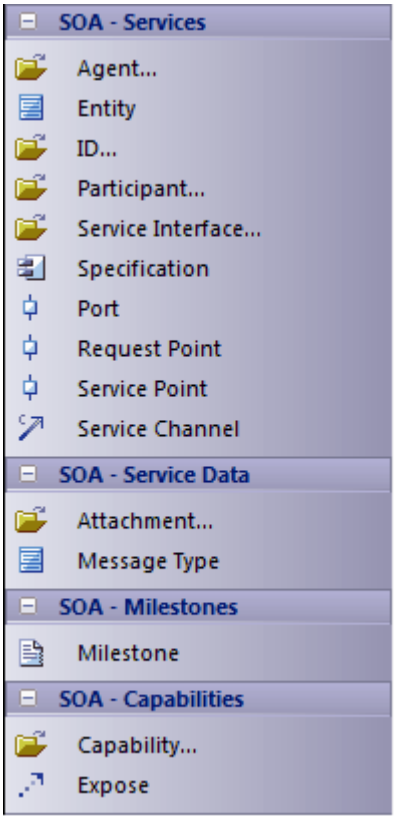
Access

Ribbon	Design > Diagram > Toolbox :  to display the 'Find Toolbox Item' dialog and specify 'SoaML Component' or 'SoaML Sequence'
Keyboard Shortcuts	Ctrl+Shift+3 :  > Specify 'SoaML Component' or 'SoaML Sequence' in the 'Find Toolbox Item' dialog
Other	Diagram caption bar | Click the  icon to display the Diagram Toolbox :  > Specify 'SoaML Component' or 'SoaML Sequence' in the 'Find Toolbox Item' dialog

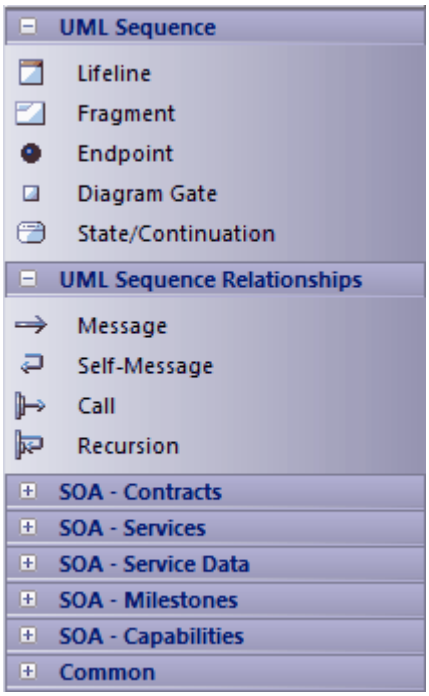
Toolbox Pages

SoaML Component Diagram Toolbox





SoaML Sequence Diagram Toolbox



SOMF 2.1

The Service-Oriented Modeling Framework (SOMF) is a service-oriented development life cycle methodology, offering a number of modeling practices and disciplines that contribute to successful service-oriented life cycle management and modeling. This text is derived from the extensive Wikipedia entry on Service Oriented Modeling:

'The Service-Oriented Modeling Framework (SOMF) has been proposed by author Michael Bell as a holistic and anthropomorphic modeling language for software development that employs disciplines and a universal language to provide tactical and strategic solutions to enterprise problems. The term "holistic language" pertains to a modeling language that can be employed to design any application, business and technological environment, either local or distributed. This universality may include design of application-level and enterprise-level solutions, including SOA landscapes or Cloud Computing environments. The term "anthropomorphic", on the other hand, affiliates the SOMF language with intuitiveness of implementation and simplicity of usage.'

'SOMF ... illustrates the major elements that identify the “what to do” aspects of a service development scheme. These are the modeling pillars that will enable practitioners to craft an effective project plan and to identify the milestones of a service-oriented initiative—either a small or large-scale business or a technological venture.'

SOMF in Enterprise Architect

In Enterprise Architect, SOMF 2.1 is implemented as a profile within an MDG Technology that is integrated with the Enterprise Architect installer. The SOMF 2.1 facilities are provided in the form of:

- Eleven SOMF diagram types, accessed through the 'New Diagram' dialog:
  - Conceptual
  - Analysis
  - Cloud Computing
  - Logical Design Relationship
  - Logical Design Composition
  - Business Integration
  - Conceptual Architecture
  - Asset Utilization
  - Transaction
  - Transaction Directory
  - Reference Architecture
- SOMF pages in the Toolbox - Enterprise Architect includes several Toolbox pages of modeling structures for each SOMF 2.1 diagram type, located through the Toolbox search facilities; these provide a wide breadth of SOMF modeling capabilities
- SOMF element and relationship entries in the Toolbox Shortcut menu and Quick Linker

National Information Exchange Modeling (NIEM) 2.1

National Information Exchange Modeling (NIEM) provides a common framework that is used to define how information can be shared between systems, government agencies and departments. The NIEM UML Profile helps you to:

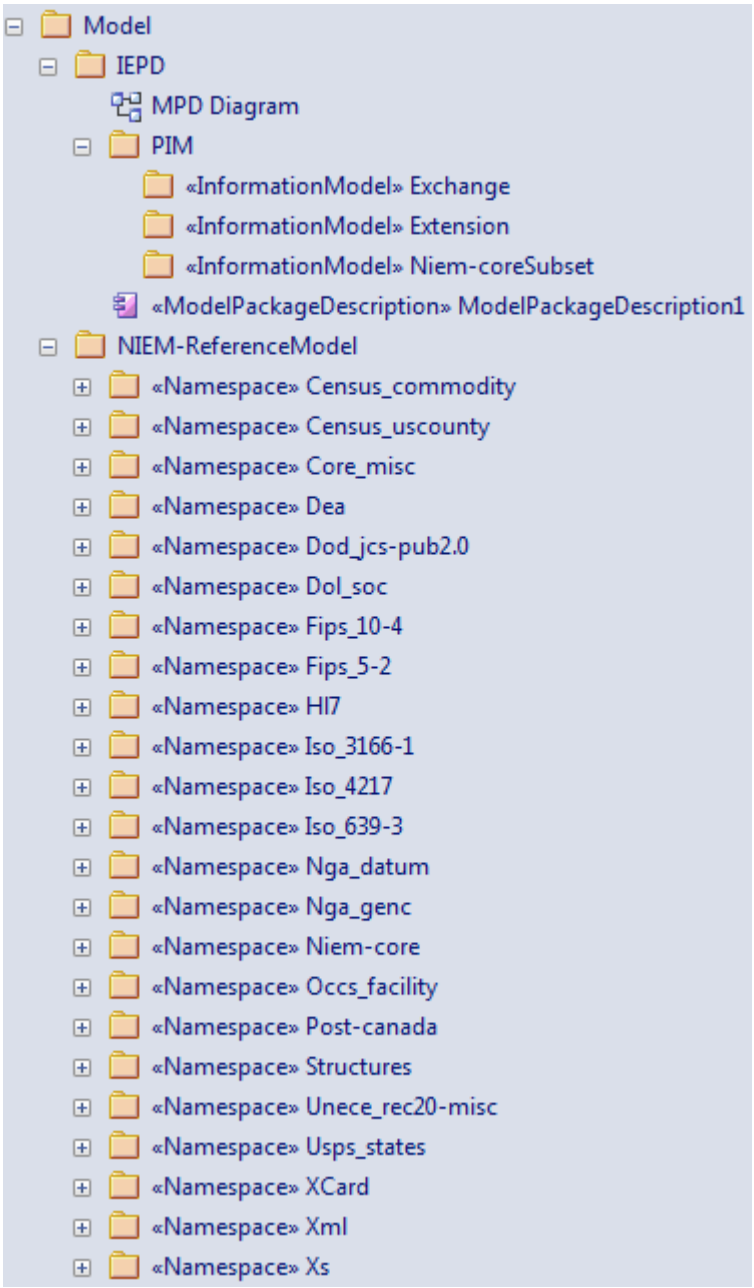
- Create and develop UML-based Information Exchange Package Documentation (IEPD) models, either:
  - Generating an IEPD from an Enterprise Architect Pattern to produce all necessary exchange files, static artifacts, metadata and catalog files, or
  - Using the Schema Composer to generate your own NIEM subset namespaces, automatically detecting inter-dependencies, and using the resulting subset schema to build your own IEPD
- Create PIM, PSM and Model Package Description (MPD) diagrams, using the NIEM Toolbox pages
- Import NIEM Reference Schema into your model
- Generate NIEM Schema from your model

Create a NIEM IEPD Model from a Pattern

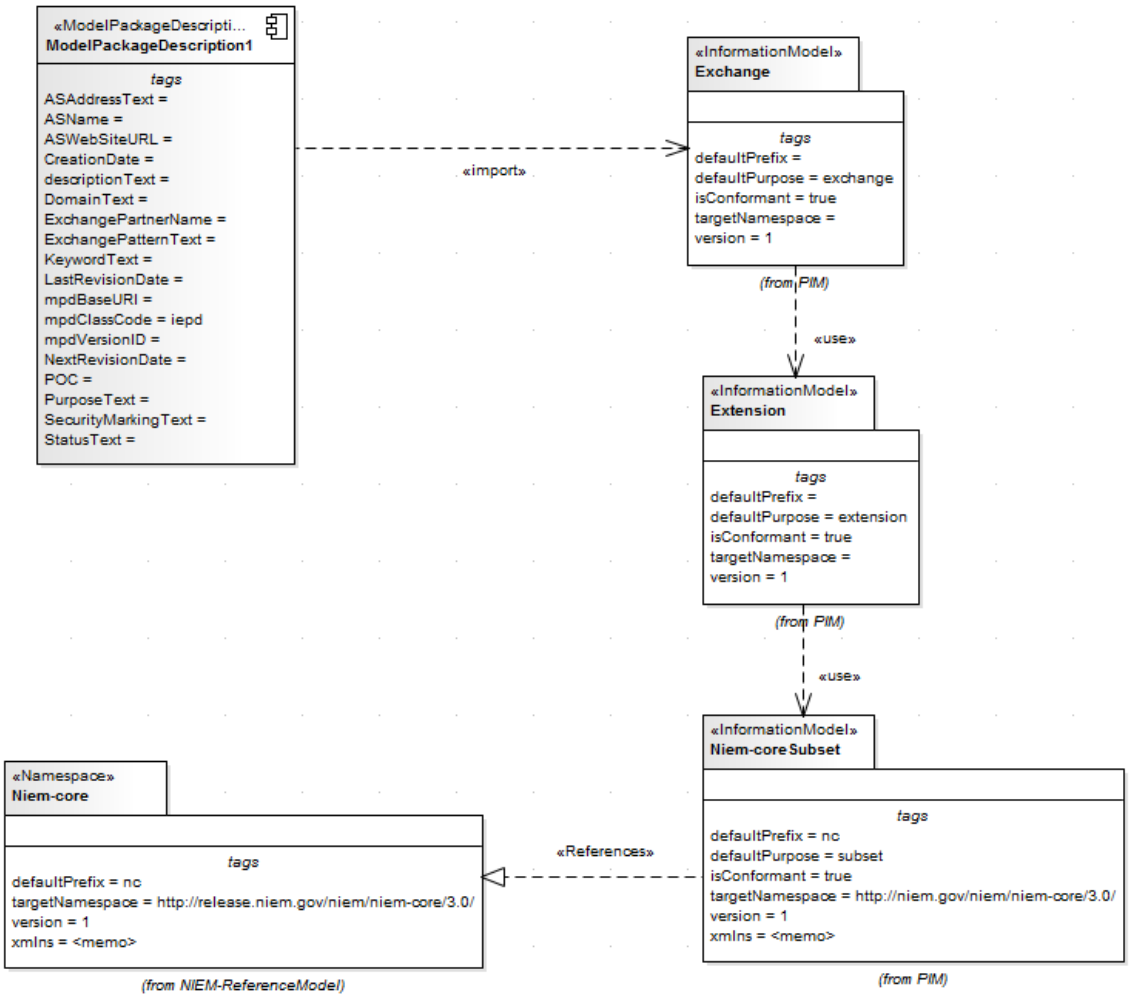
The NIEM UML Profile provides a model Pattern from which to build IEPD models. You can apply this Pattern in your NIEM project, using the Model Wizard.

1. In the Start Page 'Create from Pattern' tab (Model Wizard), select 'Information Exchange > NIEM' in the '<<perspective>>' field.
2. In the 'NIEM 3 and 4' list, scroll through the technologies and click on the required 'NIEM *n.n* Reference Model', then Ctrl+Click on the corresponding 'NIEM *n* IEPD Starter Model'.
3. Click on the Create Model(s) button.

The system generates a new model containing an IEPD Package (itself containing a PIM Package), and a NIEM ReferenceModel Package. The Reference Model can take some time to download.



The IEPD Package contains a top-level Model Package Description (MPD) diagram (as shown), which contains the MPD Component and all the namespaces and files related to it.



The PIM Package consists of all the namespaces and subset namespaces for the IEPD. The relationships between the namespaces and the MPD Component is shown in the MPD diagram. The MPD Component must import at least one namespace for successful NIEM schema generation.

The NIEM ReferenceModel Package includes all the NIEM reference schema models for the selected NIEM version.

NIEM Diagrams

You can also create all of the appropriate diagrams from the

NIEM diagram set and from the corresponding NIEM Diagram Toolbox pages. These diagrams are of three types:

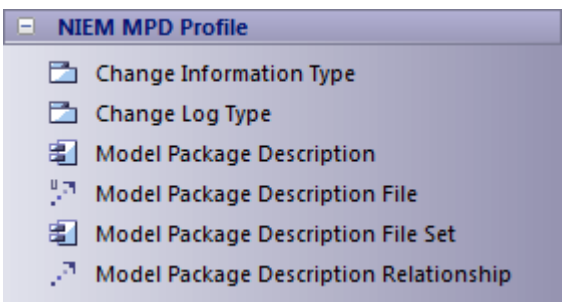
- NIEM Platform Independent Model (PIM) diagram
- NIEM Platform Specific Model (PSM) diagram
- NIEM Model Package Description (MPD) diagram

The templates from which to develop these diagrams are available through the 'New Diagram' dialog.

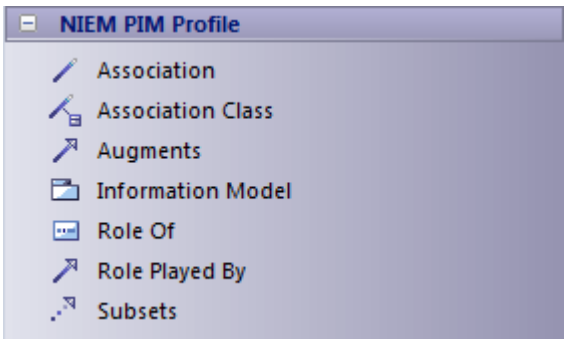
NIEM Toolbox Pages

Each diagram type has its own page of elements and connectors in the Diagram Toolbox. The NIEM UML Profile also provides a page of elements and connectors common to all three diagram types.

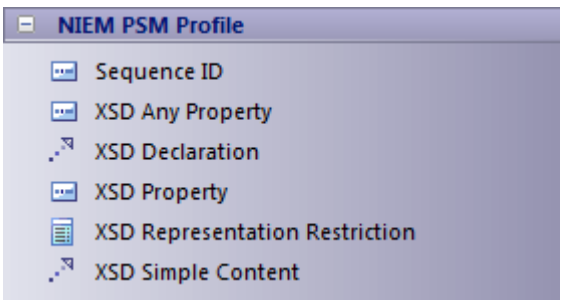
NIEM Model Package Description (MPD) Profile toolbox



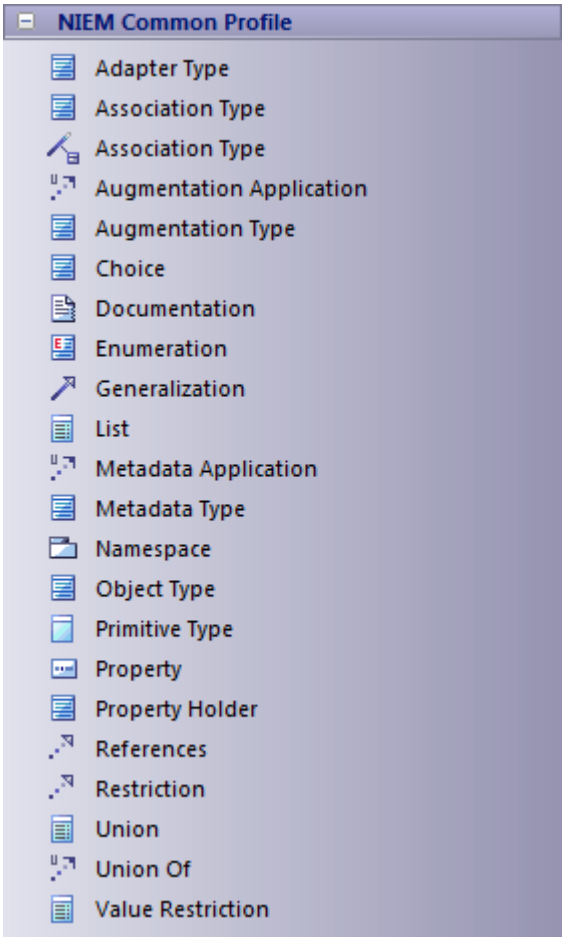
NIEM Platform Independent Model Toolbox



NIEM Platform Specific Toolbox



NIEM Common Profile Toolbox



Import NIEM Reference Schema

Step	Action
1	In the Browser window, right-click on the target Package and select the 'Specialize | NIEM 2.1 | Import NIEM 2.1 Schema' menu option.
2	On the 'Import XML Schema' dialog, in the 'Directory' field, type in or browse for the directory

	containing the schema to be imported, and select the .xsd schema files to import.
3	Under 'Import XSD Elements/Attributes as:' select the 'UML Attributes' radio button.
4	Click on the Import button. The NIEM model for the schema will be imported into the selected Package.

Generate NIEM Subset Namespaces

You can generate a subset namespace using the Enterprise Architect Schema Composer. This requires the NIEM Reference Model to be available in the model, as it is part of the IEPD Model Pattern.

Step	Action
1	Select the 'Develop > Schema Modeling > Schema Composer > Open Schema Composer' ribbon option.
2	Click on the New button to the right of the 'Profile' field. The 'New Message' dialog displays.
3	In the 'Name' field, type the name of the subset, and

	in the 'Namespace' field type the http address of the namespace.
4	In the 'Schema Set' field click on the drop-down arrow and select the 'National Information Exchange Model (NIEM)' option.
5	In the 'Save In' panel, select the 'Model Artifact' radio button.
6	Click on the  icon and use the Navigator to select the namespace/information Model Package in IEPD | PIM, under which to generate the subset.
7	In the Browser window, open the NIEM ReferenceModel Package NIEM-core. Drag the Activity from this Package onto the left hand column of the Schema Composer. The attributes of this element are listed in the middle column of the Schema Composer.
8	Click on the checkbox for each of the attributes you require - for example, ActivityName and ActivityDateRepresentation. The corresponding Classes/NIEM object types are added to the left-hand column, whilst the right-hand column displays them as subset items.

9	Click on the Update button to save the subset profile. The status of the subset items displays against the item name in the left hand column and in the panel at the foot of the column.
10	Click on the Generate button. The 'Schema Export' dialog displays.
11	Select the checkbox against the items to generate, in the 'Technologies' panel. 'NIEM Model Subset' must be selected.
12	Click on the Generate button and, on the 'Find Package' dialog, select the namespace/ information model in which to generate the subset.
13	Click on the OK button, and on the second OK button. The subset model is generated.

NIEM Schema Generation

Once your NIEM IEPD model with its Extension information model, Exchange information model and Subset information model is complete, you can generate schema from it.

Ste	Action

p	
1	Right-click on the MPD Component, which imports the Exchange model, and select the 'Specialize| NIEM 2.1 | Generate NIEM 2.1 Schema' option. The 'Generate NIEM MPD Schemas' dialog displays.
2	In the 'Directory' field, type or browse for the directory path into which to generate the schema.
3	In the 'NIEM Version' field, click on the drop-down arrow and select the NIEM version for which to generate the schema. The static MPD artifacts and common artifacts (Catalog, Metadata) that will be generated are listed in the 'MPD Artifacts' panel, each with its relative path. The 'Namespace Schema(s)' panel shows the schema files that will be generated for the information models.
4	Click on the Generate button. Once the generation has completed successfully, click on the View Schema button to access the catalog file.

National Information Exchange Modeling (NIEM)

National Information Exchange Model (NIEM) provides a common framework that is used to define how information can be shared between systems, government agencies and organizations. Enterprise Architect's NIEM UML Profile helps you to:

- Create and develop UML-based Information Exchange Package Documentation (IEPD) models, by providing starter models, model Patterns and a number of toolboxes for creating IEPD models and schema models
- Generate complete IEPDs from your IEPD model
- Generate NIEM conformant schemas from your information models
- Import NIEM Reference Schema into your model
- Create NIEM subset namespaces, composed from elements of the NIEM Reference Schemas
- Create PIM, PSM and Model Package Description (MPD) diagrams, using the NIEM Toolbox pages

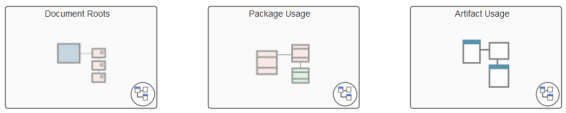
This illustration shows the NIEM 5.0 starter model, a Pattern provided as a part of NIEM in Enterprise Architect. (See the *Creating a NIEM IEPD* Help topic.)

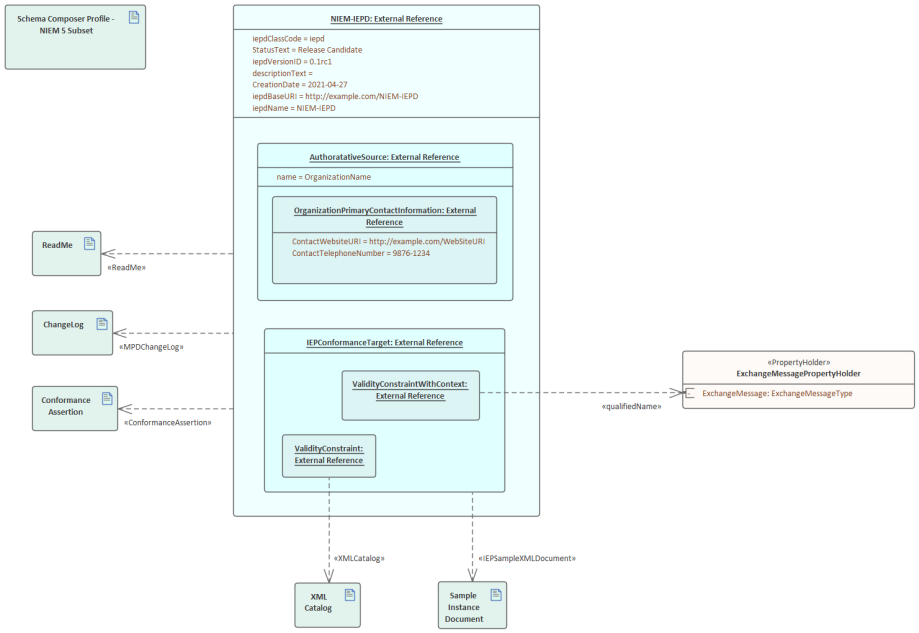
IEPD Overview

This IEPD Overview diagram provides a quick overview of what is contained in the NIEM Starter Model.

In the top-left corner is a Schema Composer profile artifact. You can double-click on this artifact to open the Schema Composer, pre-configured to create a NIEM subset schema.

The other diagrams in the Starter Model, can be opened using the buttons to the right. Each of these diagrams focus in on just one aspect of the IEDP model, giving you room to add additional items without causing the diagram to become cluttered.





UML Profile for NIEM

Enterprise Architect integrates with a UML Profile for NIEM (supporting NIEM 5, 4 and 3), along with a number of model Patterns to help you get started in modeling your NIEM project.

The profile defines a collection of stereotypes for use in building NIEM models. It also defines three different diagram types: Model Package Description (MPD) diagram, Platform Independent Model (PIM) diagram and Platform Specific Model (PSM) diagram. Each of these diagram types has corresponding Diagram Toolbox pages, from which you can select items to add to your model by dropping them onto a diagram.

Access

Use any of the methods outlined here to display the Diagram Toolbox, then click on  to display the 'Find Toolbox Item' dialog and specify 'NIEM 3.0 MPD' (or 'PIM' or 'PSM').

The Diagram Toolbox that corresponds to a particular diagram type becomes active whenever you open a diagram of that type. However, you can also access any Diagram Toolbox at any time, using this method:

- From the top of the Diagram Toolbox, click on  to display the 'Find Toolbox Item' dialog and specify '<profile> <toolbox>'

To reset the Toolbox to the default type for the current diagram, simply close then reopen the diagram.

Ribbon	Design > Diagram > Toolbox
Keyboard Shortcuts	Ctrl+Shift+3
Other	Click the » icon on the Diagram caption bar to display the Diagram Toolbox

Diagram Toolbox Pages

The NIEM Diagram Toolboxes provide quick access to the elements and connectors that you commonly use in a particular type of diagram.

The MPD Diagram Toolbox is grouped into a number of separate pages: Model Patterns, Relationships, File Type Usage and Schema Document Usage. The PIM and PSM diagrams share a common Toolbox page, as well as each having their own specific Toolbox page.

Common Toolbox Items

The NIEM Common Profile consists of stereotypes that are used in both the NIEM PIM Profile and the NIEM PSM Profile.

Icon	Description
AdapterType	A NIEM adapter type is a NIEM object type that adapts external components for use within NIEM.
AssociationType	A NIEM association type establishes a relationship between objects, along with the properties of that relationship.
AssociationType	A NIEM association type establishes a relationship between objects, along with the properties of that relationship.
AugmentationType	A NIEM augmentation type is a complex type that provides a reusable block of data that can be added to object types or association types.
Choice	A Choice Class groups a set of attributes whose values are mutually exclusive.
Documentation	A Documentation comment is the data definition of the element that owns it.
Generalizatio	

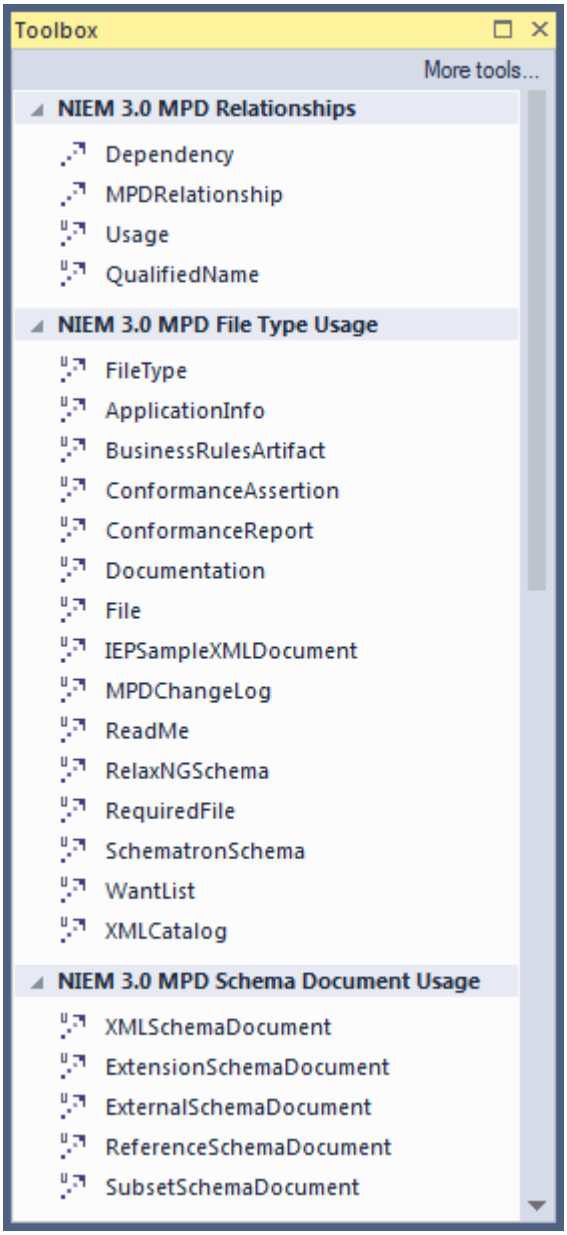
n	A UML Generalization
List	A List is a DataType whose values consist of a finite length (possibly empty) sequence of values of another DataType, which is the item type of the List.
LocalVocabulary	Local vocabulary defines a set of domain specific terms or abbreviations that then can be used in NIEM names and definitions.
LocalTerm	The LocalTerm stereotype defines a domain-specific word, phrase, acronym, or other string of characters used in a LocalVocabulary.
MetadataApplication	The «MetadataApplication» stereotype applies to a Usage between a «MetadataType» Class and either another «MetadataType» Class or a Property. It represents a constraint on a NIEM «MetadataType» that limits the application of the NIEM «MetadataType» to specific schema types or schema elements.
MetadataType	A NIEM metadata type describes data about data, that is, information that is not

	descriptive of objects and their relationships, but is descriptive of the data itself.
Namespace	A Namespace Package represents a NIEM namespace identified by a target namespace URI.
NIEMType	A NIEMType is a Class that represents one of the specific semantic kinds of NIEM complex types (that is, types that can have attributive structure). NIEMType is abstract.
ObjectType	A NIEM object type represents some kind of object: a thing with its own lifespan that has some existence.
PrimitiveType	The NIEM Primitive Type Library defines a predefined set of UML primitive types to be used in NIEM-UML models. To insure integrity and consistency of the type system used at the PIM level with the generation of NIEM compliant schema, the primitive types in this library are based on XML schema primitive types.
Property	

PropertyHolder	A PropertyHolder is a Class holding global Properties that are not the subject of any specific NIEM type. Property declarations of this kind define the object type of the property without restricting its use to a specific type of subject.
References	The References stereotype applies to a Realization between Properties, Classes or Packages. It allows for Properties in one Class to be defined by reference to Properties in another class.
Representation	The NIEM Representation Pattern, allows for a type to contain a representation element, and the various representations for that element type are in the substitution group for that representation element.
Restriction	A Restriction Realization represents a relationship between two type definitions: the first is derived by restriction from the second.
Union	A Union is a DataType whose value space is the union of one or more other

	DataTypes, which are the member types of the Union.
UnionOf	The UnionOf stereotype is applied to a Usage dependency, the client of which must be a Union DataType and the supplier of which must be a DataType that represents a legal union member type. A UnionOf dependency specifies that the supplier DataType is a member type of the client Union.
ValueRestriction	

NIEM 3.0 MPD Toolbox



MPD Toolbox Items

The Model Package Description Profile comprises stereotypes and artifacts that are used to model NIEM MPDs.

Icon	Description

Relationships	
Dependency	A UML Dependency relationship.
MPDRelationship	The ModelPackageDescriptionRelationship stereotype applies to a Dependency that represents a relationship between MPDs or between an MPD and another resource (such as a NIEM specification; as in the case of conforms-to).
Usage	A UML usage relationship
QualifiedName	The <<qualifiedName>> Usage connector is used to specify the Document Element of an IEP. To identify a Document Element of an IEP in UML: • Add an instance of IEPConformanceTargetType to the IEPConformanceTarget slot of the ModelPackageDescription Artifact instance • Add a QualifiedNamesType instance to the ValidityConstraintWithContext slot of the new IEPConformanceTargetType instance • Add a Usage with applied Stereotype

	«qualifiedName» where the client is the new QualifedNamesType instance and the supplier is a Property representing an XSD Element
File Type Usage	
FileType	The <<FileType>> Usage connector is a data type for describing an MPD file artifact. It is also the base type from which many other <<FileType>> Usage connectors are derived.
ApplicationInfo	The <<ApplicationInfo>> connector extends the <<FileType>> usage connector. It is used to specify an MPD artifact that is used by a software tool (for example, import, export, input and output).
BusinessRulesArtifact	The <<BusinessRulesArtifact>> connector extends the <<FileType>> usage connector. It is used to specify an MPD artifact that contains business rules and constraints on exchange content.
Conformance Assertion	The <<ConformanceAssertion>> connector extends the <<FileType>>

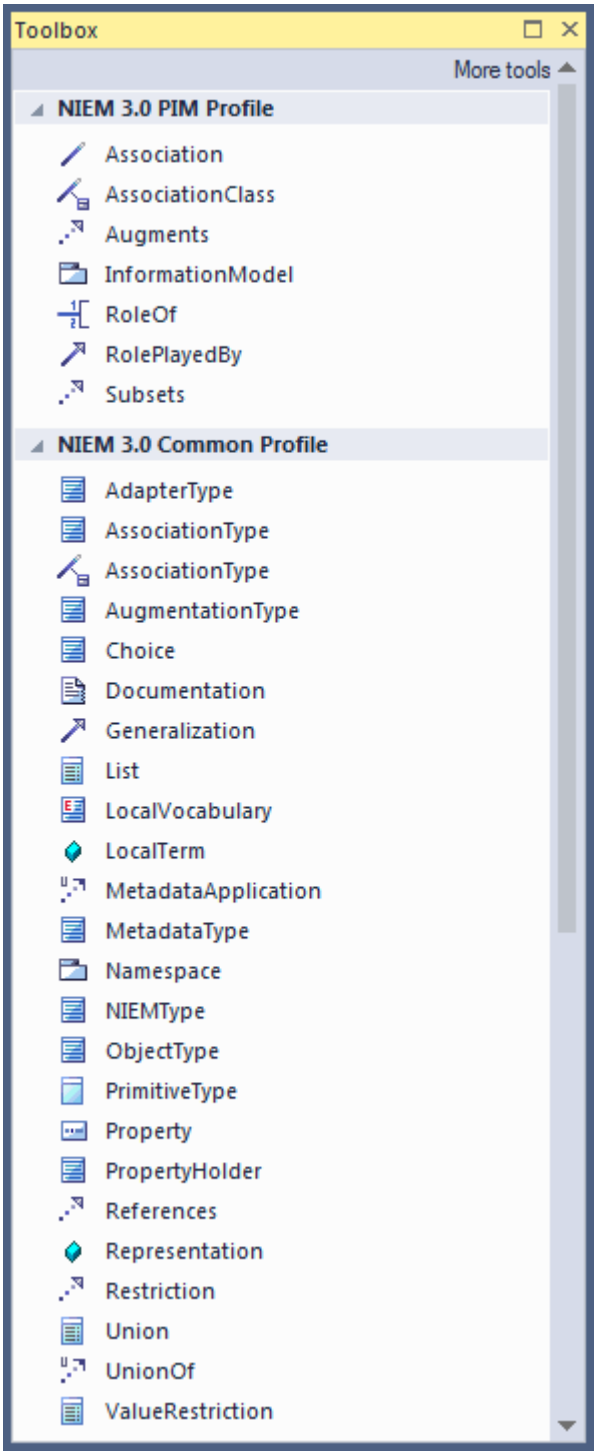
	usage connector. It is used to specify an MPD artifact that represents a declaration that a NIEM IEPD is NIEM-conformant.
Conformance Report	The <<ConformanceReport>> connector extends the <<FileType>> usage connector. It is used to specify an MPD artifact either auto-generated by a NIEM-aware software tool or manually prepared that checks NIEM conformance and/or quality and renders a detailed report of results.
Documentation	The <<Documentation>> connector extends the <<FileType>> usage connector. It is used to specify an MPD artifact that is a form of explanatory documentation.
File	The <<File>> connector extends the <<FileType>> usage connector. It is used to specify a generic electronic file artifact in an MPD; a file stored on a computer system.
IEPSampleXMLDocument	The <<IEPSampleXMLDocument>> connector extends the <<FileType>> usage connector. It is used to specify an example MPD instance XML document

	or IEP artifact.
MPDChangeLog	The <<MPDChangeLog>> connector extends the <<FileType>> usage connector. It is used to specify an MPD artifact that contains a record of the MPD changes.
ReadMe	The <<ReadMe>> connector extends the <<FileType>> usage connector. It is used to specify an MPD read-me artifact.
RelaxNGSchema	The <<RelaxNG>> connector extends the <<FileType>> usage connector. It is used to specify a RelaxNG schema.
RequiredFile	The <<RequiredFile>> connector extends the <<FileType>> usage connector. It is used to specify an MPD file artifact that another artifact depends on and should not be separated from.
SchematronSchema	The <<SchematronSchema>> connector extends the <<FileType>> usage connector. It is used to specify a Schematron schema document.
WantList	The <<WantList>> connector extends the <<FileType>> usage connector. It is used

	to specify an MPD artifact that represents a NIEM schema subset and is used as an import or export for the NIEM SSGT.
XMLCatalog	The <<XMLCatalog>> connector extends the <<FileType>> usage connector. It is used to specify an MPD artifact that is an OASIS XML catalog.
Schema Document Usage	
XMLSchema Document	The <<XMLSchemaDocument>> connector extends the <<FileType>> usage connector. It is used to specify an MPD artifact that is an XML schema document (that is, an XSD that is not necessarily a NIEM subset, extension, or reference schema).
ExtensionSchemaDocument	The <<ExtensionSchemaDocument>> connector extends the <<XMLSchemaDocument>> usage connector. It is used to specify an MPD artifact that is a NIEM extension schema document.
ExternalSche	The <<ExternalSchemaDocument>>

maDocument	connector extends the <<XMLSchemaDocument>> usage connector. It is used to specify an MPD artifact that is a schema document external to NIEM.
ReferenceSchemaDocument	The <<ReferenceSchemaDocument>> connector extends the <<XMLSchemaDocument>> usage connector. It is used to specify an MPD artifact that is a reference schema document (from a release, domain update, or core update).
SubsetSchemaDocument	The <<SubsetSchemaDocument>> connector extends the <<XMLSchemaDocument>> usage connector. It is used to specify an MPD artifact that is a subset schema document.

NIEM 3.0 PIM Toolbox



PIM Toolbox Items

The NIEM PIM Profile comprises stereotypes that are used

in NIEM PIMs but not NIEM PSMs.

Icon	Description
Association	A UML Association.
AssociationClass	A UML Association Class.
Augments	A stereotyped Realization connector, used to specify that one Class (the supplier) Augments another Class (the client).
Information Model	InformationModel is a stereotyped Package that provides a platform-independent perspective on the structure of information to be exchanged in NIEM messages. It represents a NIEM namespace, but can also specify a default purpose, such as subset, exchange or extension.
RoleOf	The RoleOf stereotype is applied to an AssociationEnd to specify the type of the role of the associated property.
RolePlayedBy	A stereotyped Generalization connector specifying that the role played by instances of the general Class will be the

	type of the special Class.
Subsets	The Subsets connector is a stereotyped realization that specifies a subset relationship between a subset client (the derived element) and its reference supplier (the base element).

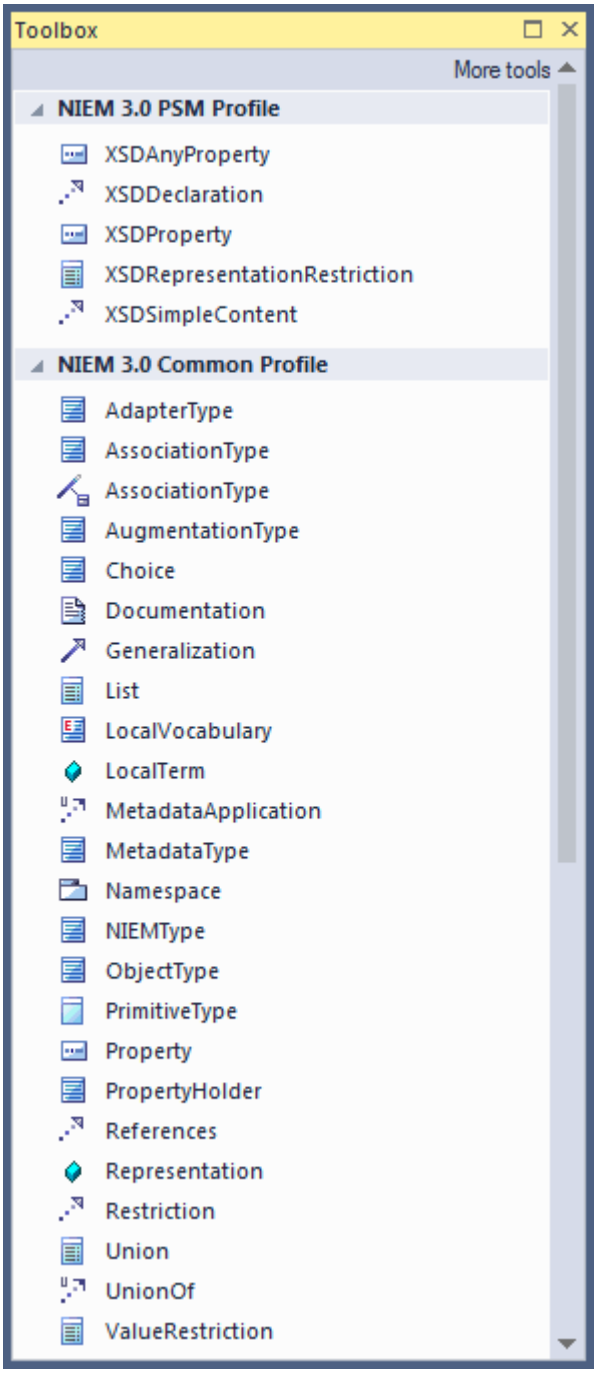
PSM Toolbox Items

The NIEM PSM Profile comprises stereotypes that are used in NIEM PSMs. These stereotypes need not be used with a NIEM PIM, but they can be in order to provide additional platform-specific markup.

Icon	Description
XSDAnyProperty	XSDAnyProperty stereotype represents a property that is unrestricted with respect to its type, which is implemented in XML Schema as the xs:any particle.
XSDDeclaration	The XSDDeclaration stereotype is a specialization of the common References stereotype.
XSDProperty	An XSDProperty Property represents a

	NIEM property, which is implemented in XML Schema as either an attribute declaration and use or an element declaration and particle
XSDRepresentationRestriction	XSDRepresentationRestriction specifies a restriction on the representation in an XML schema of the values of a base DataType.
XSDSimpleContent	The «XSDSimpleContent» stereotype represents a relationship between two type definitions: the first is a complex type definition with simple content, the second is a simple type.

NIEM 3.0 PSM Toolbox


Toolbox
More tools
NIEM 3.0 PSM Profile
XSDAnyProperty
XSDDeclaration
XSDProperty
XSDRepresentationRestriction
XSDSimpleContent
NIEM 3.0 Common Profile
AdapterType
AssociationType
AssociationType
AugmentationType
Choice
Documentation
Generalization
List
LocalVocabulary
LocalTerm
MetadataApplication
MetadataType
Namespace
NIEMType
ObjectType
PrimitiveType
Property
PropertyHolder
References
Representation
Restriction
Union
UnionOf
ValueRestriction

Download the NIEM Reference Model

The NIEM 5 Reference Model is a UML representation of the content of the NIEM 5 Release Package XSD files.

It contains Packages representing NIEM-core, as well as the various Domain schemas included in the NIEM 5 release, their associated Codes lists and other associated Packages. The NIEM 5 reference model is available for download into your Enterprise Architect project, from the Sparx Systems Reusable Asset Server.

Earlier versions of NIEM reference models are also available for download from the Sparx Systems Reusable Asset Server.

Access

Display the Model Wizard (Start Page 'Create from Pattern' tab) using any of the methods outlined here.

In the Model Wizard, select the Information Exchange > NIEM Perspective and then select 'NIEM 3, 4 and 5'.

Select a Reference Model, MPD Types and Starter Model, as required.

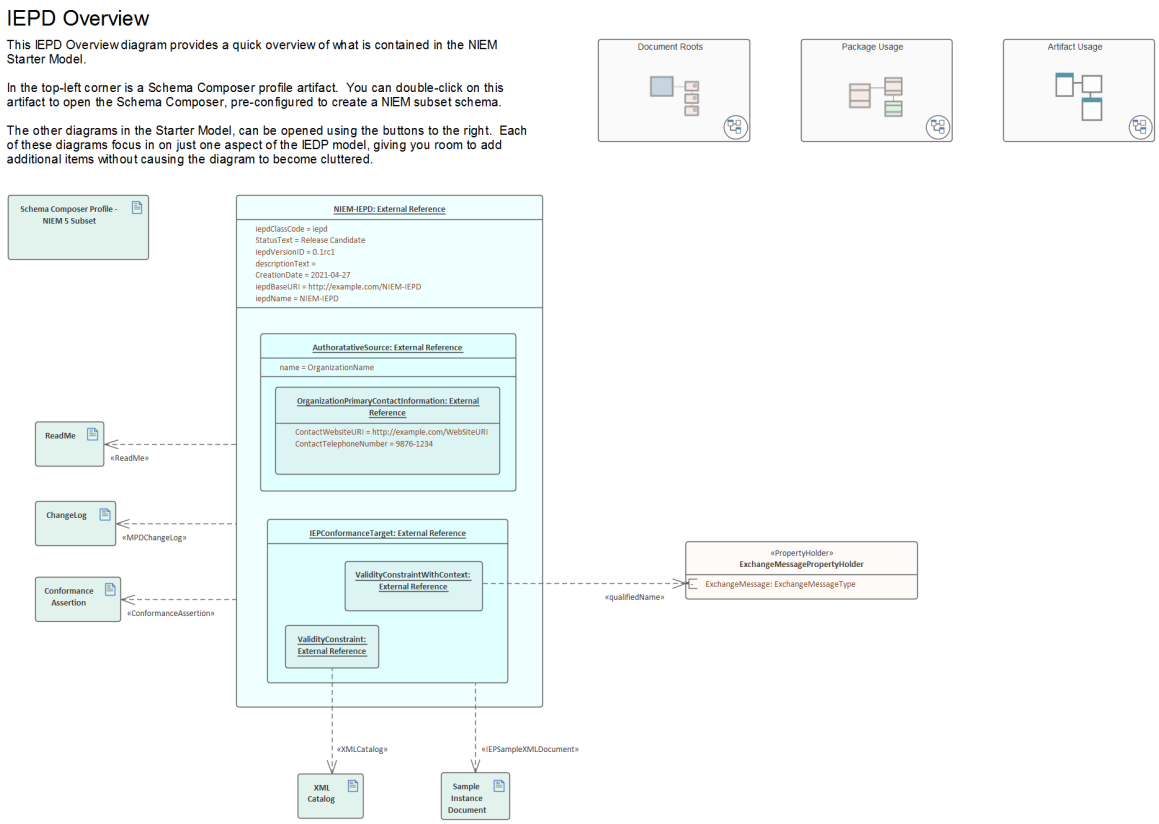
Ribbon	Design > Package > Model Wizard
Context Menu	Right-click on Package | Add a Model Using Wizard

Keyboard Shortcuts	Ctrl+Shift+M
Other	Browser window caption bar menu | New Model from Pattern

Creating a NIEM IEPD

Enterprise Architect's NIEM profile provides a basic IEPD model as a starting point from which you can build your own IEPD model.

You can add the IEPD starter model to your project using the Model Wizard ('Create from Pattern' tab of the Start Page).



The IEPD Starter Model Pattern, available from the Model Wizard.

This topic provides an overview of the steps required to create a new NIEM IEPD model in Enterprise Architect, and to generate an IEPD from that model.

Creating a NIEM IEPD model and generating a NIEM IEPD

Steps	Description
Create a new Enterprise Architect project	Launch Enterprise Architect and create a new project. Click on the Start Page 'Create from Pattern' tab (Model Wizard). Click on the <perspective name> button and select 'Information Exchange | NIEM' and expand the 'NIEM 3, 4 and 5' Perspective. It is essential that your NIEM project contains the NIEM IEPD Types and at least one of the NIEM Reference Models. Select the 'NIEM 5.0 Reference Model' as well as 'NIEM 5 IEPD Types'. Click on the Create Model(s) button to download and import the selected models into your project. Also available in the Model Wizard is a model pattern for a basic NIEM IEPD. This is intended as a starting point for your NIEM project. Optionally, select the 'NIEM 5 IEPD

	Starter Model' and click on the Create Model(s) button.
Create an IEPD model	If you chose not to include the IEPD model in the previous step, you can create your own model now. In the Browser window, create a new Package or (View node) to hold your IEPD model. Within the new Package, create a NIEM IEPD diagram. You can add instances of the types available in the NIEM IEPD Types Package to your diagram (and to your IEPD model), by dragging them onto your IEPD diagram. Use the Browser window to locate the Class types that you require, then press Ctrl as you drag the element into position on your diagram. The system prompts you to choose an action; either: • Place a link to the Class on the diagram, or • Create and add a new instance specification of the Class For the IEPD model, you would generally use Object Instances. To begin with, create an object instance

	of the IEPD Class. (As you will see shortly, you need an instance of the IEPD class to drive the generation of your IEPD.)
Customize your model	Whether you choose to download the IEPD Starter Model, or create your own IEPD model by dragging instances from the Browser window, you must set values for the properties of the Object instances that are appropriate to the model you are creating. This is achieved by setting the run-state properties of the various object instances used in your IEPD model. The Package 'NIEM IEDP Types' contains definitions for the Class 'IEPD', as well as a number of other Classes. These other Classes are referenced as classifiers for attributes of the 'IEPD' Class. Relationships between the various Classes defined in this Package can be viewed on the diagram 'NIEM-UML IEPD Types'. The 'IEPD' Class has a number of attributes that are simple string types, as well as some attributes that are classified by types that are defined within the 'NIEM IEDP Types' Package. When setting the run state values for

	properties with simple types in the IEPD object, you can use the 'Set Run State' command. This can be accessed by right-clicking the Object on a diagram, then choosing 'Features | Set Run State...' (or by pressing Ctrl+Shift+R). Where properties reference other Classes as their types, you cannot simply enter a run state value. Enterprise Architect supports two methods of specifying values for these properties, each method requires creation of an Object instance of the referenced Class. You should create an Object instance of the type corresponding to the property, then either create an Association between the two objects and set a role name for the property being set, or nest the Object as a child within the Object whose properties are being set and name the child Object using the name of the property being set. When associating an Object, the Object's name is not important, but the role name must match the name of the property being set. For example, you might create an Object instance of the type IEPConformanceTargetType and nest it

	within the IEPD Object. In this case, the child Object must be named 'IEPConformanceTarget' to correspond with the attribute of that name in the Class definition. Ensure that the child Object is indeed nested within the parent, by inspecting the hierarchy shown in the Browser window. If using a role name on an Association, create the 'property' Object as a separate (non-nested) Object instance, then create an Association from the 'owner' to the 'object' and finally specify a role name for the target Object. For example, create an Association from the Model Package Description Object to an Object instance of IEPConformanceTargetType. Open the 'Properties' dialog for the Association and name the role of the target as 'IEPConformanceTarget', to correspond to the name of the attribute in the 'IEPD' Class. Again, in this scenario, the name of the Object itself is not important, it could even be anonymous, but the role name must match the name of the attribute whose value you are setting. Note that an IEPD Object might specify many IEPConformanceTargets. You must create an Object instance for each one

	and each one must be named 'IEPConformanceTarget'. Either of these techniques can then be used to set properties within the IEPConformanceTargetType Object. For example, to set the value of the ValidityConstraintWithContext attribute, create an Object instance of the Class ValidityConstraintWithContextType (which might be an instance of the derived type QualifiedNamesType) and either name it and nest it, or associate it and name the role. File use can be modeled by adding Artifact elements to the diagram and linking with the required File Type Usage connector from the toolbox.
Create your data model	This is where you model the data that will be sent in your information exchange message. In NIEM, this is typically modeled within Packages that have the <<InformationModel>> stereotype, representing the different namespaces used in the model. These Packages typically include a NIEM-core Package that is a subset of the NIEM-core Reference Model Package, and two

	extension Packages that extend what is available from NIEM-core, one of which represents the exchange message. Your project might also require subsets of other NIEM schemas, such as those from the Biometrics or EmergencyManagement domains. For further information on creating data models, see the Help topics *Creating a NIEM Data Model* and *Subsetting NIEM with the Schema Composer*.
Generate the IEPD	Your NIEM model does not have to be complete before you generate an IEPD from it. Generating the IEPD can be considered an iterative process. You might perform a generation of just your namespace schemas before you have completed your IEPD, and before you have defined your conformance targets. You might generate with a fully described IEPD instance and conformance targets before you have defined your Information Models. You can continue to update your model and generate your IEPD however you see fit. To generate your IEPD, select the IEPD instance specification, either on the diagram or in the Browser window.

	Right-click on the IEPD instance to open the context menu or go to the 'Specialize' ribbon and select the 'Technologies > NIEM > Generate NIEM Schema' option. The Generate NIEM IEPD Schemas window opens. This window lists the Namespace Schemas that are used in your model, and you can select which of these to generate. You can also choose which of the NIEM infrastructure schemas to include in the generation. In this window you can also set the root directory for generation of the output files. Once you have made your selection and specified the output folder, click on the Generate button to commence generation of the IEPD. For detailed information on the Generate NIEM IEPD Schemas window, see the Help topic *NIEM IEPD Generation*.

Notes

- ALL projects containing NIEM models must include the

NIEM IEPD Types Package downloaded via the Start Page 'Create from Pattern' tab (Model Wizard); the IEPD instance is central to your NIEM model
The instance and the relationships to <<InformationModel>> Packages and other artifacts are used to drive the IEPD Generation; without an IEPD instance in your model, you will not be able to generate an IEPD

- Usually, you must have at least one of the NIEM Reference Models imported into your project; the reference models contain UML representations of the NIEM-core reference schema, as well as the many domain specific reference schemas, which must be available in your project if you intend to create subset schemas using Enterprise Architect's Schema Composer

NIEM IEPD Generation

Generating the IEPD can be considered an iterative process. You do not have to wait until your NIEM model is complete before you generate an IEPD from it.

Your NIEM 'IEPD OVerview' diagram should contain an instance specification of an IEPD. The IEPD instance and its relationships to Conformance Target instances as well as other artifacts, is a representation of the IEPD catalog. When you generate an IEPD from your model, Enterprise Architect will generate a catalog file and other artifacts, based on the items in your IEPD model. It will also generate NIEM schemas for the <<InformationModel>> packages referenced by your model. The result will be a collection of files output into the directory you specify for the generation process.

Steps for generating an IEPD

Step	Action
1	Your NIEM IEPD diagram should contain an Instance Specification of an IEPD. Select the IEPD instance, either on the diagram or in the Browser window.

2	From the 'Specialize' ribbon, choose the option 'Technologies > NIEM > Generate NIEM Schema'. The 'Generate NIEM IEPD Schemas' dialog displays.
3	In the 'Directory' field, type or browse for the directory path into which to generate the IEPD.
4	The 'NIEM Version' field defaults to '5.0'. If generating a NIEM 3 or 4 IEPD, set this field to the appropriate value. The 'IEPD Artifacts' panel lists the static IEPD Artifacts and common Artifacts (such as Structures and Catalog) used in this model, each with its relative path. Select or deselect the checkboxes beside these items to generate or skip these items. The 'Namespace Schema(s)' panel shows the schema files that will be generated for the information models. Select or deselect the checkbox beside a Namespace Schema to generate or skip that schema. Select a Namespace Schema to display Package details for that schema.
5	Click on the Generate button. Once the generation has completed successfully, clicking the View Schema button opens Windows

	Explorer, showing the contents of the output directory used for generation. If the Catalog Artifact has been deselected, then clicking the View Schema button will open an editor to view the schema file associated with the currently selected Namespace Package.

Notes

- The output location of the schema file generated for a Package is specified by the 'pathURI' tag value on the Usage connector that relates the Package to the IEPD instance specification; default values are set by the Schema Composer when the subset Packages are created, but the values can be overridden by the user

Creating a NIEM Data Model

One of the underlying principles behind NIEM is re-use of a common reference vocabulary - a predefined set of data elements and definitions used to define information exchanges. To this end, one of the core tasks in building a NIEM data model is creating a subset of the NIEM reference schema. The goal is to model as much of your data exchange as is practical, by re-using types and elements that are already defined in the NIEM reference model.

A NIEM data model usually consists of a number of Packages with the <<InformationModel>> stereotype applied.

Typically, a model will have one Package representing a NIEM-core subset schema, some other Packages representing subsets of particular Domain schemas, and one or more Packages representing extension schemas. The extension schema Packages provide those elements required by the model that are not available from the NIEM Reference Model. Often, the root element of the exchange message is separated from more general elements, and modeled in an extension schema Package dedicated to the specific exchange.

Steps for Creating a NIEM Data Model

Step	Detail

Import the NIEM Reference Model	Many of the activities involved in creating NIEM models, rely on using the NIEM Reference Model. If you haven't already done so, import the Reference Model into your Enterprise Architect project before proceeding further. For more information, see the Help topic Download the NIEM Reference Model.
Create a Subset of the NIEM-core Reference Package	There are a number of reasons for creating subsets of the NIEM namespace schemas when creating NIEM IEPDs, but the two most important reasons are: • The reference schemas are very large; subsetting produces much smaller schema files that, in turn, leads to faster validation of the schemas • Elements within the reference schemas are very loosely constrained; the subsetting process allows modelers to impose much tighter constraints, such as restricting cardinality and allowed values, to more closely reflect actual business requirements In Enterprise Architect, the subsetting process is performed using the Schema Composer. The Schema Composer allows the

	modeler to select the subset of required Classes from the source Package and, for each of the selected Classes, select a subset of required attributes. The selected Classes with their reduced attribute sets are then copied to a target Package. Most often the source Package will be the NIEM-core namespace Package from the NIEM Reference Model. In this case, the target Package will also be a namespace Package named 'NIEM-core', but it will be part of your NIEM IEPD model. Other namespace Packages from the Reference Model, such as the Domain Packages, can also be subsetted in the same way. Use Enterprise Architect's Schema Composer tool to copy a subset of the NIEM-core reference Package to the NIEM-core subset Package that is part of your IEPD model. The goal is to model as much of your data exchange as is practical, by re-using types and elements that are already defined in the NIEM-core Reference Model. In cases where your model will also make use of NIEM Domain Packages, this subsetting process should be repeated for each domain Package that you use.

	For more information, see the Help topic *Subsetting NIEM with the Schema Composer*.
Create Extension Packages	When creating a NIEM data model, the aim is to model as much of your data exchange as possible using types and elements from the NIEM Reference Model. What cannot be modeled by re-using existing NIEM elements is then modeled in 'extension' namespace Packages, by creating new types and elements using elements from the NIEM-UML profiles, with all types ultimately deriving from XML schema primitive types. Both the NIEM Starter Model (from the Model Wizard) and the IEPD Starter Model Pattern (from the Diagram Toolbox) provide <<InformationModel>> Packages in which to model the various schemas. Using PIM diagrams within these Packages, you can build models of your different schemas, by adding elements from the Diagram Toolbox. It is suggested that you use the diagram in the 'exchange' Package, to assemble the high-level model of your exchange, using

	types and elements from other schema Packages as required. Most IEPDs require extension schemas to define specific types and properties that are unique to the data exchange being defined. However, the NIEM model does not define specific message types or structures for assembling all of the objects in an exchange. It is therefore up to the creator of the IEPD to write an extension schema that declares the root element and the basic structure of the messages. The root element of the exchange brings together all of the objects and associations defined in the exchange. Whilst you are not required to create a separate schema to declare the root element and basic structure of the message, it can be beneficial to separate message-specific extensions into an 'exchange' schema and more generic extensions into 'extension' schemas. Exchange schemas contain definitions that are unique to a message type or group of message types. This generally includes only the root element and its type and possibly some structural elements that form the basic framework

	of the message. Organizing schema elements into 'exchange' and generic 'extension' groupings also offers the possibility of sharing the more generic schema across multiple IEPDs, whereas the 'exchange' schema is usually specific to one particular IEPD. You can also have multiple 'exchange' schemas in order to represent different message types or groups of different message types.

Subsetting NIEM with the Schema Composer

Enterprise Architect's Schema Composer is a tool that can greatly simplify the process of creating subsets from the NIEM Reference Model namespace Packages.

Access

Use either of the methods outlined here to display the Schema Composer window and then display the 'New Model Transform' dialog,

Enter a name for the new model transform, then from the 'Schema Set' drop-down list choose 'National Information Exchange Model (NIEM)'.

Save the profile as a Model Artifact within a suitable Package in your project (the root Package of your IEPD is suitable - then the Artifact will be easy to find).

Ribbon	Develop > Schema Modeling > Schema Composer > Open Schema Composer : New > Model Transform

Creating a Subset Model

NIEM experts suggest that a good first step is to create a UML model of your XML exchange, as it allows you to capture your business requirements without being unduly influenced by how things are done in NIEM.

Once you have a first draft of a UML model for your exchange, you can then begin to re-create that model using NIEM.

Initially, finding appropriate types and properties within the NIEM Reference Model can seem to be an impossible task. This will become much easier as you gain experience and familiarity with the content of the NIEM model.

Most of the NIEM types that you will commonly use, such as PersonType, OrganizationType, DocumentType or ActivityType, have numerous attributes, of which you will generally require only a few. This is where subsetting becomes useful.

If you are trying to model a person, using their name, address and birthdate, you would choose PersonType and AddressType from NIEM-core. From those types, select only the properties that you require for your model.

Where the selected properties reference other types, those types will be automatically added to the Schema Composer.

When you 'generate' your subset, Enterprise Architect creates the target schema Packages required by the subset, then copies the selected types with their reduced attribute sets into the target Packages.

Further Refining Your Subset

Once you have created your subset, you can further refine it by making adjustments to the cardinalities of the properties within the types or by restricting the allowed values of the properties.

To adjust the cardinality or restrict the permissible values of a property, select that property in the center pane of the Schema Composer, then right-click and choose 'Restrict this property'. The 'Property Restrictions' dialog is displayed, where you can adjust the cardinality or apply restrictions to the property as required.

Click on 'Update' to save the changes to your model transform profile, then click on 'Generate' to regenerate the subset model with the restrictions applied.

NIEM subsetting is often an iterative process. Using the saved model transform profile, you can reload, update and regenerate your subset as you require, throughout the various stages of IEPD development.

Subsetting NIEM Using the Schema Composer

Ste	Action

p	
1	Open the Schema Composer. (See *Access: Ribbon*)
2	Create a new Schema Composer profile. Click on the New button and select 'Model Transform'. In the dialog that opens, specify a name for the profile and select 'NIEM' in the 'Schema Set' field. (The 'Namespace' field on this dialog is not used for NIEM, as NIEM uses Tagged Values on its Model Packages to specify namespaces.) Choose a location to save your new profile, then click on the OK button.
3	In the Browser window, locate the required types PersonType and AddressType, in the NIEM-core Package of the Reference Model. Drag and drop the required types from the Browser window onto the 'Classes' pane of the Schema Composer.
4	Now select one of the types, say PersonType, in the Schema Composer's 'Classes' pane. The full list of attributes for PersonType is shown in the 'Attributes' pane.
5	Use the checkboxes in the 'Attributes' list to select

	the attributes of 'PersonType' to use in your exchange model. In this case, select the checkboxes for 'PersonBirthDate' and 'PersonName'. As you select these attributes, the Schema Composer automatically adds the types 'DateType' and 'PersonNameType' to the list of Classes, as these types are referenced by the attributes you just selected.
6	Now select 'DateType' in the 'Classes' pane. 'DateType' has four attributes, DateAccuracyAbstract, DateAugmentationPoint, DateMarginOfErrorDuration and DateRepresentation. The first three of these attributes are date metadata - they do not hold a date value. The fourth, DateRepresentation, is an abstract attribute, so it does not directly hold date values either. It is used as a placeholder for the attribute that will ultimately hold the date value. The NIEM model commonly uses XML Schema abstract elements and substitution groups. The abstract elements add some complexity to the creation of a subset, because you are required to add the abstract element, as well as those elements that will be substituted in place of the abstract element. For example, most date-related types contain the abstract element nc:DateRepresentation that can be substituted by nc:Date, nc:DateTime, and so on.

7	Select the attribute DateType.DateRepresentation. You will notice that another type, DateRepresentationPropertyHolder has been added to the 'Classes' list.
8	Select DateRepresentationPropertyHolder in the 'Classes' list. The untyped attribute DateRepresentation is known as the 'head' of a substitution group. This attribute must be selected in the client of the substitution, DateType, as well as in the supplier of the substitution, DateRepresentationPropertyHolder. The attribute that is the head of the substitution group is pre-selected for you, so you only need to select the attribute that will eventually be substituted for DateRepresentation in DateType. Select the attribute Date:date - it will be used as the DateRepresentation that will actually hold a data value. Where substitution groups are involved, it is a common mistake to simply add the abstract element without also adding the substitutable element from the related PropertyHolder type.
9	Repeat the process for the PersonName attribute, by selecting PersonGivenName, PersonMiddleName and PersonSurName from the PersonNameType class.

10	To save your current selection of Classes and attributes to the profile you are creating, click on the Update button. This updates the profile with your current selection, allowing it to be reloaded at a later date if you need to perform further work on it. This facilitates an iterative process of creating the subset Package.
11	Now click on the 'Generate' option. Choose 'NIEM Model Subset' in the 'Schema Export' dialog and click on the Generate button. Navigate to the Package hierarchy containing the IEPD that you are building. Select the parent Package that will contain the subset packages, then click on the OK button.
12	The Classes you have selected in the Schema Composer will be copied to the target Packages, with just the subset of attributes that you have selected.

Notes

- Please read through each of the walk-through examples - each one contains important information
- The Schema Composer functionality that supports NIEM development, assists in creating Subset Schemas; it does

not assist in producing Extension Schemas

Walk Through Examples

If you are new to using the Schema Composer for NIEM, please take the time to read through these examples. Each example contains important information that will help to ensure that your models use valid NIEM subsets, which will ultimately produce valid XML schema files.

Example 1: Adding Classes and Selecting Attributes

This 'walk-through' example demonstrates how to use Enterprise Architect's Schema Composer to perform basic operations of adding classes and selecting attributes to be included in a NIEM subset package.

Step	Description
1	Open an Enterprise Architect project that contains the NIEM 5.0 Reference Model and also the NIEM IEPD Types. If you don't have such a project, then open a new project and load the Reference Model and IEPD Types, using the Model Wizard (Start Page 'Create from Pattern' tab).
2	Using the Start Page 'Create from Pattern' tab (Model

	Wizard), add a fresh copy of the NIEM 5 IEPD Starter Model to your project. You should rename the object instance 'NIEM-IEPD' to something more meaningful. When generating the IEPD, the name of this object instance is used to name the root folder in which the IEDP is created. If you wish, rename the Package 'NIEM 5 Starter Model' to something more appropriate as well.
3	The starter model contains a Schema Composer artifact named 'Schema Composer profile - NIEM 5 subset'. Locate this artifact in the Browser window, then double-click on it. This will open the Schema Composer and load the profile 'Schema Composer profile - NIEM 5 subset'. The lower part of the Schema Composer contains three columns. From left to right, they are labelled 'Classes', 'Attributes' and 'Schema'.
4	Using the Browser window, locate the Package 'niem-core' in the NIEM 5.0 Reference Model. Within that Package, locate the Class 'AircraftType'. Drag and drop 'AircraftType' onto the left-hand column of the Schema Composer (labelled 'Classes'). You will notice that Classes 'ConveyanceType' and 'ItemType' are added automatically to the list of Classes. 'ItemType' and 'ConveyanceType' are supertypes

	from which 'AircraftType' is derived.
5	Select AircraftType in the 'Classes' column. You will notice that the center column, 'Attributes', displays the full list of attributes belonging to this Class. The attributes of the parent Classes are also listed. To include an attribute in the subset schema, you simply place a checkmark beside it. (You should choose only attributes of the Class that is currently selected in the 'Classes' list. If you require attributes of a parent Class, select that Class, then select its attributes.) Place a checkmark beside AircraftTailIdentification. The type of AircraftTailIdentification is IdentificationType. Notice that IdentificationType has been added to the list of Classes. Enterprise Architect automatically adds to the 'Classes' list, those classifiers that are referenced as types of the attributes you select.
6	Select the Class IdentificationType in the left-hand column of the Schema Composer. In the center column, place a check mark beside the attribute IdentificationID. The type of IdentificationID is 'string'. The type 'string' is a Primitive type - it is not added to the list of Classes.

7	Now, select the Class ConveyanceType in the left-hand column of the Schema Composer. Place a check mark beside the attribute ConveyanceMotorizedIndicator. The type 'boolean' is a Primitive type - it is not added to the list of Classes.
8	Select the Class ItemType in the left-hand column of the Schema Composer. Place check marks beside the attributes ItemMakeName, ItemModelName and ItemModelYearDate. The types ProperNameTextType and TextType are automatically added to the list of Classes. TextType is the base Class for ProperNameTextType.
9	Click on the Update button to save the selected Classes and attributes to the profile, then click on the Generate button. In the window that opens, select 'NIEM Model Subset', then click on the Generate button. You will be prompted to select a Package within which the subset model will be created. Typically, you would choose the Package that is the parent of the Exchange schema Package. In the starter model, the exchange Package is named 'IEPD-Exchange' and its parent Package is named 'NIEM 5 Starter Model', although you might have renamed these

	earlier in step 2. Select the Package 'NIEM 5 Starter Model', then click on the OK button. Note: When creating more complex models, your subset might include Classes from several different <<InformationModel>> Packages. Enterprise Architect's Schema Composer will automatically create the required target Packages and copy the Classes you are subsetting into the target Packages whose Tagged Value 'targetNamespace' matches that of the source Package from which the original Class was drawn. The subset <<InformationModel>> Packages will be created as children of the Package you choose as the generation target.
10	Once the generation is complete, expand the target <<InformationModel>> packages. You will see the classes you selected with their reduced sets of attributes.

Example 2: Using Association Types

This 'walk-through' example demonstrates how to use Enterprise Architect's Schema Composer to add association types and the types they reference, to your NIEM subset package.

Step	Description
1	Open an Enterprise Architect project that contains the NIEM 5.0 Reference Model and also the NIEM IEPD Types. If you don't have such a project, then open a new project and load the Reference Model and IEPD Types using the Model Wizard (Start Page 'Create from Pattern' tab).
2	Using the Model Wizard (Start Page 'Create from Pattern' tab) add a fresh copy of the NIEM 5 IEPD Starter Model to your project. You should rename the object instance 'NIEM-IEPD' to something more meaningful. When generating the IEPD, the name of this object instance is used to name the root folder in which the IEDP is created. If you wish, rename the Package 'NIEM 5 Starter Model' to something more appropriate as well.
3	The starter model contains a Schema Composer artifact named 'Schema Composer profile - NIEM 5 subset'. Locate this artifact in the Browser window, then double-click on it. This will open the Schema Composer and load the profile 'Schema Composer profile - NIEM 5 subset'. The lower part of the Schema Composer contains three columns. From left to right, they are labeled

	'Classes', 'Attributes' and 'Schema'.
4	Using the Browser window, locate the Package 'niem-core' in the NIEM 5.0 Reference Model. Within that Package, locate the Class 'PersonLocationAssociationType'. Drag and drop 'PersonLocationAssociationType' onto the left-hand column of the Schema Composer (labelled 'Classes'). You will notice that the center column 'Attributes' displays PersonLocationAssociationType.Attributes and also PersonLocationAssociationType.Associations. Place check marks beside both of the associations, Location and Person. The types LocationType and PersonType are automatically added to the Schema Composer's 'Classes' list.
5	The Class PersonLocationAssociationType is derived from the supertype 'nc:AssocationType', but in this case the supertype is not automatically added to the Classes list. If you want to include any attributes of the supertype 'nc:AssociationType' in your generated subset, you must add 'nc:AssociationType' to the Schema Composer's Class list manually, then select the required attributes.

	If you don't want to specifically include attributes of 'nc:AssociationType', then there is no need to add it to the Classes list. When the schema file is eventually generated from the subset Package, Enterprise Architect will generate an element and type definition for 'nc:AssociationType' if and when it is required, even if it is not explicitly modeled.
6	Click on the Update button, then click on the Generate button. In the window that opens, select 'NIEM Model Subset', then click on the Generate button. You will be prompted to select a Package within which the subset model will be created. Typically, you would choose the Package that is the parent of the Exchange schema Package. In the starter model, the exchange Package is named 'IEPD-Exchange' and its parent Package is named 'NIEM 5 Starter Model', although you might have renamed these earlier in step 2. Select the Package 'NIEM 5 Starter Model', then click on the OK button.
7	Locate the <<InformationModel>> Package named 'niem-core' within the subset model. Create a NIEM PIM diagram within this Package, then drag and drop the three Classes in this Package onto the diagram. You will notice that the properties 'Person'

	and 'Location' are modeled as AssociationEnds on the Associations between PersonLocationAssociationType and the types PersonType and LocationType.

Example 3: Using Substitution Groups and Property Holders

This 'walk-through' example demonstrates how to use Enterprise Architect's Schema Composer to correctly add substitution groups and property holders to your NIEM subset Package.

Step	Description
1	Open an Enterprise Architect project that contains the NIEM 5.0 Reference Model and also the NIEM IEPD Types. If you don't have such a project, then open a new project and load the Reference Model and IEPD Types using the Model Wizard.
2	Using the Start Page 'Create from Pattern' tab (Model Wizard), add a fresh copy of the NIEM 5 IEPD Starter Model to your project. You should rename the object instance 'NIEM-IEPD'

	to something more meaningful. When generating the IEPD, the name of this object instance is used to name the root folder in which the IEDP is created. If you wish, rename the Package 'NIEM 5 Starter Model' to something more appropriate as well.
3	The starter model contains a Schema Composer artifact named 'Schema Composer profile - NIEM 5 subset'. Locate this artifact in the Browser window, then double-click on it. This will open the Schema Composer and load the profile 'Schema Composer profile - NIEM 5 subset'. The lower part of the Schema Composer contains three columns. From left to right, they are labeled 'Classes', 'Attributes' and 'Schema'.
4	Using the Browser window, locate the Package 'niem-core' in the NIEM 5.0 Reference Model. Within that Package, locate the Class 'AircraftType'. Drag and drop 'AircraftType' onto the left-hand column of the Schema Composer (labeled 'Classes'). You will notice that the Classes ConveyanceType and ItemType are added automatically to the list of Classes. ItemType and ConveyanceType are supertypes from which AircraftType is derived.
5	Select the Class 'AircraftType' in the left-hand

	column of the Schema Composer. In the center column, place a check mark beside the attribute AircraftWingColorAbstract (notice that this attribute has no type specified). The Class AircraftWingColorAbstractPropertyHolder is automatically added to the list of Classes.
6	Select the Class 'AircraftWingColorAbstractPropertyHolder' in the left-hand column. Notice that this Class also has an attribute named 'AircraftWingColorAbstract' that has no type specified. This attribute is pre-selected for you - it should remain selected. Simply place a check mark beside AircraftWingColorText.
7	In this case, the attribute AircraftWingColorAbstract is the head of the substitution group and provides the connection between the client Class AircraftType and the supplier Class AircraftWingColorAbstractPropertyHolder. AircraftWingColorText is the actual attribute (of type TextType) that will be added to AircraftType.
8	Some PropertyHolder types will have several attributes - the substitution group head, plus a number of others. The attribute that is the head of the substitution group must always be selected in both

	the client and the supplier Classes. Enterprise Architect pre-selects this attribute for you in the supplier Class (the PropertyHolder). You then only need to select the attribute(s) from the supplier that you want to substitute in place of the head of the substitution group.

Example NIEM Schema

This page provides an overview of defining a new NIEM compliant schema, from start to finish.

The framework Packages that are required for NIEM modeling have been described in previous topics. The Model Wizard (Start Page 'Create from Pattern' tab) also provides a Package that acts as a convenient starting point for defining your IEPD. When this is imported to your model you will find diagrams containing instances of the IEPD types, with the run-state set to show the core properties that you are most likely to need to set.

This section describes the process of taking the sample IEPD from the Pattern, and creating a 'Hello World' style message, where a request is made for a personalized message based on a facial image. The response will be the identity of the pictured person and a personalized message for them.

Import NIEM framework Packages

Modeling with NIEM in Enterprise Architect starts with the standard types defined by the NIEM Technical Architecture Committee and the Object Management Group NIEM-UML specification, as described here. These are available from our Reusable Asset Server and the Model Patterns wizard.

To import these into your model:

- Open the Start Page 'Create from Pattern' tab (the Model Wizard)
- Find the Perspective 'NIEM 3, 4 and 5'
- Select the Packages required for your model
- Click on the Create Model(s) button to import the selected patterns into your model.

Note:

- All NIEM 5 models require the NIEM IEPD Types Package as well as one of the NIEM Reference Model Packages
- All NIEM 3 and 4 models require the NIEM MPD Types Package as well as one of the NIEM Reference Model Packages
- All NIEM 2.1 models require the NIEM 2.1 Reference Model Package but not an MPD Types Package, as the NIEM 2.1 MPD elements are available from the NIEM 2.1 MPD Diagram Toolbox

Component	Details
NIEM Framework	The power of NIEM comes primarily from the extensive library of types that you can use to build your own schemas. Enterprise Architect provides complete NIEM frameworks for NIEM 5, as well as all versions of NIEM 3 and NIEM 4. These frameworks are all available from the Start Page 'Create from Pattern' tab (the Model Wizard).

	This tutorial is using the NIEM 5.0 framework, so select that pattern for import.
IEPD Types from NIEM-UML	A user-defined NIEM schema is built around an IEPD that defines, for consumers of the schema, how to use the various XSD files that are included and what message types are being defined. When modeling in UML, an IEPD is created using instances of a number of Classes defined in the UML Profile. Enterprise Architect provides these Classes in a Package that is available from the Start Page 'Create from Pattern' tab (the Model Wizard). All NIEM 5 models will require these IEPD types, so select the 'NIEM 5 IEPD Types' pattern for import.

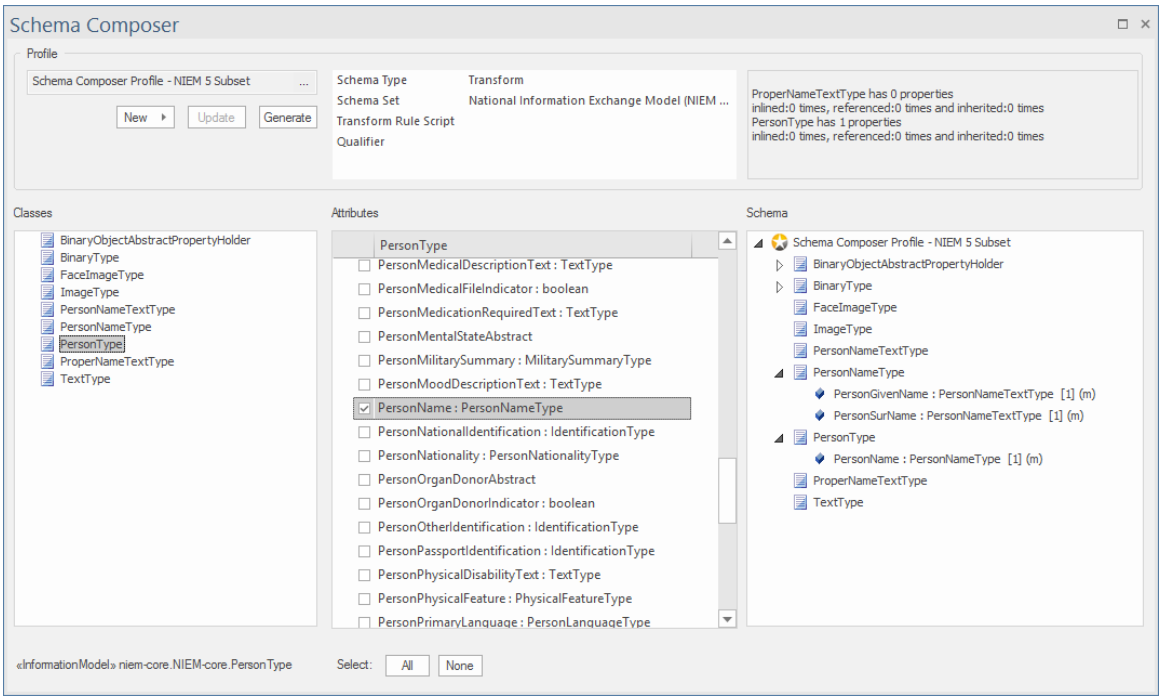
Subset NIEM Namespaces

The Starter Model pattern includes a Schema Composer Artifact to use for specifying a subset. Double-click on it to open the Schema Composer and begin the subsetting process.

We want our request message to send a facial image to be used for facial recognition. To do that, we need to subset the appropriate types from the Biometrics Package. Start by locating the type FaceImageType in the Domains\Biom Package within the NIEM 5.0 Reference Model. Drag this type into the Schema Composer. The super-types from which this type inherits are automatically added to the Schema Composer.

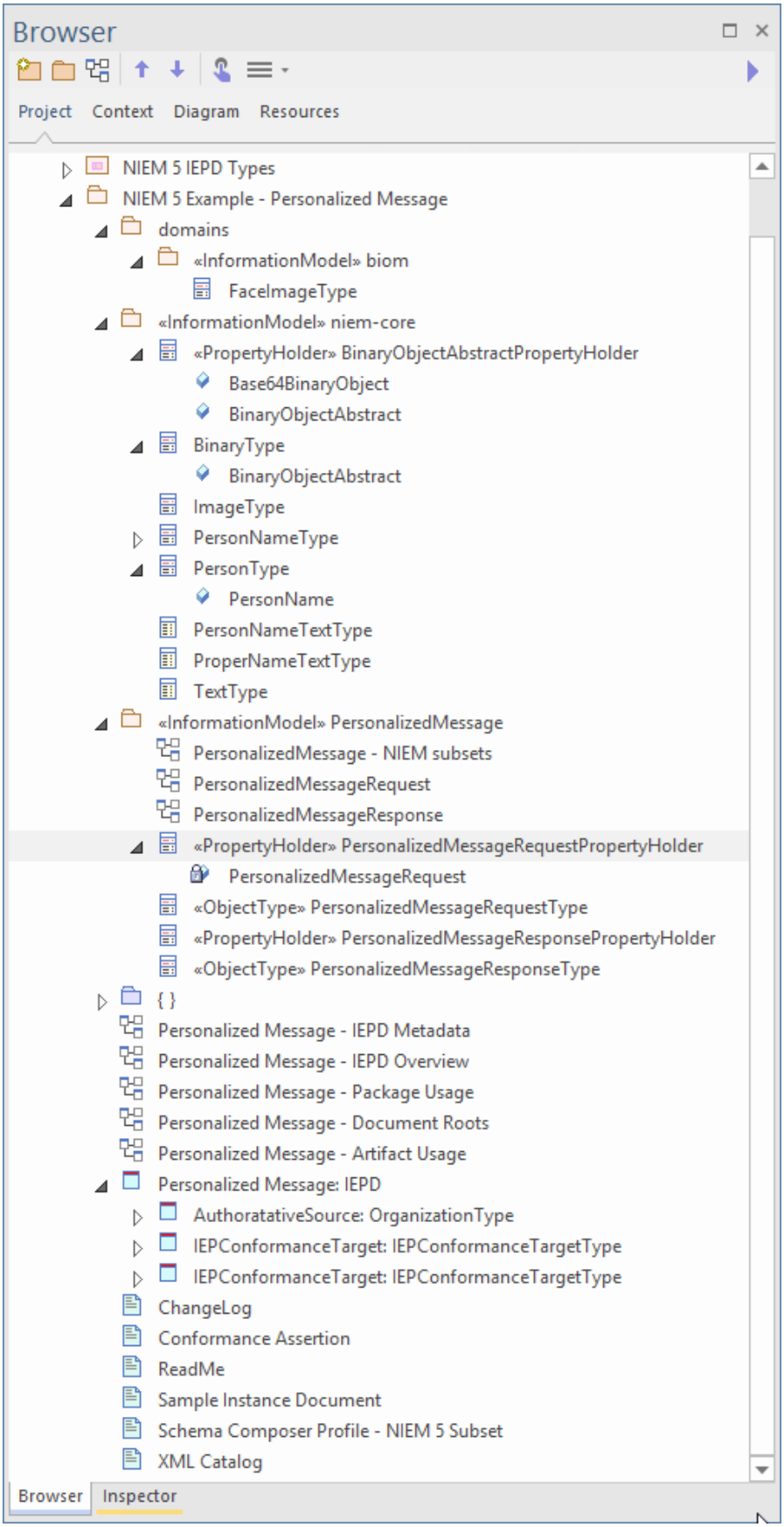
Our response message requires PersonType from the 'niem-core' Package. Drag this type onto the Schema Composer as well.

This image shows the selection of a subset of the types and properties across a number of namespaces within the NIEM 5.0 Reference Model:

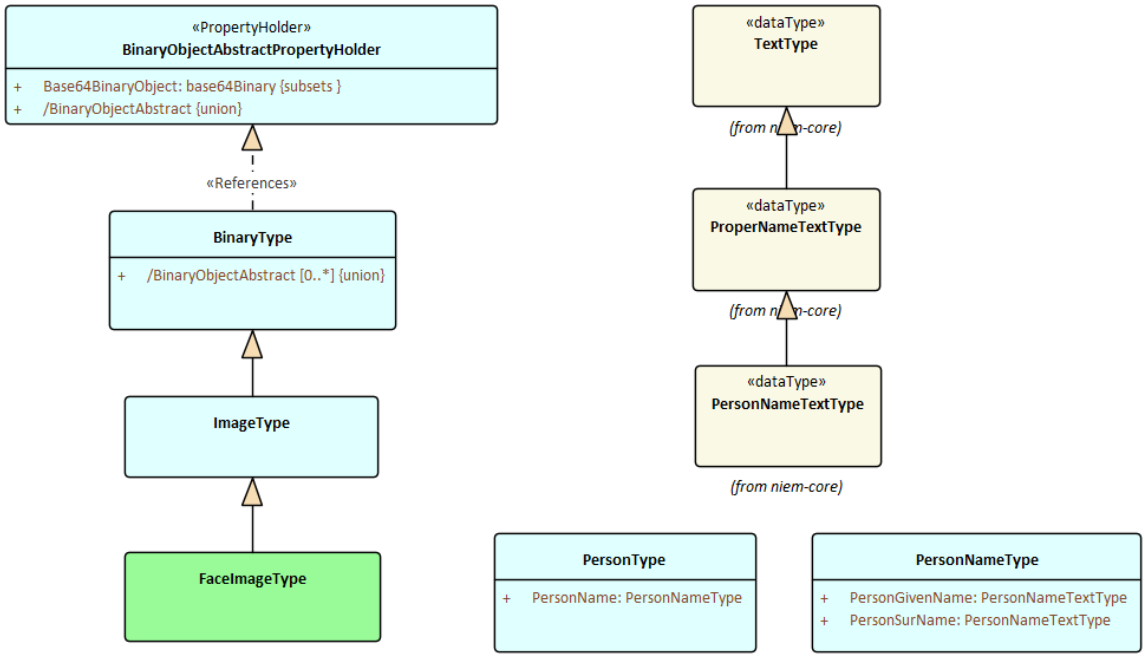


Once the required types are selected, you can generate the subset. When prompted, select the **parent** Package into which the subset namespace Packages will be generated. After generation, the Classes in the subset Packages should

resemble this:

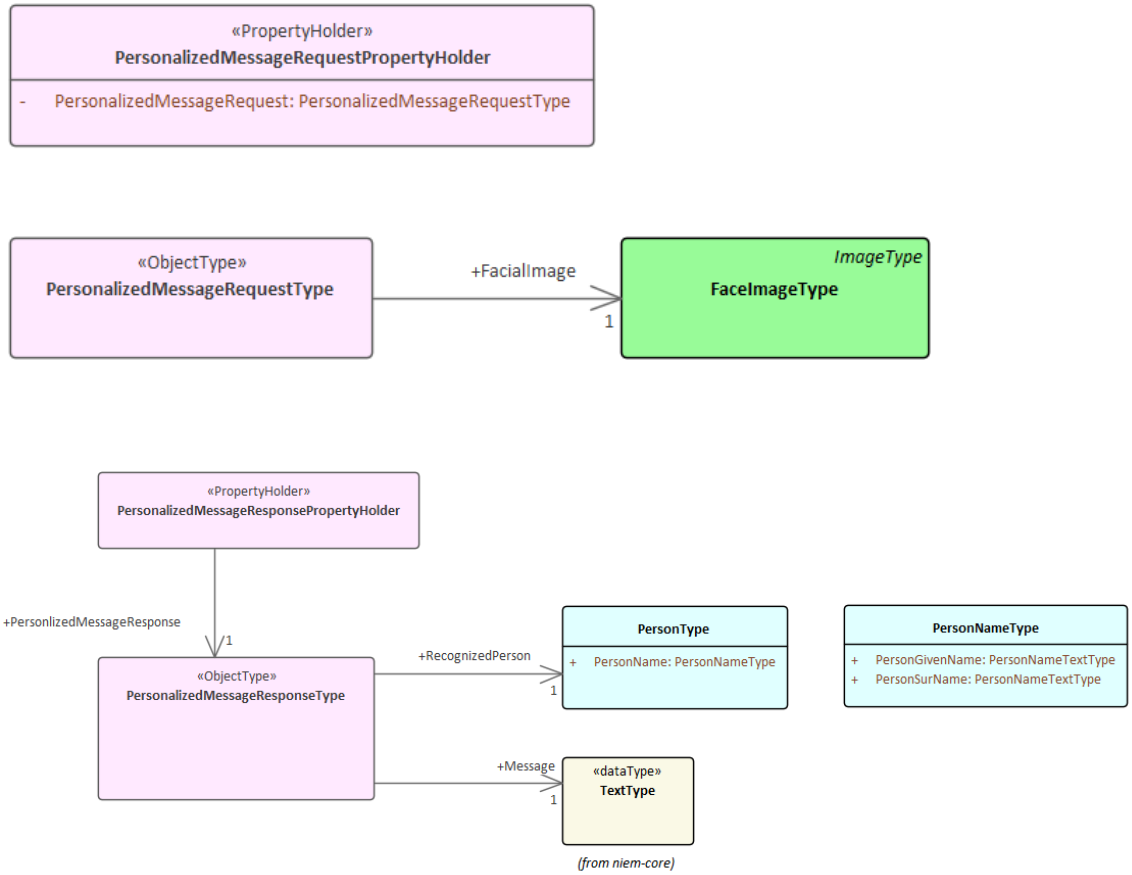


We can now create a NIEM PIM diagram and place all our subset Classes on that diagram, to produce something that resembles this:



Create extension types

We will be defining two messages, a request and a response. For each of these messages, we need to define the document root elements. These will be modeled as extensions to the NIEM schema. Now that we have defined our subset Packages we can define these document roots. Because we are only creating two simple document types, all that is needed is a PropertyHolder and ObjectType for each message. The ObjectTypes link to the types we've selected from the NIEM framework, to describe the contents of each message as shown:



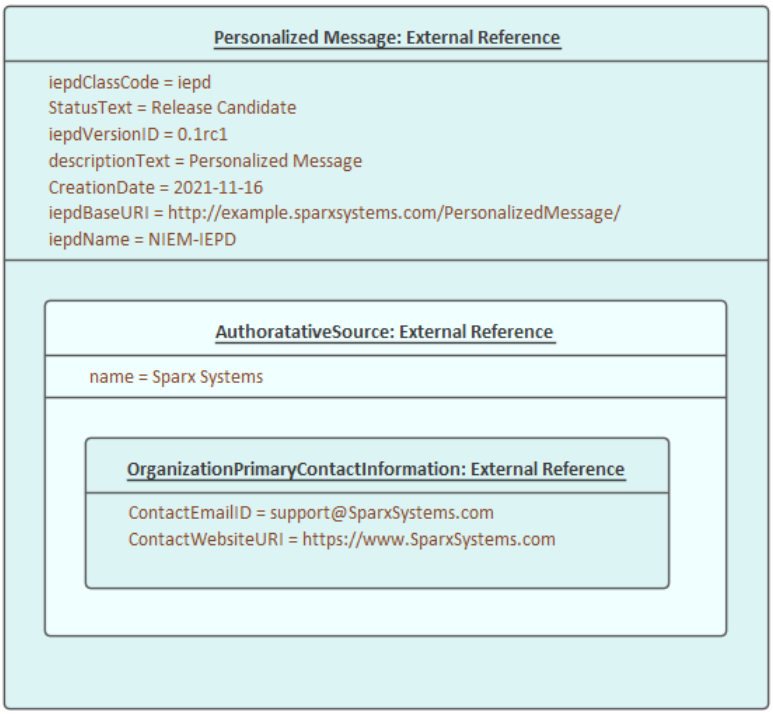
Customize the IEPD

The instance of the IEPD class carries information that identifies the IEPD and the links between it and the various other model artifacts determine what is generated (and where it is generated to) when generating the schema files and catalog files. The main points are described below.

Component	Description
IEPD Metadata	The top level object in the Pattern is an instance of the IEPD class. The name of the IEPD is the name of the Object itself.

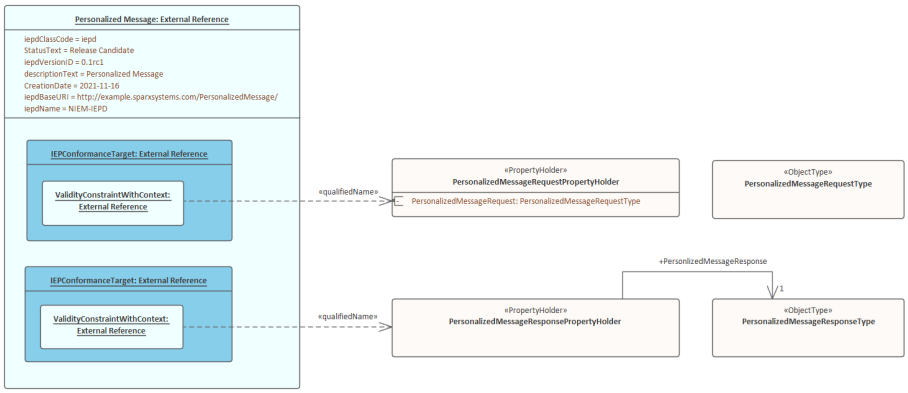
All other properties are in the Run-State of the object.

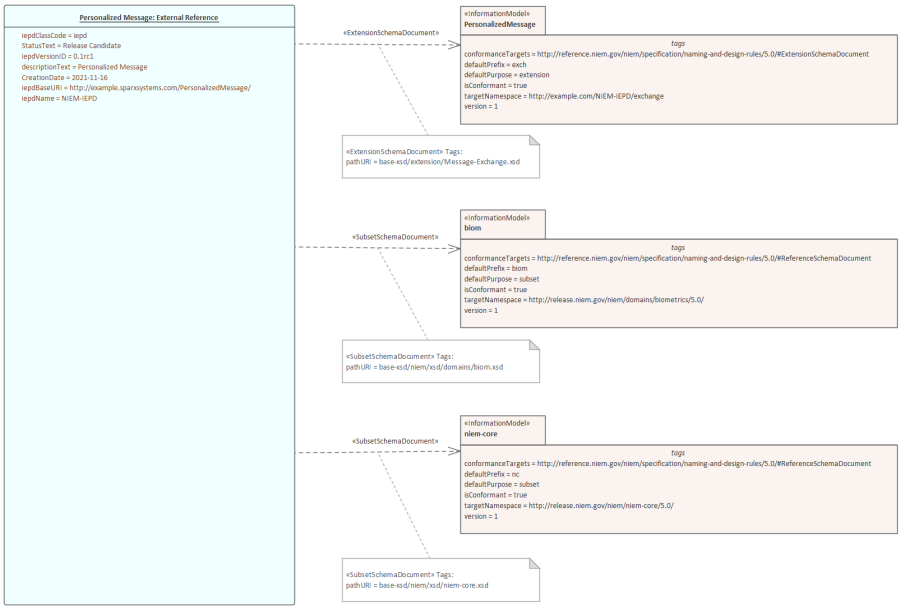
This figure shows how the IEPD might look after providing real information.



NIEM-UML recommends the last section of iepdBaseURI matches the name of the IEPD, and specifies that the iepdVersionID will be appended to the iepdBaseURI to produce the generated iepdURI. This example follows that convention.

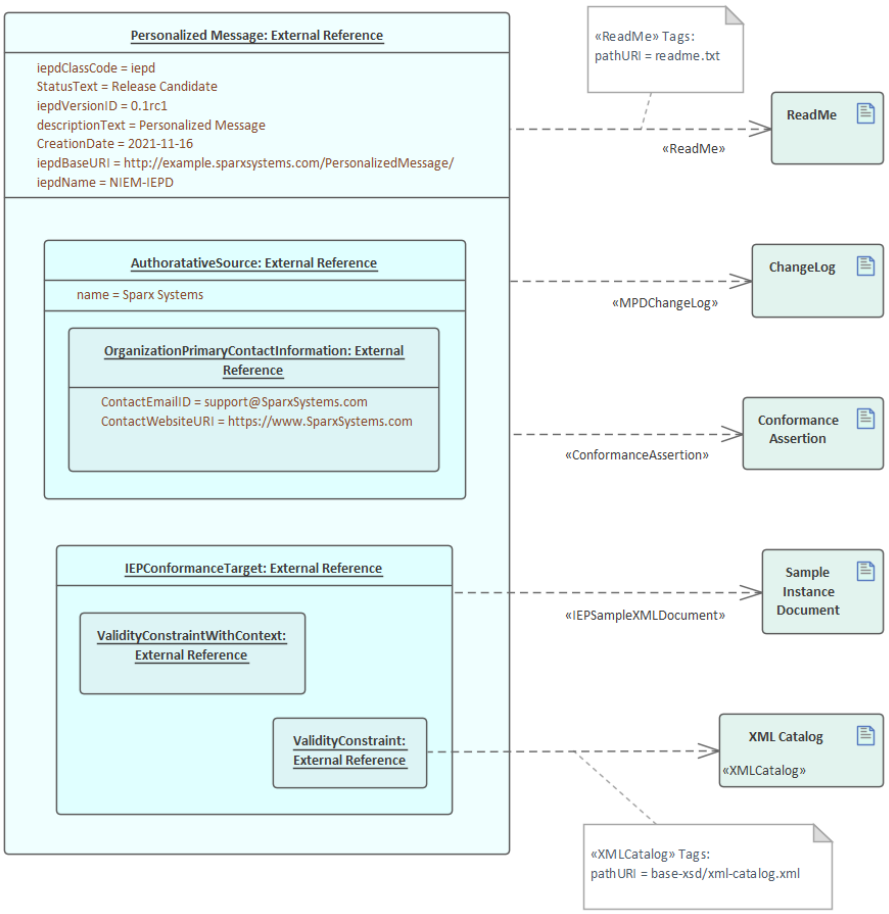
The Pattern defaults the value of iepdClassCode to 'iepd'. This means that the IEPD is intended to represent an Information Exchange Package Document (IEPD). This is the most common type of IEPD, and it is what we

	want to create, so it has been left with the default value.
Defined Document Types	An IEPD is expected to define one or more document types. Each one will be an instance of IEPConformanceTargetType named 'IEPConformanceTarget'. The provided model Pattern already includes one of these, but we need a second one as shown here:  Note the instances of QualifiedNamesType, with the qualifiedName relationship to a PropertyHolder. This specifies that the top level of the document being described will be an element from one of the contained attributes. The section *Create Extension Packages* in the topic *Creating a NIEM Data Model* describes how this is defined.
Package	The relationships connecting the IEPD

Usage	instance to the Information Models specify which schema files are to be generated with this IEPD. In this example we use types from two different NIEM namespaces. The sub-setting process has created an InformationModel Package for each, where the Namespace Tagged Values match the original, and the purpose is set to subset. We also created an extension Package where we defined our own types and how the NIEM types will be used. This figure shows how this looks:  The relationships used also specify how the Package is used and the relative path to the schema defined by that Package.
Additional Files	All IEPD Packages are expected by NIEM to contain - at a minimum - a

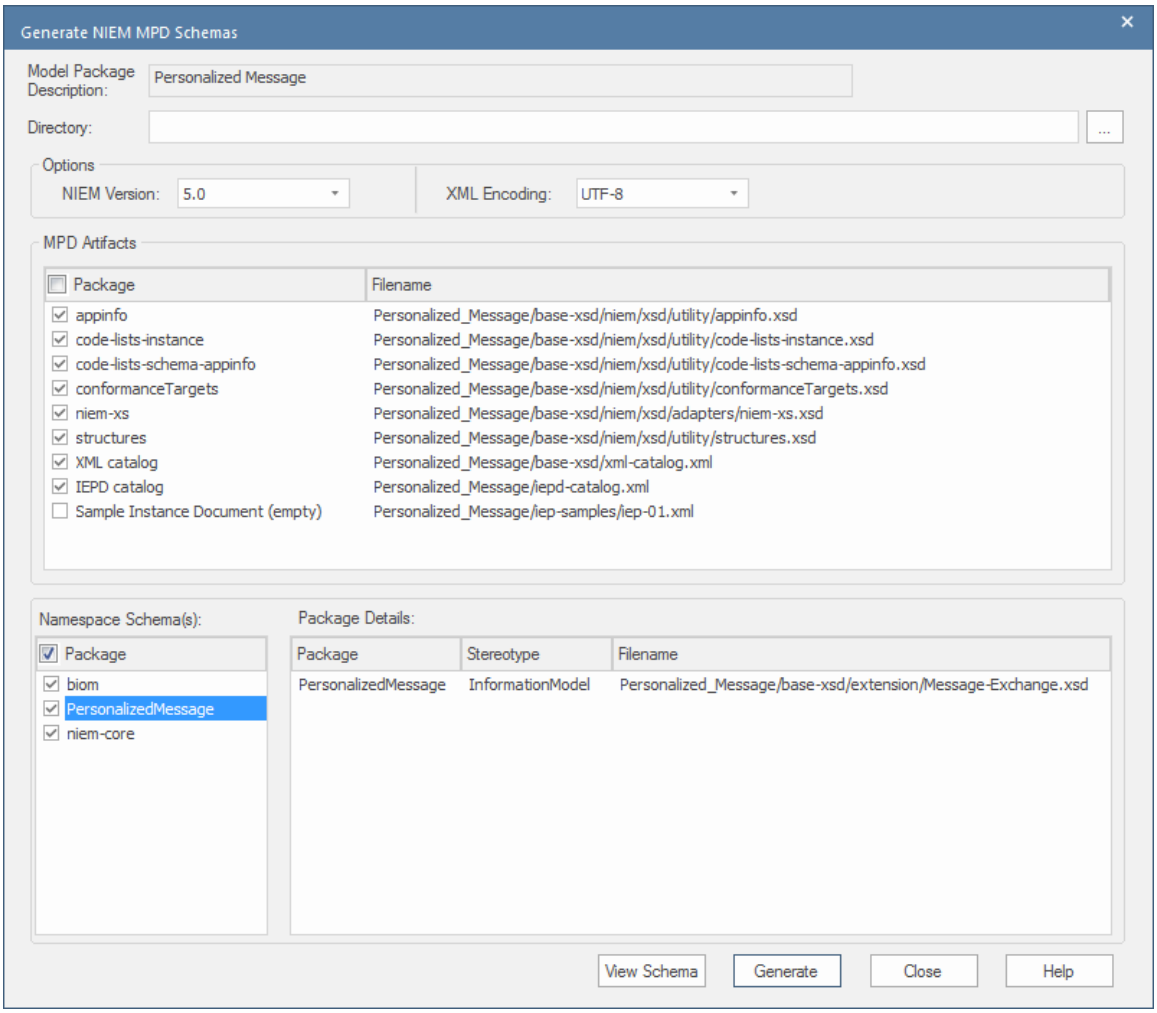
change log and a readme, but there are several other types of Artifact that are also supported. In Enterprise Architect, each is defined using a stereotyped relationship to an Artifact. As with the Package use, the relationship specifies where the file will be located.

In this image a ReadMe, ChangeLog and a sample document are described for each of the document types. This will add information about those files to the target catalog file. The files will not be created by Enterprise Architect, and their content is beyond the scope of this tutorial.



Generate IEPD

Right-click on the IEPD object instance.
Select 'Specialize | NIEM | Generate NIEM Schema'.



The dialog shows the standard NIEM artifacts and the list of linked namespaces that can be generated as schemas. Set the target directory and click on the Generate button to create the modeled IEPD.

Import NIEM XML Schema

As well as generating NIEM schema in Enterprise Architect, you can import (reverse engineer) an external NIEM-specific XML Schema file into your Enterprise Architect project as a UML model.

Access

Ribbon	Specialize > Technologies > NIEM > Import NIEM Schema Specialize > Technologies > NIEM 2.1 > Import NIEM 2.1 Schema
Context Menu	In the Browser window: Right-click Package | Specialize | NIEM | Import NIEM Schema Right-click Package | Specialize | NIEM 2.1 | Import NIEM 2.1 Schema

Import a NIEM specific XML Schema

Option	Action

Package	Displays the name of the currently-selected Package in the Browser window, as the Package into which to import the NIEM schema. You can verify that you are using the appropriate Package by clicking on the  button and checking the 'Navigator' dialog; select a different Package if necessary.
Directory	Click on the  button and browse for the directory containing the source NIEM Schema file(s). Click on each file to import, and then click on the browser Open button.
Selected File(s)	Lists the XML Schema file(s) selected for import.
Import referenced XML Schema(s)	Select this checkbox if you want to import any other XML Schema that is referenced by any of the files listed in the 'Selected File(s)' field.
Skip Schema if Namespace in Model	Select this checkbox if you want to skip importing an XML Schema if it already exists in the model. Enterprise Architect will use the Schema

	namespace and name to determine if it exists in the Model.
Create Diagram for XML Schema(s)	Select this checkbox to create a Class diagram (a NIEM PIM diagram) under each imported Namespace Package.
Layout created Diagram	(Enabled only if the 'Create Diagram for XML Schema(s)' option is selected.) Select this checkbox to automatically lay out the created Class diagram(s).
Import	Click on this button to start the import process. The progress of the import is reported in the 'NIEM Importer' tab of the System Output window. A message box also displays to indicate when the import is complete; click on the OK button to clear the message.
Close	Click on this button to close the 'Schema Importer' dialog.
Help	Click on this button to display this Help page.

Notes

- Enterprise Architect uses the *schemaLocation* attribute in the XSD Import and XSD Include elements of an XML Schema, to determine the dependencies between the files; this attribute must be set to a valid file path (and not a URL) for the dependent XML Schema(s) to be imported correctly
- The 'Create Diagram for XML Schema(s)' option generates a diagram for each imported schema file, but displays the diagrams only for the schema files specifically selected by the user; it does not display the diagram for a referenced schema file
- If you import large schema files, it is recommended that you deselect the 'Create Diagram for XML Schema(s)' option, as this considerably increases the time taken by the import

The Requirements Model

Manage all Aspects of Requirements from Elicitation to Validation and Reuse

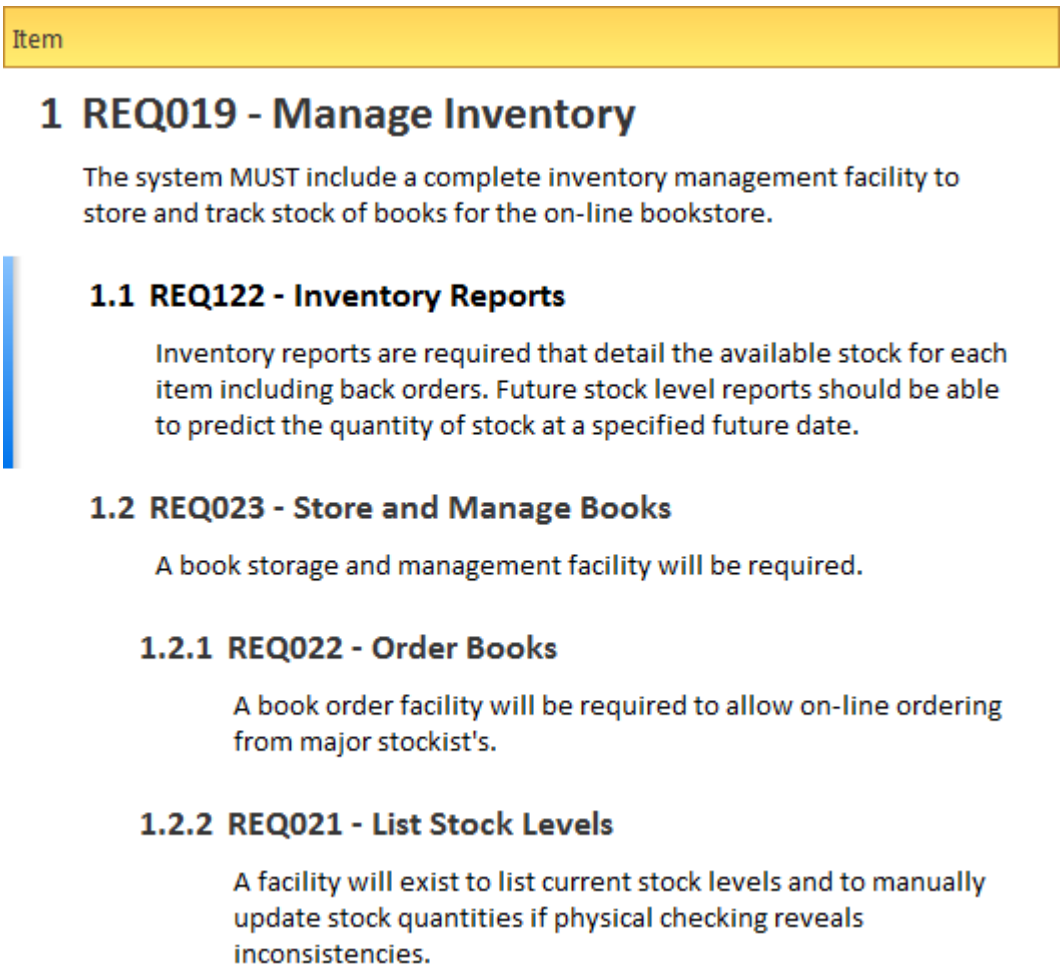
Requirement Engineering is one of the most important disciplines in the system lifecycle and has a documented impact on the success of projects.. Enterprise Architect is a sophisticated platform for developing and managing Requirements, and regardless of the domain, the size of the project or the method being followed, Enterprise Architect provides tools that make it easy to manage the largest of Requirement repositories in complex projects.

Analysts can work together via a collaborative platform with role based Security, Discussions, the Library window, Model Mail and a range of other tools to encourage best practice and productivity.

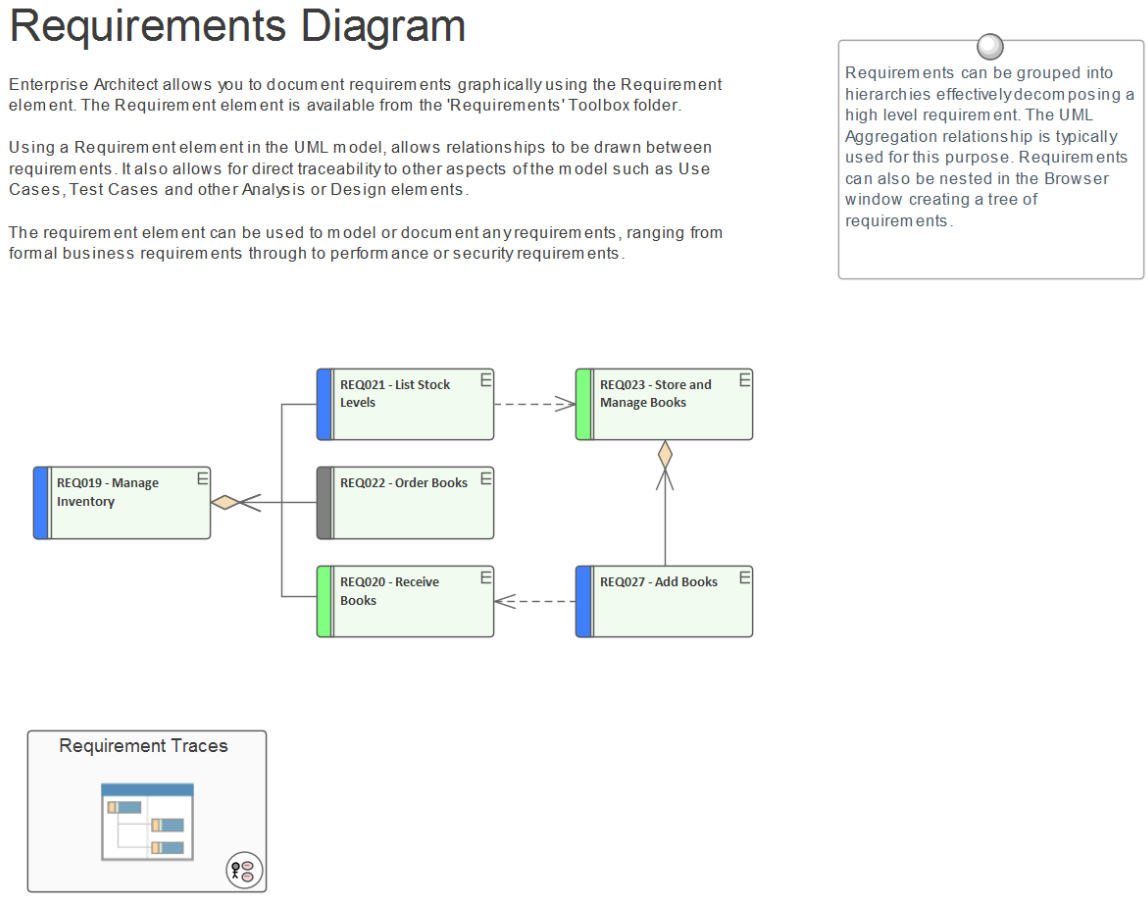
Requirement Development

Requirement Development consists of all the activities and tasks associated with discovering, evaluating, recording, documenting and validating the Requirements for a particular project. Requirements are discovered, analyzed, specified and verified. Enterprise Architect has a wide range of tools and features to assist the Analyst as they develop Requirements. The centerpiece for Requirement

Development is the Specification Manager, through which the Requirement Analyst can enter, view and manage Requirements in textual form as if in a spreadsheet. Requirement properties such as Status, Priority and Author can be edited in-line, and filters can be applied to restrict the display to particular requirements.

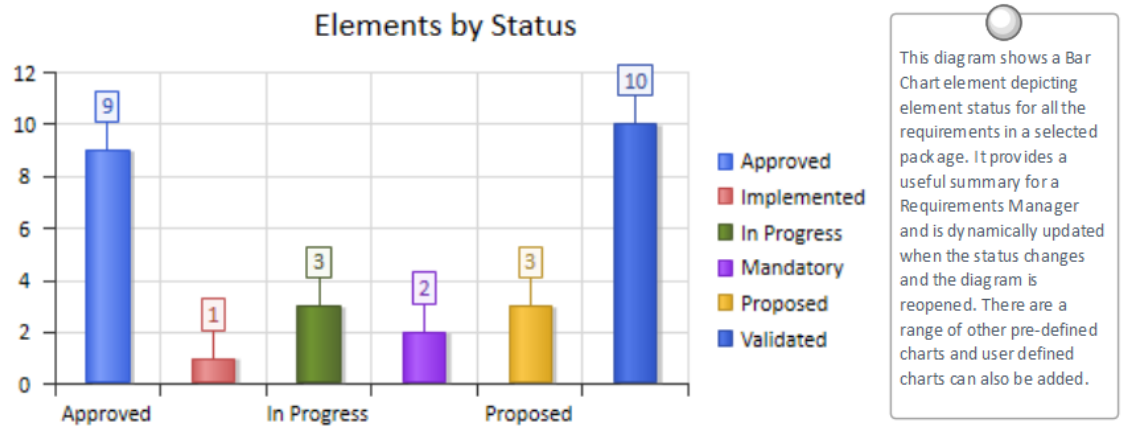


The Specification Manager can be used in conjunction with a platform of other tools such as diagrams, the Traceability window and the Discussions facility.



Requirement Management

Requirement Management comprises the activities to maintain a set of Requirements that represent an accord or agreement between the project team and the customer. It also has a focus on ensuring that the Requirements are acceptable to the Design and Development Teams, and that they are sufficiently specific to be implemented into working business, software or hardware systems.



Requirement Documentation

A number of documents are commonly produced as part of the Requirement Engineering discipline, such as the Software (System) Requirements Specification and Use Case Reports, and these can be generated automatically from a Requirement Model using built-in templates. In addition a wide range of other documents can be produced using built-in or customized templates.

Requirement Processes and Standards

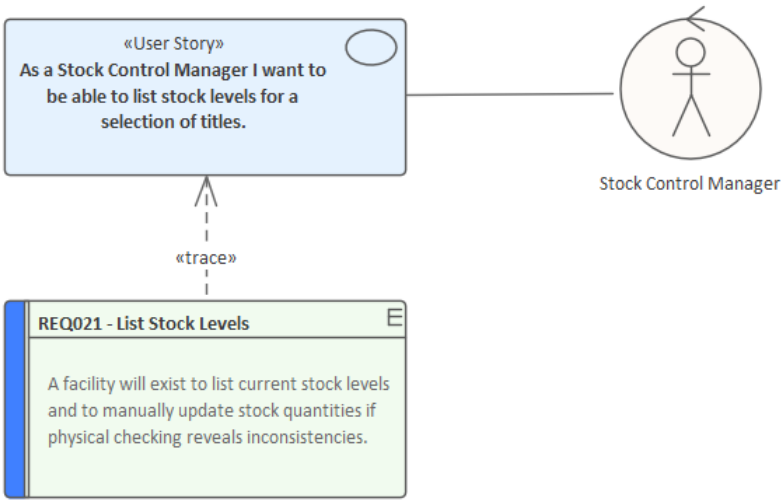
Enterprise Architect is an open platform that supports any Requirement engineering process. The tool has a rich feature set and is highly configurable, and its flexible design means that whatever method is being used you will find features to help. So whether the team is using Formal Requirements, Use Cases, User Stories or Story-Boards in any combination, Enterprise Architect can be used to develop, manage and document the Requirements. The implementation of the UML extension mechanisms means that any type of Requirement can be created and managed using built-in types or by using stereotyped elements and Tagged Values.

User Stories

User Stories are useful as an alternative way of describing user Requirements. They are typically used as part of an Agile development process, to provide a simple but clear description of what the user does or needs to do as part of the role they perform.

A User Story can be created using the stereotyped Artifact available from the Artifact Toolbox page, or as a stereotyped Use Case.

This diagram shows how a User Story can be modeled using a stereotyped Use Case. This allows the User Story to be described and to show the connection to a Persona.



Introduction to Requirement Modeling

Requirement Engineering is one of the most important disciplines in the system lifecycle and, when done well, it will set the foundation for a successful project or program of work, ultimately ensuring that great value is delivered to the users and other stakeholders. Enterprise Architect is a sophisticated and intuitive platform for developing and managing Requirements gleaned from modeling stakeholder statements, business cases, business drivers and capabilities to define detailed Functional and Non-functional Requirements. Requirements can be prioritized, traced and tracked, and changes can be recorded, baselined, versioned and audited. Analysts can work together in a collaborative platform with role based Security, Discussions, the Library window, Model Mail and a range of other tools to encourage best practice and productivity, either directly on the local system or through Pro Cloud Services.

How it Will Help You

You will typically come to the topic of Requirement Engineering with some existing knowledge or experience, even if it is something that has been learnt in lectures or by on the job training, or perhaps by using a different tool. You will benefit by understanding the product features and the tools that are available to develop and manage Requirements in Enterprise Architect, and this will enable you to be more

productive both when working alone and as a member of a team.

Who Will Benefit?

Anyone involved in the development or management of Requirements, whether at a strategic level, a business value level or a system development level, will benefit from reading this information. This includes a wide range of roles including Strategic Thinkers; Business and Requirement Analysts; Enterprise, Business, Technical and Solution Architects; Project and Program Managers; Developers, Test Designers and User Experience Designers.

What You Will Learn

This topic will teach you how to use the comprehensive features of Enterprise Architect to develop and manage Requirements, to create documentation and to work collaboratively as a member of a team using a formal or informal system life cycle process or standard.

Overview of the Documentation

Meet the Requirement Tools	Lists the key tools that are used for developing and managing Requirements, including a picture of each tool in action, where to find the tool, how to use it and how to become proficient in using the tool. There are a large number of additional useful tools that are described in the Help topic *Additional Requirement Tools*.
Requirements Overview	Puts Requirement Engineering in context by defining what Requirements are, the different levels of Requirement, characteristics of good Requirements and the business context of Requirements. The information also includes the concept of a Requirement diagram that readers coming from text based tools might not be familiar with, and how to create and view Requirements in Enterprise Architect.
Requirement Development	Discusses the activities and tasks associated with discovering, evaluating, recording, documenting and validating Requirements. The topic is conveniently divided into four sub-topics - Elicitation, Analysis, Specification and Validation - and identifies a wide range of features

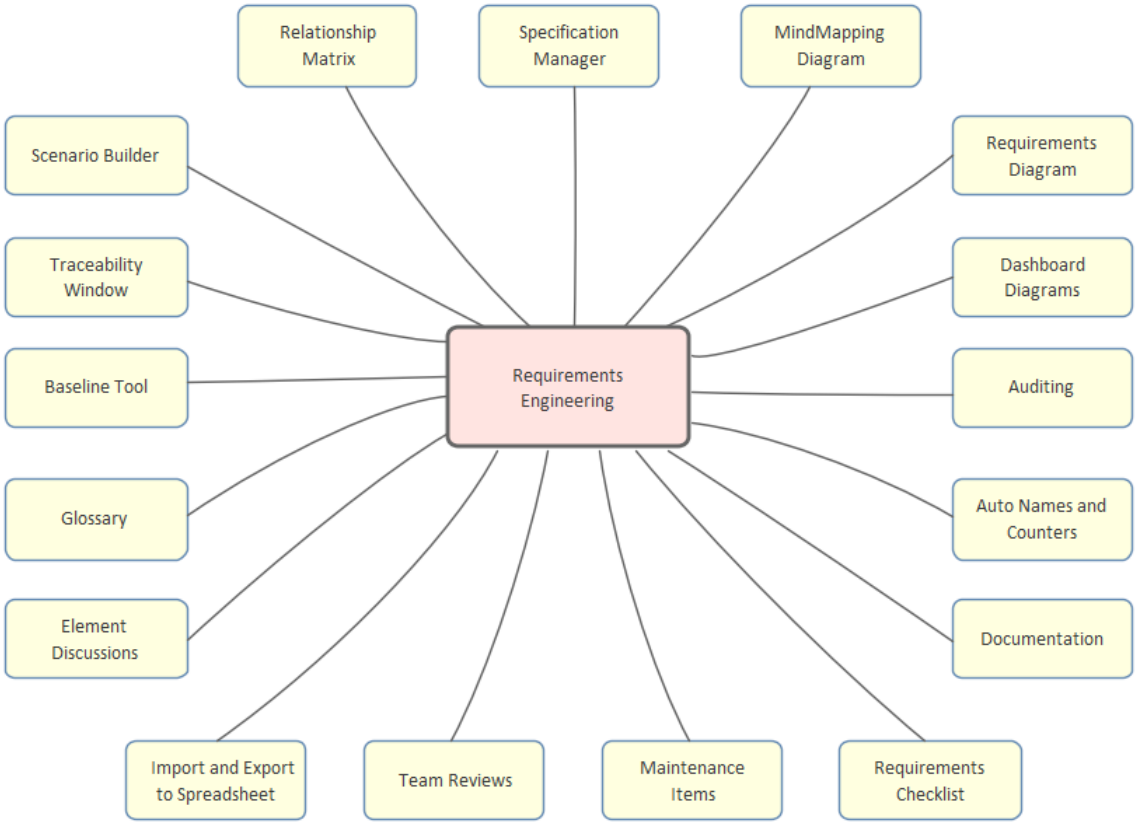
	that can be used, from Mind Mapping diagrams for recording information in elicitation workshops, to the Specification Manager for creating Requirements, to Test Cases for validating them.
Requirement Management	Describes the activities needed to maintain a set of Requirements that represent an accord or agreement between the project team and the customer. It includes composing hierarchies of Requirements, tracing other elements back to Requirements, and tracking the properties of Requirements including Status, Priority, and Difficulty. It also describes managing changing Requirements, Volatility and assessing the impact of changing Requirements.
Requirement Documentation	Describes how formal and informal Requirement documentation can be generated directly from Enterprise Architect using a series of predefined and extensible templates. This includes Glossaries, Data Dictionaries, Use Case Reports and Documents such as a System Requirements Specification.
Requirement	Puts the usage of Enterprise Architect's

t Processes and Standards	Requirement tools in the context of System and Requirement processes and standards. The topic describes how flexible the tools are and how they can be used with any process or standard.
Additional Requiremen t Tools	Lists a series of additional tools that can be used for Requirement Engineering, including a picture of each tool in action, where to find the tool, how to use it and how to become proficient in using the tool. There are a number of key tools that are the most important tools for Requirement Engineering that are described in the first topic, entitled *Meet the Requirement Tools*.

Meet the Requirement Tools

Enterprise Architect is a sophisticated and flexible Requirement modeling tool that can be used across the Requirements' life cycle from planning through to product support. The tool can be used with any Requirement Management process, and there is a wide range of features that allow analysts to work using their preferred methods, such as word-processor views, spreadsheet views, diagrams, Relationship Matrix or several other core and extended features.

This Mind Map shows the landscape of the key Requirement tools that can be used to develop and manage Requirements. While these are the primary tools there is a series of other tools described in the Help topic *Additional Requirement Tools*.


Relationship Matrix
Specification Manager
MindMapping Diagram
Scenario Builder
Requirements Diagram
Traceability Window
Dashboard Diagrams
Baseline Tool
Requirements Engineering
Auditing
Glossary
Auto Names and Counters
Element Discussions
Documentation
Import and Export to Spreadsheet
Team Reviews
Maintenance Items
Requirements Checklist

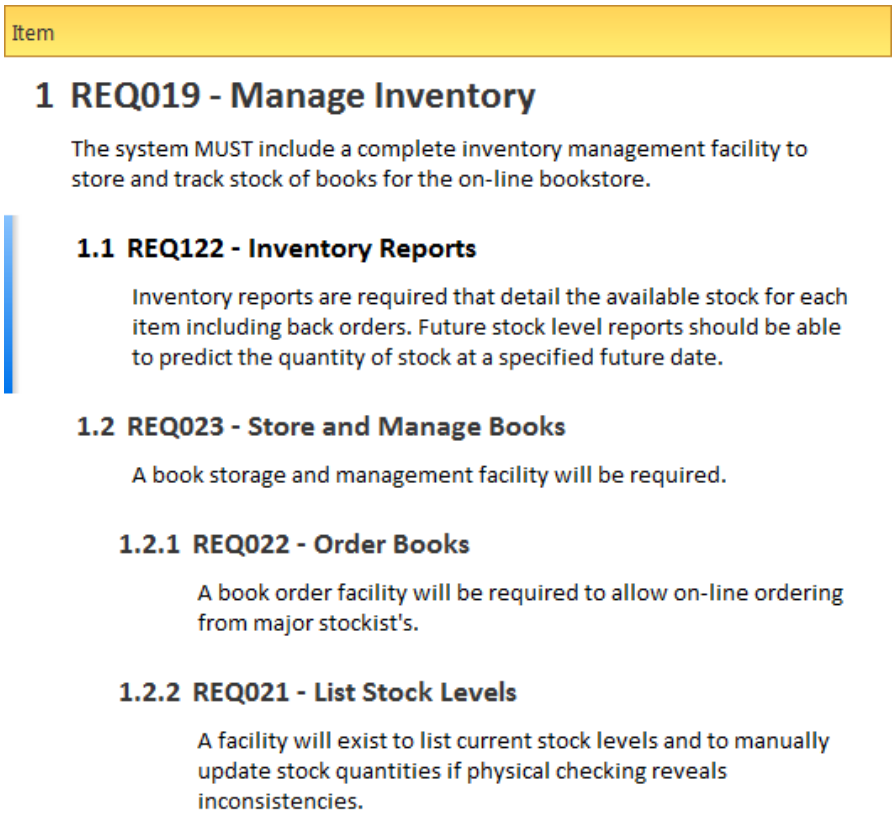
Specification Manager

Getting to Know the Specification Manager

Introducing the Specification Manager

The Specification Manager is the central tool for working with Requirements; it provides an interface resembling a Word Processor or Spreadsheet tool for entering, maintaining and viewing Requirements. New Requirements can be created with names and detailed descriptions and properties such as Status and Priority can be added from drop-down lists. Existing Requirements can be viewed and managed in a convenient view, and changing them in the Specification Manager will change them in all other places in the repository such as diagrams and windows. It is the perfect tool for those analysts more comfortable working with text rather than diagrams and who are accustomed to working in a Word Processor or Spreadsheet. It has the added advantage that the requirements are part of a model and can be traced to other elements, including Business Drivers, Stakeholders

and Solution Components.



Where to find the Specification Manager

Browser window | Right-click on Package | Specification Manager

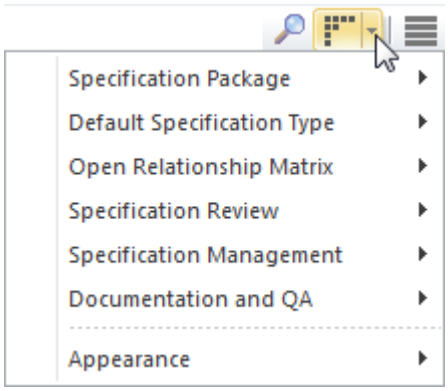
Usage of the Specification Manager

To create, view and maintain Requirements in a text based tool that resembles working in a word processor or spreadsheet. Details can be added to the Requirements and Requirement properties can be added from drop-down lists. When the Requirements are changed in the Specification Manager the changes are conveniently reflected in

the Browser window and all other windows.

Options for the Specification Manager

There are a wide range of options available from the options menu, to tailor the way you use the Specification Manager. These include Level (hierarchical) Numbering, Auto Naming, Spell Check, Documentation, Import and Export of Requirements, access to various related tools and more.



Learn more about the Specification Manager

The Specification Manager

The MDG Link for DOORS Add-In

The Model Driven Generator (MDG) Link™ for DOORS is an Add-In that provides support for linking an Enterprise Architect model to an IBM® Rational® DOORS® Requirements package. This tool is very useful when you need to perform your Requirements Management external to your Model Driven Development. Using this Add-In you can interchange Requirements defined within IBM DOORS with the traceable Requirements used within Enterprise Architect's Requirements Management features.

For all Enterprise Architect Editions other than Ultimate, you can purchase the MDG Link for DOORS separately and download the installer from the Sparx Systems website.

For full details concerning this Add-In, see the *MDG Link for DOORS* Help topic.

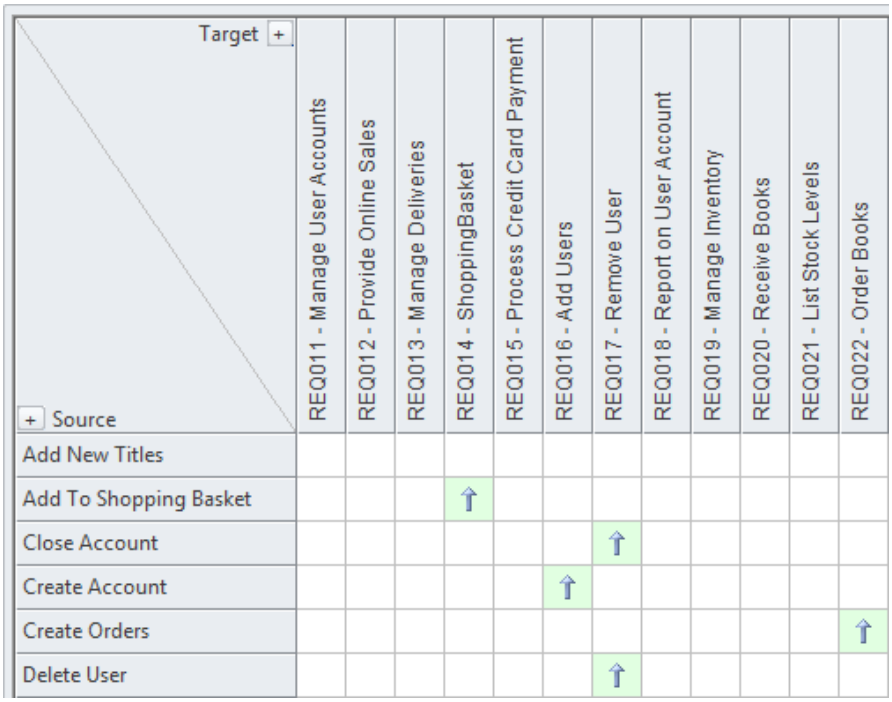
Relationship Matrix

Getting to Know the Relationship Matrix

Introducing the Relationship Matrix

The Relationship Matrix provides a visualizing compelling matrix-style view for a convenient analysis of the way that Requirements are related to each other and to other elements in the model. It can be used to view the relationships between Stakeholders and their Requirements, how Use Cases are related to Business Requirements or Functional Requirements, how Capabilities are related to Business Drivers, which Components implement a set of Requirements, and more. Any number of matrices can be defined quickly and then saved to be viewed in workshops, or included in documentation generated automatically from the model or exported to a spreadsheet file. When a matrix is created, connections can be viewed by placing the Requirements on one axis of the matrix and the connected elements on the other axis, then the cells of the matrix will indicate the direction

of the relationship.



Source \ Target	REQ011 - Manage User Accounts	REQ012 - Provide Online Sales	REQ013 - Manage Deliveries	REQ014 - ShoppingBasket	REQ015 - Process Credit Card Payment	REQ016 - Add Users	REQ017 - Remove User	REQ018 - Report on User Account	REQ019 - Manage Inventory	REQ020 - Receive Books	REQ021 - List Stock Levels	REQ022 - Order Books
Add New Titles												
Add To Shopping Basket				↑								
Close Account							↑					
Create Account						↑						
Create Orders												↑
Delete User							↑					

Where to find the Relationship Matrix

In the Browser window, click on a Package and select:

- The 'Resources' tab | Matrix Profiles | Right-click on a profile | Open Matrix Profile or
- The Start ribbon > All Windows > Design > Tools > Package Matrix

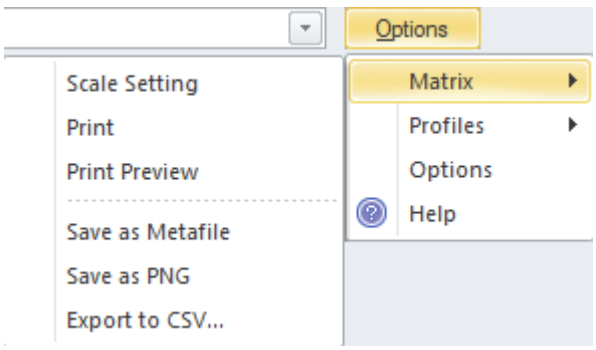
Usage of the Relationship Matrix

To display the relationships that exist between elements - such as which Requirements are realized by which Use Cases - in two Packages in a visually compelling matrix. It is useful in analyzing missing elements or relationships; for example, to determine which Requirements are not realized by

any Use Case, or which Components do not have corresponding Requirements or Use Cases. It is particularly useful in workshops with Business Stakeholders who might not be familiar with seeing Requirements in Trace diagrams.

Options for the Relationship Matrix

There is a range of options that can be set for the Relationship Matrix, including saving it to the 'Resources' tab of the Browser window or to a CSV format for opening in a spreadsheet. The appearance of the Relationship Matrix can also be altered by sorting the elements, showing an outline numbering view, and suppressing Package names. These items are available from the Options button on the Relationship Matrix.



Learn more about the Relationship Matrix

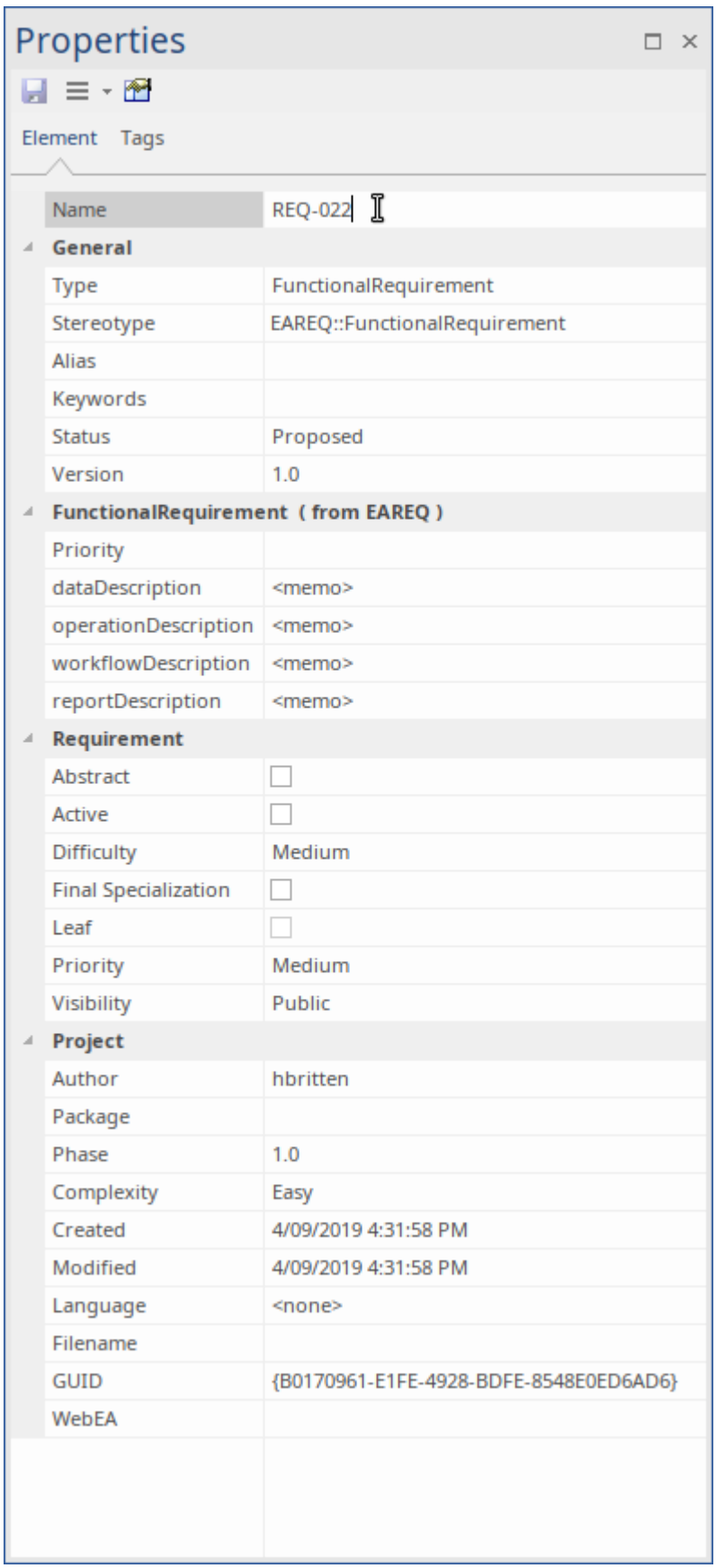
Relationship Matrix

Requirement Properties

Getting to Know the Requirement Properties

Introducing Requirement Properties

Requirement Properties define metadata about the Requirement that is useful for the management of Requirements for the purposes of prioritization and defining work Packages for the implementation teams. All Enterprise Architect elements have standard properties such as Status, Author and Phase, and the Requirement element has additional properties such as Difficulty and Priority. User-defined properties can also be created using Tagged Values.



Where to Ribbon: Design > Element > Editors >

find Requirement Properties	Properties Element Context Menu: Properties... | Properties... or Browser window Context Menu: Properties | Properties...
Usage of the Requirement Properties	The Properties define the important meta information about a Requirement, for the purposes of providing data to manage Requirements for prioritization, understanding which are the difficult Requirements, and managing the lifecycle by using Status to determine Requirements for implementation Packages.
Options for Requirement Properties	Enterprise Architect has a wide range of built-in properties for all elements, and a number of additional Requirement Properties. If other properties are needed by a modeler or team, such as the volatility (stability) of a Requirement, these can be added using the general-purpose UML extension mechanism of Tagged Values.

Learn more about Requirement Properties

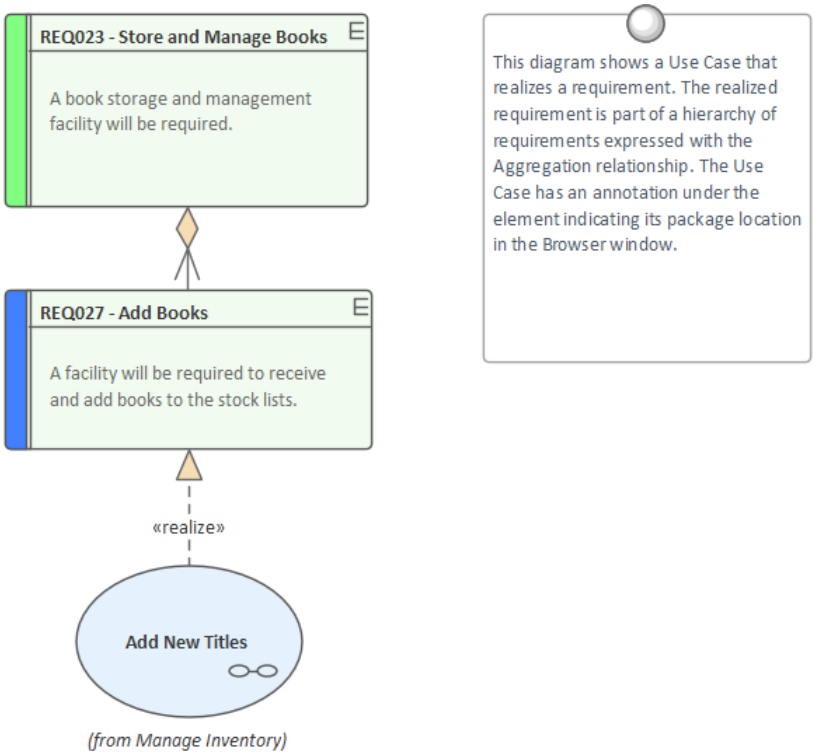
Properties Dialog

Requirements Diagram

Getting to Know the Requirements Diagram

Introducing the Requirements Diagram

The Requirements diagram provides a visual representation of how Requirements are related to each other and to other elements in the model, including Business Drivers, Constraints, Business Rules, Use Cases, User Stories, design Components and more. The diagram is one of Enterprise Architect's extended diagram types. It provides an appealing graphical representation of Requirements, that will be a welcome change for Requirements Analysts who are accustomed to working with text based tools.



Where to find the Requirements Diagram

Browser window Context Menu : Add Diagram : | Manage | Show All Perspectives | Extended | Requirements

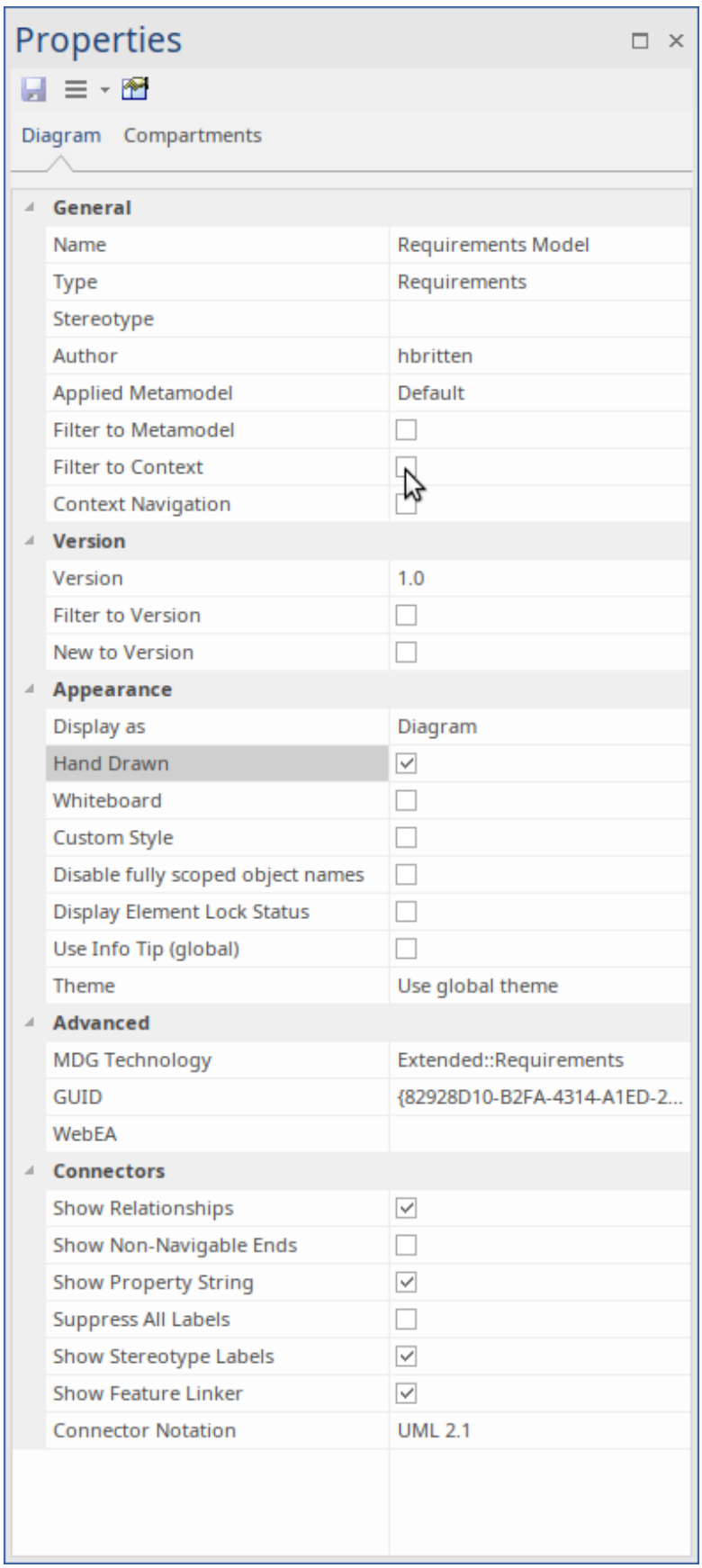
Usage of the Requirements Diagram

One usage is to show how Requirements are connected together in a hierarchy or, even more importantly, how Requirements are connected to other elements. The experienced modeler will define and manage the Requirements in the Specification Manager and then use the Requirements diagram to show how each Requirement is related to upstream process elements such as Business Drivers, and downstream process

elements such as Use Cases, User Stories, User Experience designs and solution Components.

Options for the Requirements Diagram

The appearance of a diagram can be changed to suit the audience, and details can be included, suppressed or altered to ensure the diagram meets its main objective of communication. There is a wide range of options, ranging from creating a Hand Drawn style of diagram to filtering diagram content.


Properties
Diagram
Compartments
General
Name
Requirements Model
Type
Requirements
Stereotype
Author
hbritten
Applied Metamodel
Default
Filter to Metamodel
Filter to Context
Context Navigation
Version
Version
1.0
Filter to Version
New to Version
Appearance
Display as
Diagram
Hand Drawn
Whiteboard
Custom Style
Disable fully scoped object names
Display Element Lock Status
Use Info Tip (global)
Theme
Use global theme
Advanced
MDG Technology
Extended::Requirements
GUID
{82928D10-B2FA-4314-A1ED-2...
WebEA
Connectors
Show Relationships
Show Non-Navigable Ends
Show Property String
Suppress All Labels
Show Stereotype Labels
Show Feature Linker
Connector Notation
UML 2.1

Learn more

about the Requirements Diagram

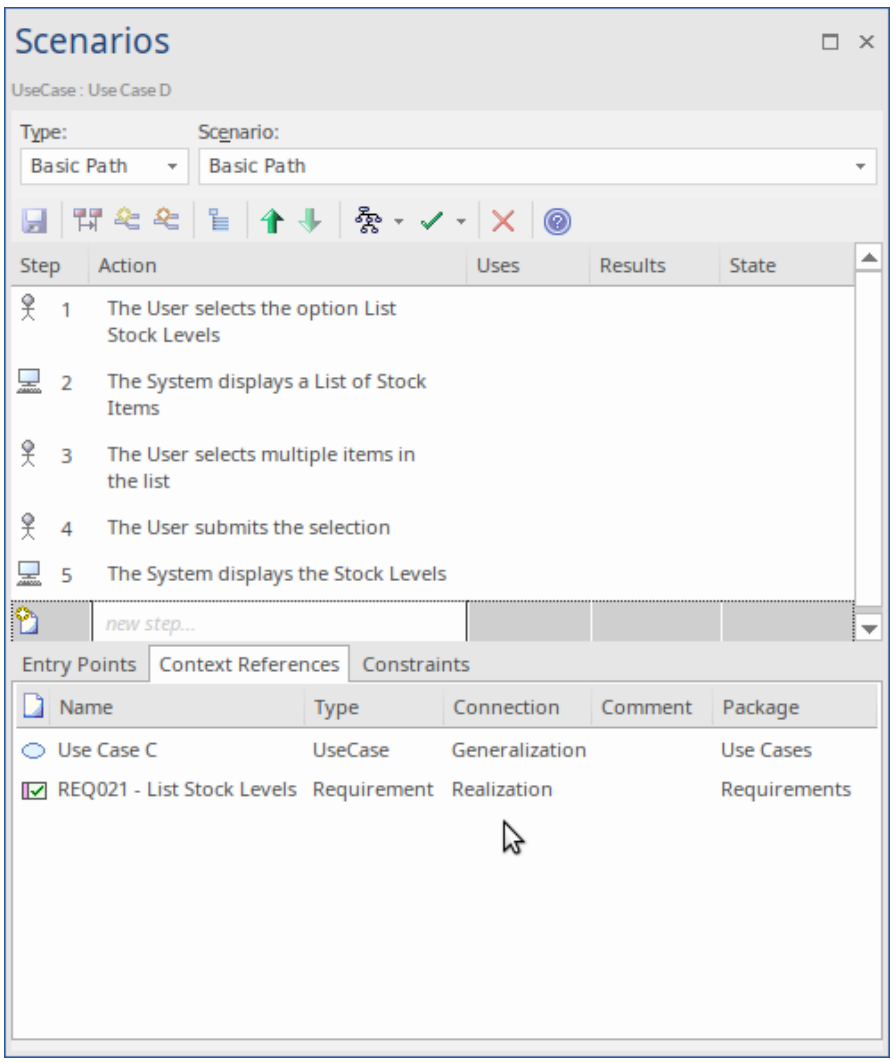
Working In Diagrams

Scenario Builder

Getting to Know the Scenario Builder

Introducing the Scenario Builder

The Scenario Builder is used to define the details of a Use Case including defining detailed descriptions, creating one or more Scenarios and defining pre-conditions, post-conditions and other constraints. The detailed steps of a Use Case can be recorded and linked to other elements in the model and these can then be generated out as a diagram providing a visual representation of the Use Case and its Scenarios. The diagram and the text can be synchronized and individual steps can then be traced to other elements such as Components that will realize the Requirement specified in the Use Case.



Where to find the Scenario Builder

Start > Application > Edit > Responsibilities > Structured Scenarios

Design > Element > Editors > Responsibilities > Structured Scenarios

Element Context Menu: Properties | Responsibilities > Scenarios | right click | Add New : Structure Editor

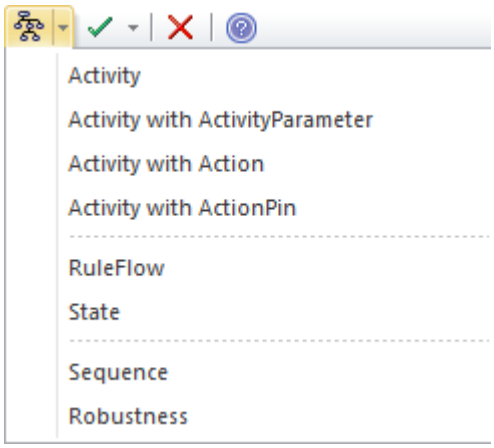
Usage of the Scenario Builder

To define the details of a Use Case and its scenarios and constraints, which can be used to replace the traditional

text-document based approach to defining Use Cases. This ensures that the Use Case diagram and the textual details of the Use Cases and its Scenarios and Constraints are all contained in the same model and can be traced. If the Use Cases are required in a document format for contractual or process reasons, a Use Case Report can be generated automatically from the models using the in-built documentation engine.

Options for the Scenario Builder

The Scenario Builder can be viewed as a tabbed or a docked window or in an element's Properties window. The steps of a Use Case including its Scenarios can be automatically generated into a number of different diagram types available from the Generate Diagram toolbar icon.



Learn more about the

Scenarios

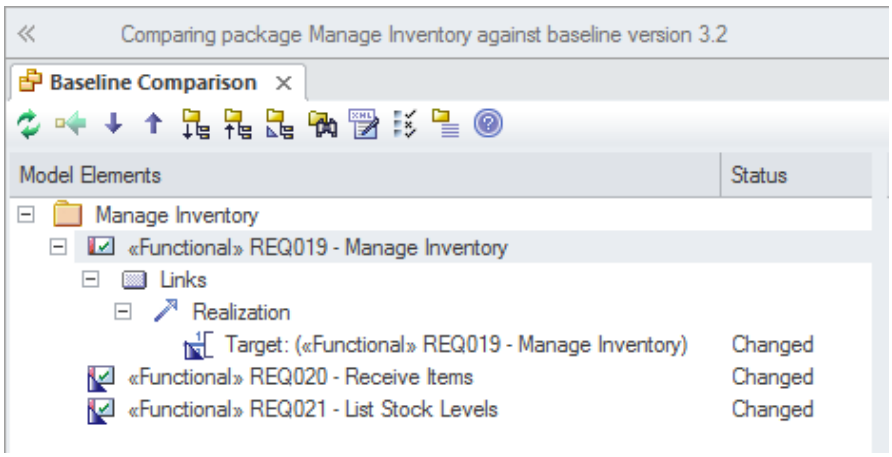
Scenario Builder

Baseline Tool

Getting to Know the Baseline Tool

Introducing the Baseline Tool

The Baseline Tool can capture a snapshot of the Requirements at a point in time and then, at a later time, the repository can be compared to this (or another baseline) for the purpose of determining what has changed. Any number of baselines can be created and labeled, and there is a baseline comparison tool that displays the differences between the baseline and the model and allows the modeler to revert a change in the model to a baseline at a granular level.



Where to find the Baseline Tool

Ribbon: Design > Package > Manage > Manage Baselines

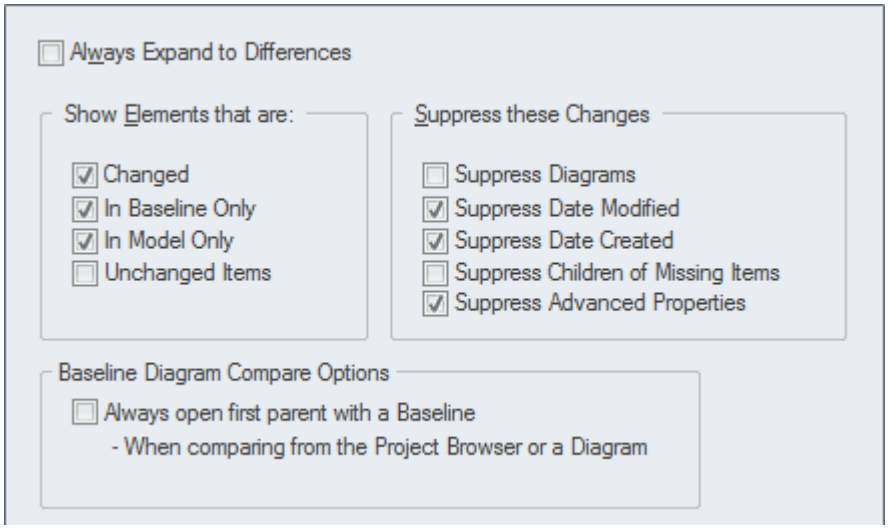
Keyboard: Ctrl+Alt+B

Usage of the Baseline Tool

Baselines are also useful when a formal requirements process is being followed or the Requirements form part of a contract, as the baseline can keep a snapshot of the requirements at important milestones such as contract signing or requirement phase sign off. This is also applicable to iterative and incremental processes such as Agile methods, as the requirements can be baselined before or even after a Sprint. When Requirements are still volatile and the Requirements' owners are still formulating their needs, a baseline can be created to take a snapshot at important points in the analysis phase, such as after an elicitation workshop.

Options for the Baseline Tool

There are several options that can be applied to configure the way the Baseline Compare tool presents information; these are available from the Options button on the Baselines window.



Learn more about the Baseline Tool

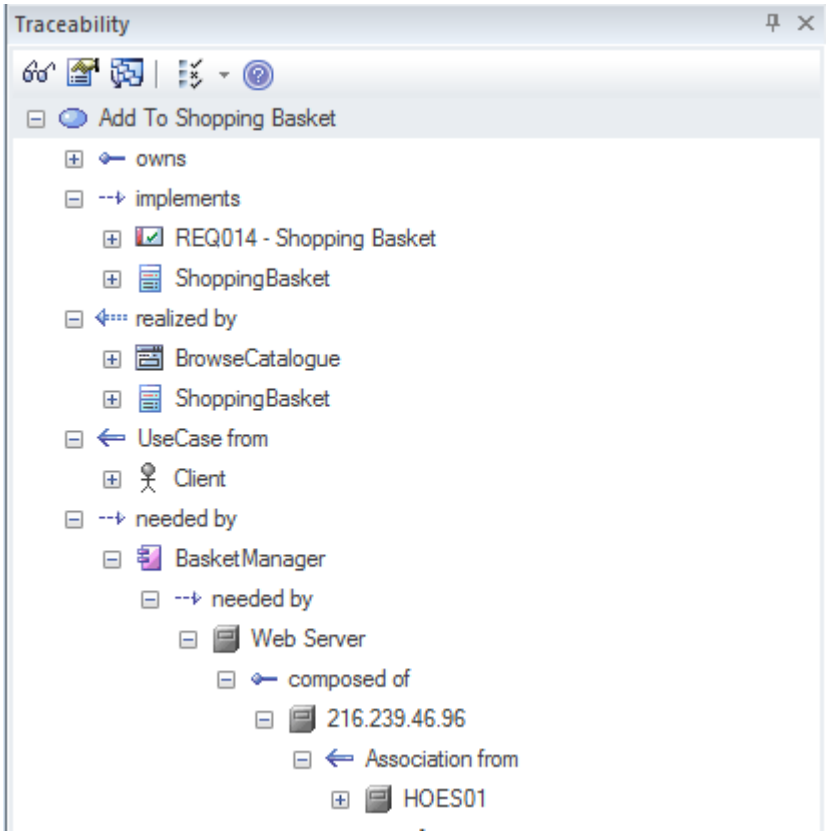
Baselines

Traceability Window

Getting to Know the Traceability Window

Introducing the Traceability Window

The Traceability window provides a hierarchical view of element connections, allowing traceability to be visualized and queried as elements are traversed in the model. This tool is particularly useful because a modeler will often choose to hide diagram relationships, but by selecting an element in the diagram and viewing its connections in the Traceability window all its relationships will be revealed.



Where to find the Traceability Window

Start > Application > Design > Traceability

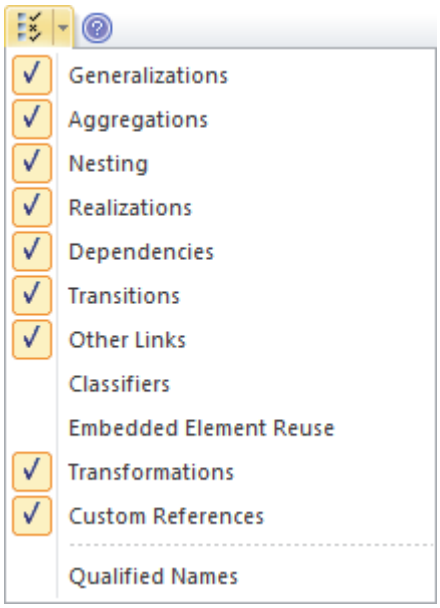
Usage of the Traceability Window

The Traceability window provides a hierarchical view of the way an element is connected to other elements in the repository, along with the type of each relationship. This window gives a complete list of all relationships that cannot be seen by viewing elements in the Browser window and that also might not appear in any diagrams. It is very useful for managing Requirements and

tracing how a Requirement is related to upstream process elements such as Business Drivers and downstream process elements such as Components. It is a useful tool, enabling newcomers to a model to gain a quick understanding of which are the important and well connected elements. Before you delete an element in the model, you should use the Traceability window to ensure that you understand that element's existing relationships.

Options for the Traceability Window

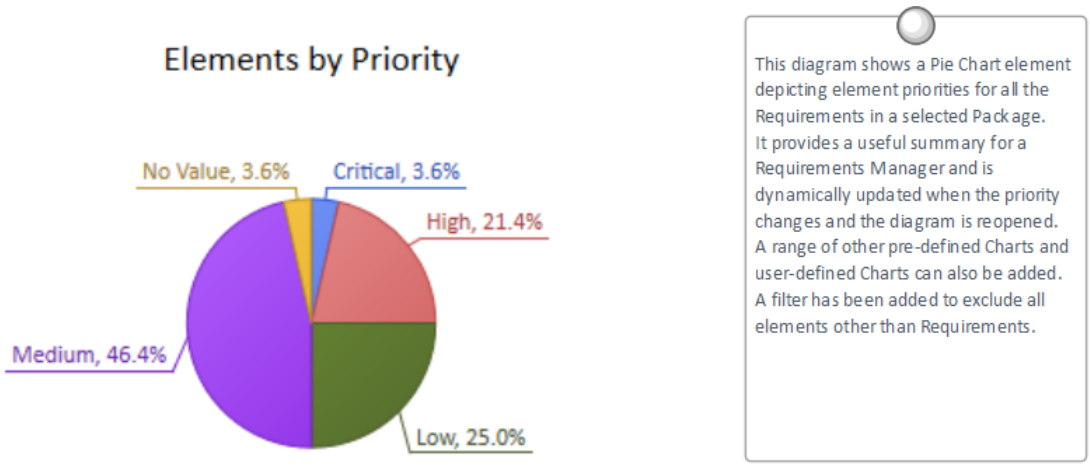
There is a series of options that restrict traceability to specified connector types; these options can be set to alter what is displayed in the window. The options are available from the toolbar at the top of the window.



Learn more about the Traceability Window	The Traceability Window

Dashboard Diagrams

Dashboard diagrams allow you to create high quality Charts and graphs to display repository information in a visually compelling way. This diagram is an example of creating a Dashboard d**iagram** in Sparx Systems Enterprise Architect; it illustrates the ratio of Requirement Priorities in a Pie Chart.

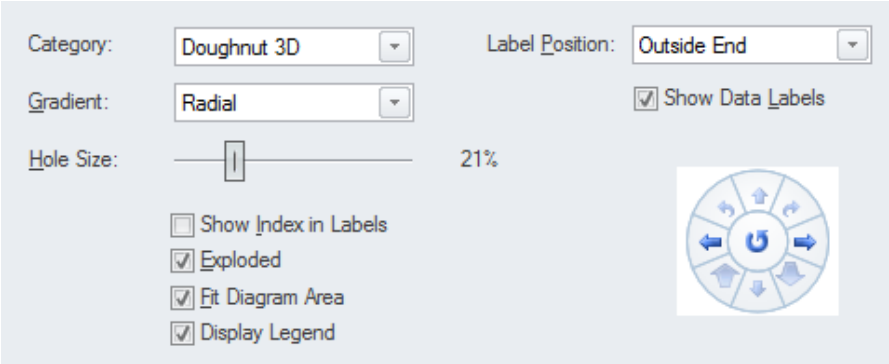


Enterprise Architect provides a Toolbox page of pre-configured Charts and graphs, but you are free to create and save any number of Charts, sourcing data from anywhere in the repository. The Charts and graphs provide valuable summary information that assists in the management of Requirements. High level reporting and project status can be easily tracked and documented using the numerous Charts and report elements available, which tightly link in with the model content and status.

Getting to Know Dashboard Diagrams

Where to find Dashboard Diagrams	Browser window | Right-click on Package | Add Diagram : ☰ | Manage | Show All Perspectives | Extended | Dashboard
Usage of Dashboard Diagrams	Dashboard diagrams present rich yet easily understood views of information - such as the status of Requirements in a particular release of the system - that can be opened inside the model or conveniently copied directly into management or project team presentations. They are useful for planning an iteration such as an Agile sprint to view how ready the Requirements are for the implementation team; for example, to view what percentage of the Requirements have been approved and are of high priority.
Options for Dashboard Diagrams	The standard Charts and graphs available from the Toolbox can be configured in a number of ways, including changing the source, applying filters or modifying the appearance of the Chart as indicated in this diagram, available from the Chart's

Properties window using the 'Appearance' section.



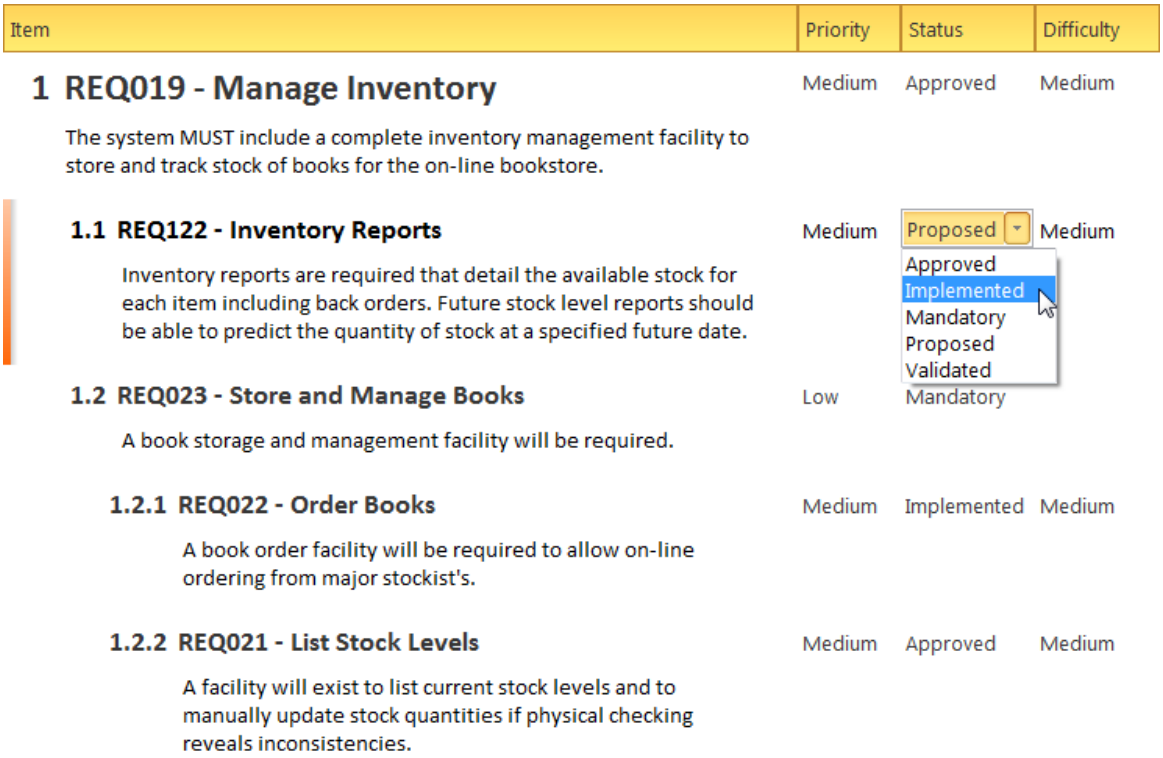
Learn more about Dashboard Diagrams

- Standard Charts

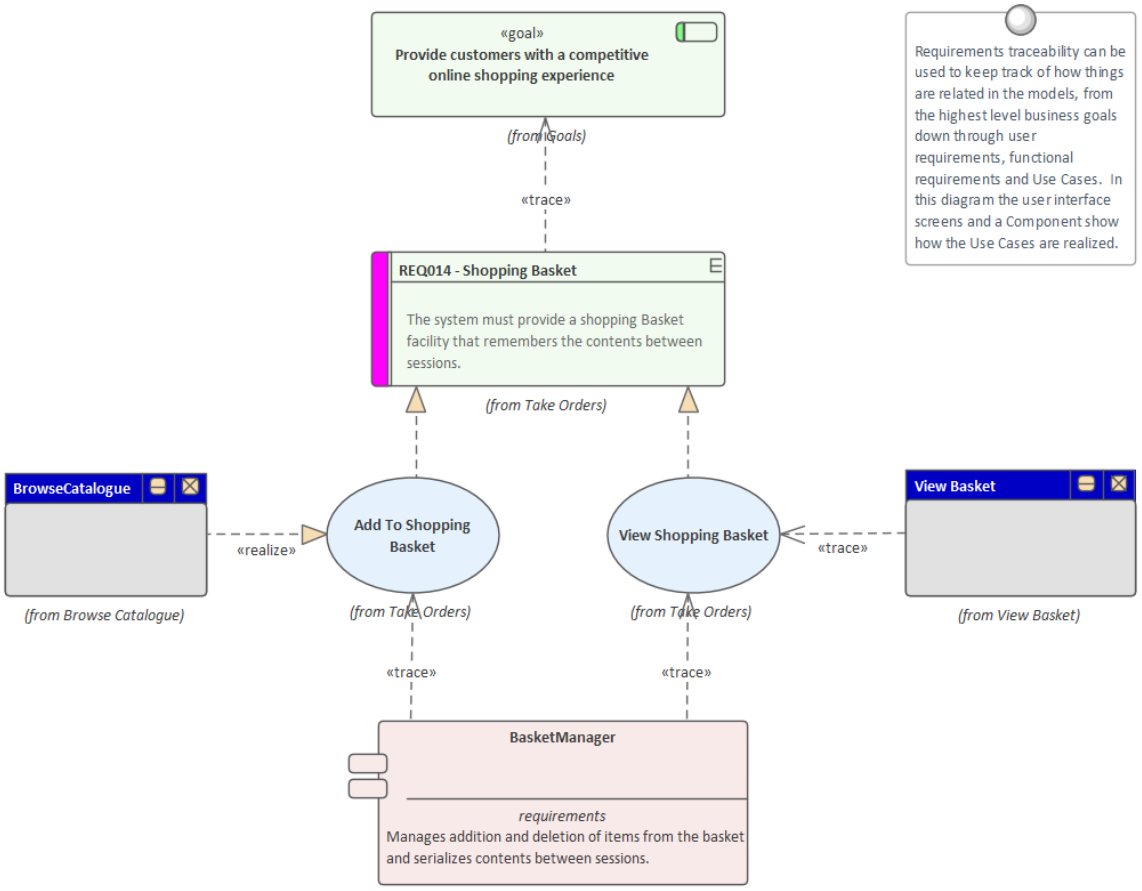
Requirements Overview

The discipline of Requirement Engineering is one of the most critical parts of any system development process. Whether it is an enterprise, business, engineering, real-time, software or hardware system, the definition and management of requirements is critical to the success of any endeavor. Clear and unambiguous articulation of requirements will ensure that the implementation team has the problem defined, giving the best chance of the correct solution being implemented. Enterprise Architect equips the Requirement Analyst and Manager with a formidable set of tools to take on this important challenge.

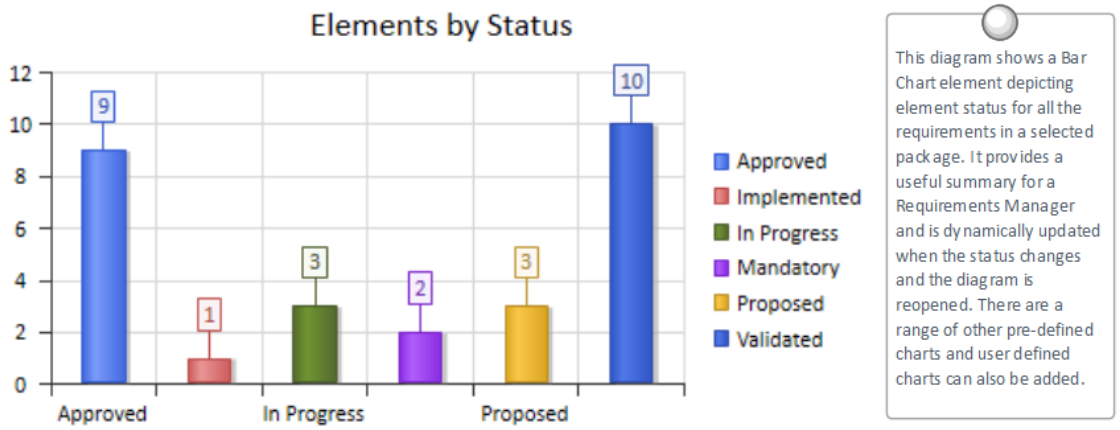
The multi-featured Specification Manager allows requirements to be entered, viewed and managed in a spreadsheet format, facilitating rapid input and editing of requirements. Requirement properties, including Tagged Values, can be edited in-line and values can be selected from drop-down lists.



Requirements can be viewed in diagrams and related to other model elements, creating compelling representations of traces between specifications and implementations.



Requirements can be managed in a variety of ways, from the use of the Project Gantt Chart to the Dashboard diagrams that show properties such as Status for a set of requirements.



These are just a few of the tools that are available to the Requirement Analyst and Managers.

What are Requirements

A requirement is a capability or condition that must be met to ensure that a solution meets the needs of its stakeholders. There is often robust debate about exactly what constitutes a requirement and some proponents will include notions such as Business Drivers and Policies and Business Rules while others will have a much more restrictive view of the requirements. Also a number of requirement methods are Use Case centric and only articulate requirements at a business level while others augment the Use Cases with detailed functional requirements required by the developers. The highly iterative methods such as Agile typically use User Stories and Requirements together but defer the elaboration of Requirements until an iteration (sprint) is being planned. Enterprise Architect provides generic tools to support any requirements method and any type of Requirement can be created and managed using built-in types or by using stereotyped elements and Tagged Values.

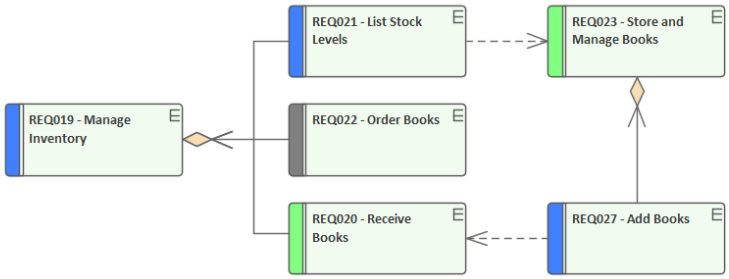
Requirements Diagram

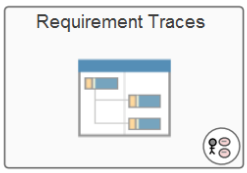
Enterprise Architect allows you to document requirements graphically using the Requirement element. The Requirement element is available from the 'Requirements' Toolbox folder.

Using a Requirement element in the UML model, allows relationships to be drawn between requirements. It also allows for direct traceability to other aspects of the model such as Use Cases, Test Cases and other Analysis or Design elements.

The requirement element can be used to model or document any requirements, ranging from formal business requirements through to performance or security requirements.

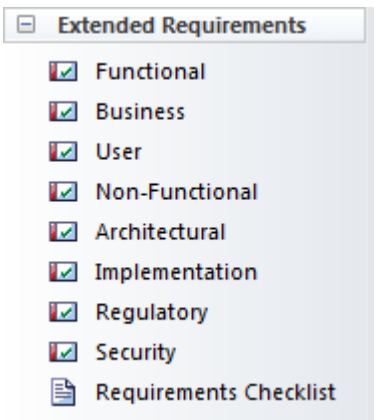
Requirements can be grouped into hierarchies effectively decomposing a high level requirement. The UML Aggregation relationship is typically used for this purpose. Requirements can also be nested in the Browser window creating a tree of requirements.





Levels and Types of Requirements

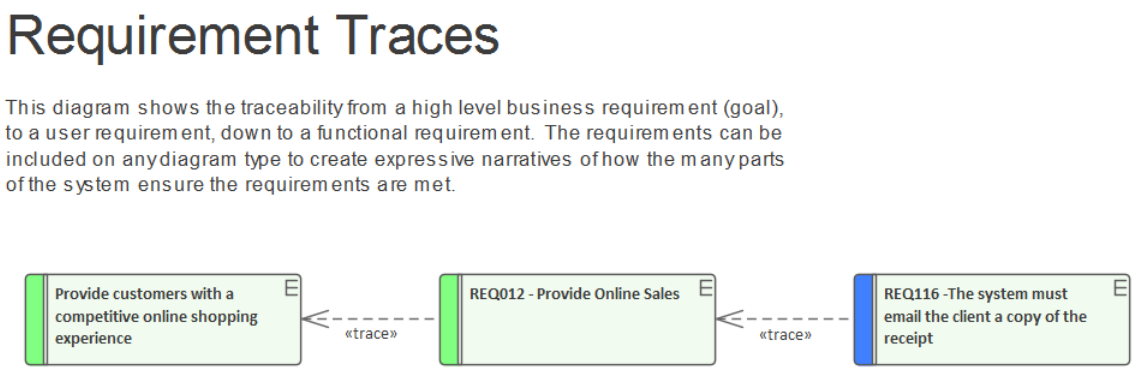
There are many different types of requirement, ranging from high level business requirements down to detailed technical requirements that specify an intricate part of a computer algorithm or hardware device. There are also types based on the source of the requirement - such as stakeholder requirements - or the location in the process - such as transition requirements. There is often confusion and debate about exactly what constitutes a requirement, so some teams will define Business Rules and Policies as requirements and others will view them as business specifications. Regardless of the method or the process that is being followed, Enterprise Architect allows the analyst to create sophisticated models of all requirement types.



Business Requirements

Business Requirements are high-level requirements that express the objectives and desired outcomes of an organization. They are often disregarded as being 'fluffy' by

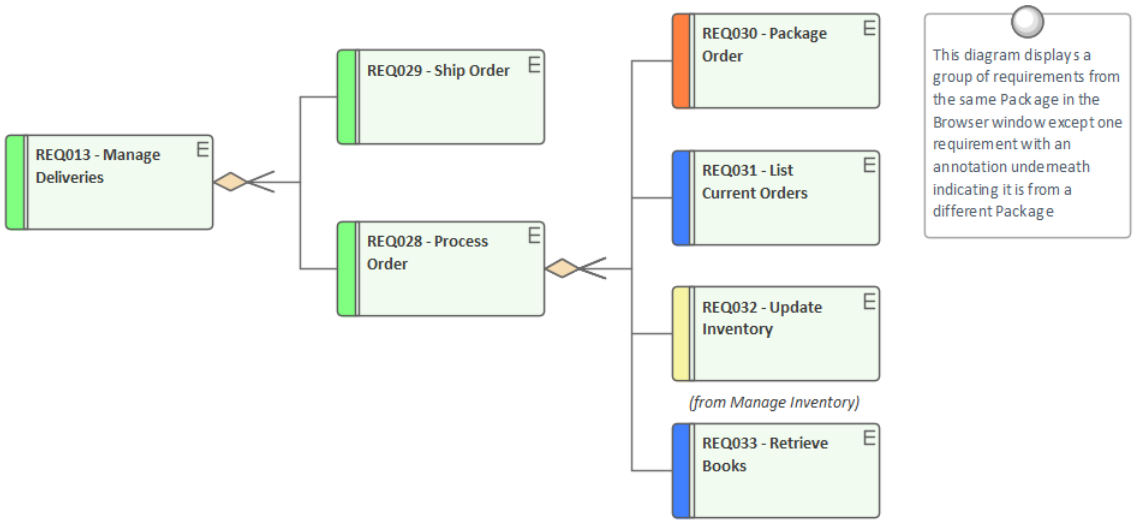
engineers who cannot see how they would be implemented, but if they are articulated well they can be broken down to measurable statements. They are typically defined in a business case or other statements by the product owner or sponsor, the marketing department or the customer. They attempt to articulate why the organization is spending money and resources on the project. Enterprise Architect has a Business Requirement element available from the 'Requirements' toolbox page for this purpose.



Functional Requirements

Functional Requirements are the bridge between the business and technical teams and provide the definition of what the system must do for its users that will in turn meet the business goals. Some methodologists believe that Functional Requirements can be described using only Use Cases or User Stories, but this appears to be a purist view and in practice there seems to be a need for detailed textual Requirements that describe what the architect must design

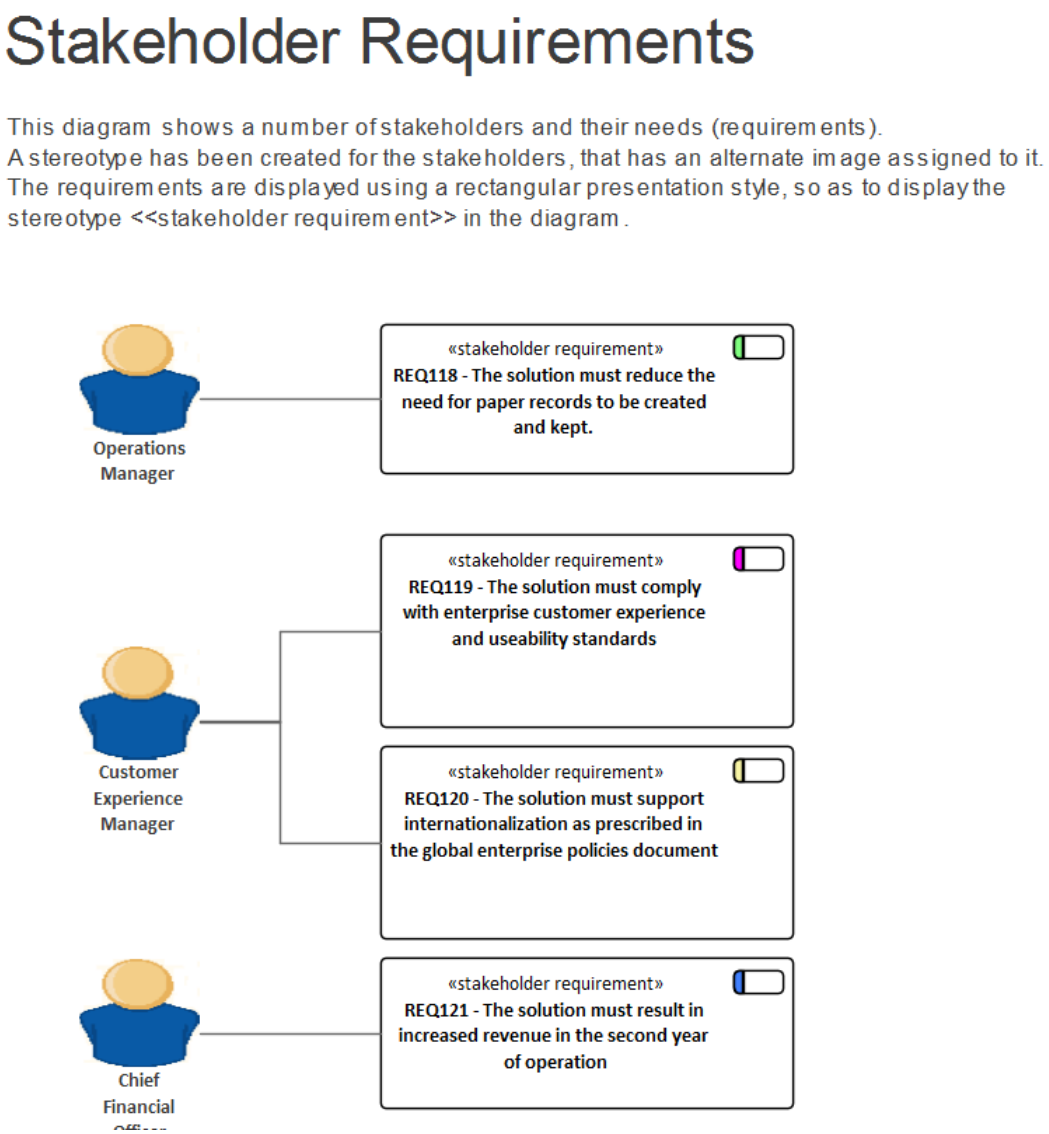
and the developer must implement. Enterprise Architect has a Functional Requirement element available from the 'Requirements' toolbox page. There is also an Architectural Requirement available from the 'Extended Requirements' page of the Requirements toolbox. In addition there is support for modeling Use Cases and Scenarios using the Scenario Builder.



Stakeholder Requirements

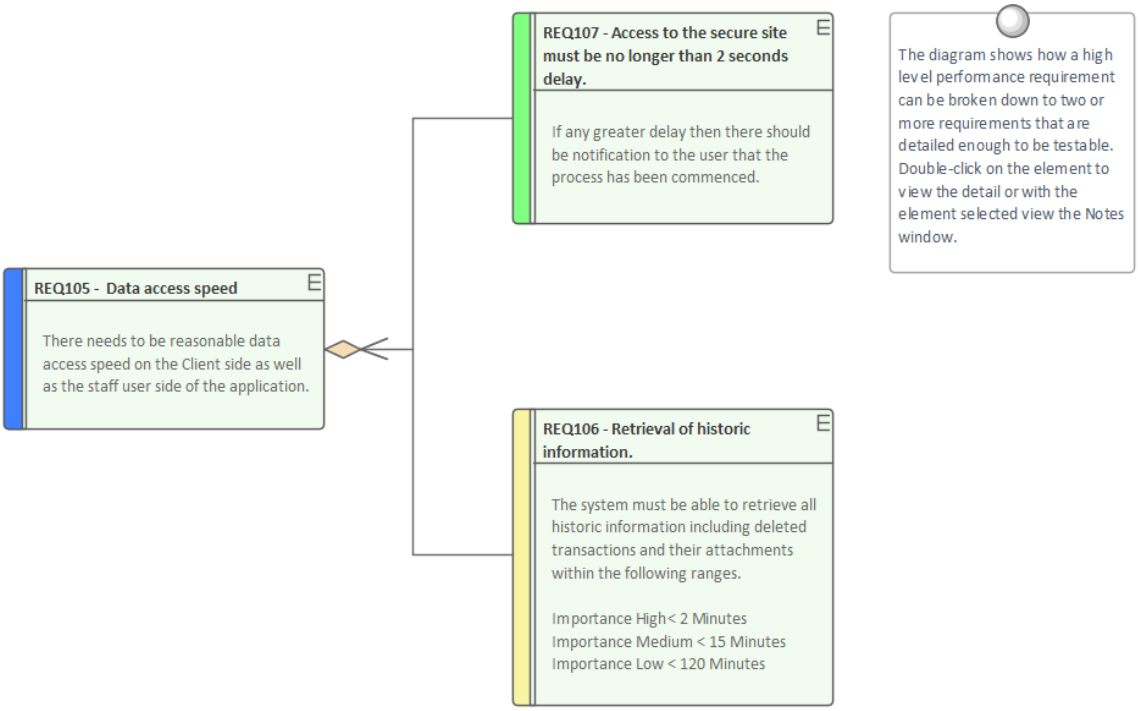
Stakeholder Requirements are statements of the stakeholders' needs and expectations and describe the features that must be met if the business requirements are to be fulfilled. Analysts tend to focus on the functional aspects of the needs but stakeholders' expectations might include performance and reliability and a variety of other non-functional needs. Both are critical and act as precursors to the definition of the functional and non functional

requirements that will be consumed by the designers and implementers to create solutions that meet the customer's expectations. Enterprise Architect has a Requirement element that can be stereotyped to <<stakeholder requirement>> available from the 'Requirements' toolbox page for this purpose.



Non Functional Requirements

Non-Functional Requirements and Quality Attributes describe how well a system will perform when it is operating. These typically define or constrain how the system should be behave as a whole and include attributes such as how well it performs, how secure it is, how many times it develops a fault and how easily it can be extended.



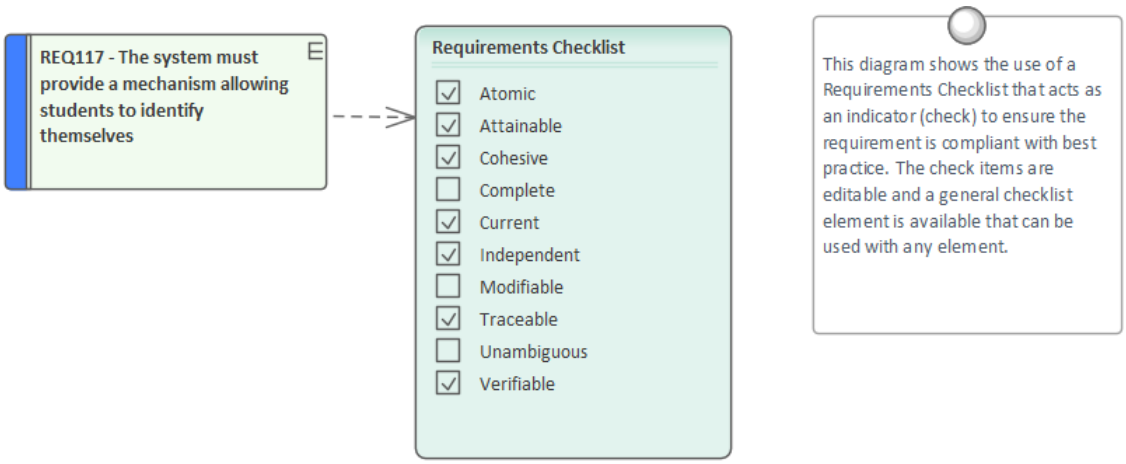
Transition Requirements

Transition Requirements define what is needed to transform the business and systems from the current state to the future state. They define a transitory situation and once the system has been fully implemented the requirements and their implementation will not be visible. They define things such as training, conversion and reformatting of data and parallel

runs of business and technology systems.

Characteristics of Good Requirements

More often than not errors and deficiencies in systems can be traced back to requirements engineering, and the literature frequently mentions the small cost of correcting a requirement compared to the large cost of correcting the system once it is built. Well articulated, managed and tested Requirements are therefore imperative to any system development process. Enterprise Architect has a convenient Requirements Checklist element available from the 'Extended Requirements' page of the Requirements Toolbox.

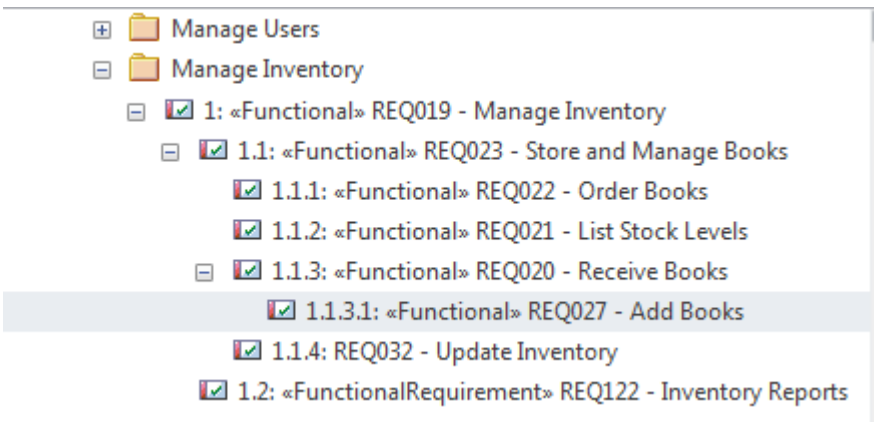


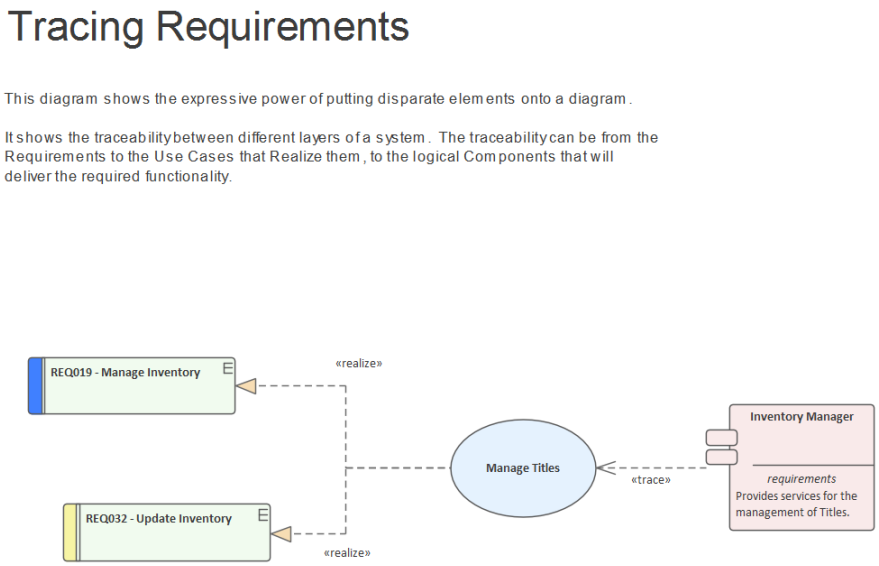
The Checklist can be used to indicate if a Requirement is ready for implementation.

Qualities of Good Requirements

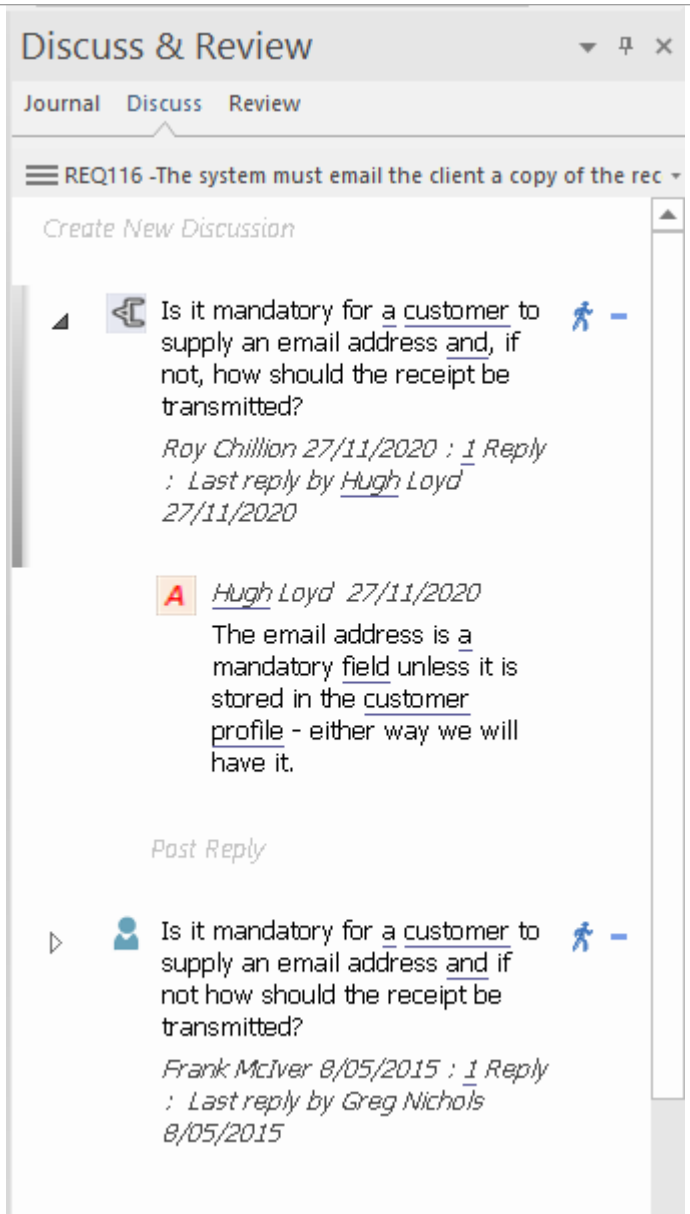
To be effective a set of Requirements must be complete and fully record the stakeholders' needs consistently, cohesively and unambiguously. Enterprise Architect provides an extensive set of features and tools for helping the analyst

produce sets of Requirements that are of high quality.

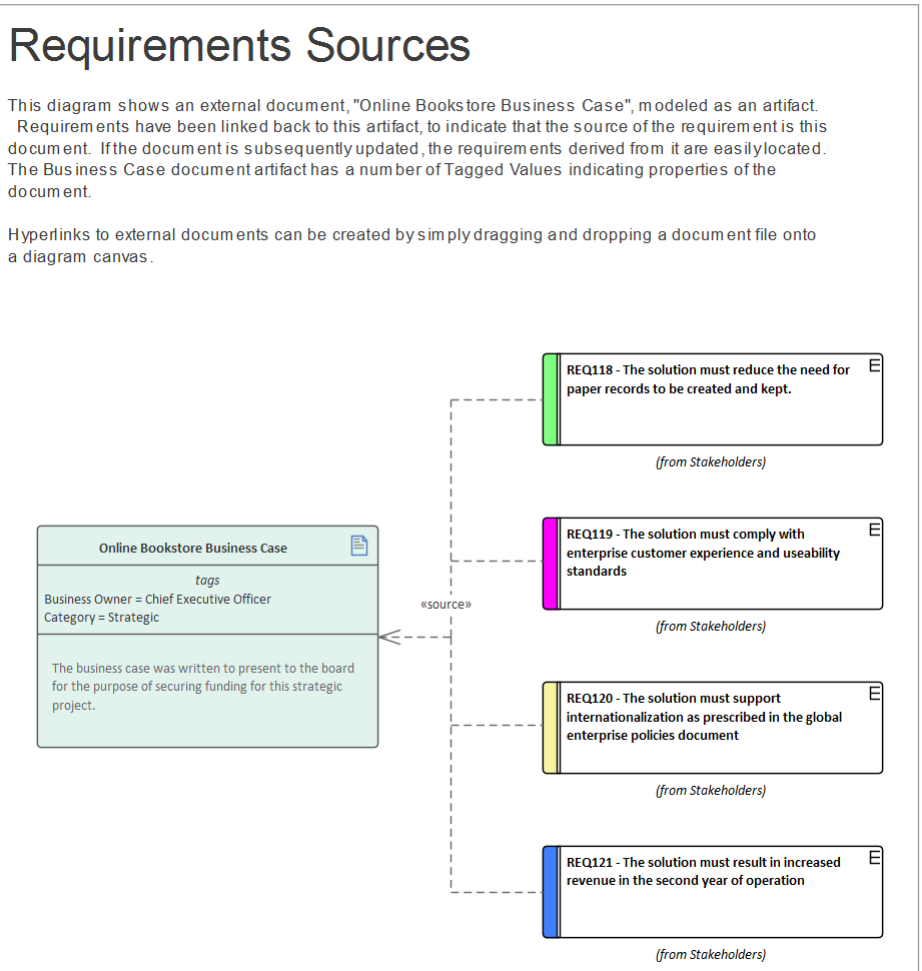
Quality	Description
Atomic	A requirement should articulate a single stakeholder need or a quality attribute. When a requirement contains multiple needs it is not possible to analyze the needs independently. Enterprise Architect can assist by allowing modelers to create hierarchies of requirements in the Browser window, which can be broken down to an atomic requirement. 
Attainable	The need specified in the requirement must be achievable. If a requirement is not attainable the system will not be able to deliver the business value required by the stakeholders. Enterprise Architect can assist by allowing each requirement to be traced to an implementation element such as a Use Case or a Component. The Relationship Matrix can be used to

	quickly identify those requirements that are not traced to a lower level element. 
Cohesive	The requirements as a set must be consistent and cohesive and express the behavior of the system; any gaps must be determined and overlap between requirements must be resolved. Following a requirements process will assist greatly, and Enterprise Architect has a number of facilities that will make it easy to keep the requirements cohesive. Missing requirements can be identified using the Relationship Matrix where, for example, a matrix of stakeholders and their requirements would quickly identify stakeholders who didn't have requirements.
Complete	Each requirement must fully describe the

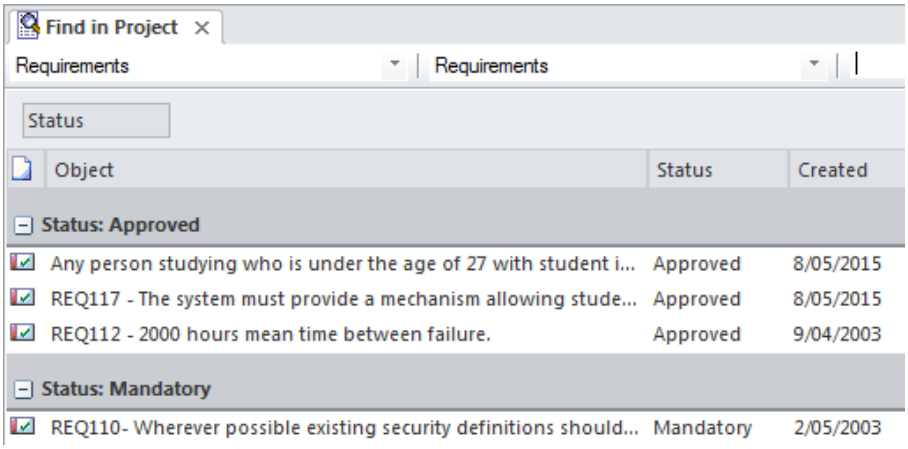
	necessary functionality or behavior that will result in the stakeholder's need being met. Enterprise Architect can help by team members using the Model Library facility or the Discuss & Review window. Some analysts prefer to mark requirements as needing to be completed, by appending the Requirement element with a tag such as 'TBC'. Enterprise Architect can assist by allowing the analyst to search across the requirements Packages for this tag and return a list of elements that require further work. A Model View could also be set up using this search to populate the view. The Discuss & Review window is also helpful because the information added is not part of the Requirement itself and does not contaminate the Requirement's notes with text that isn't part of the Requirements definition.

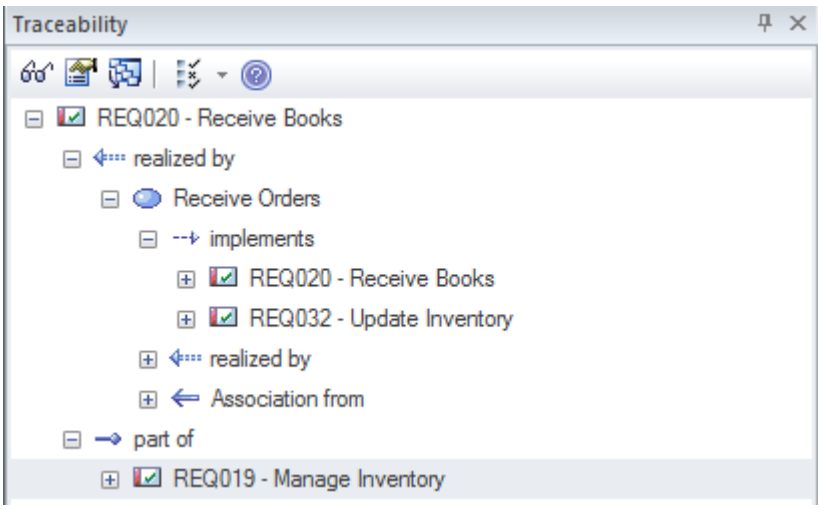


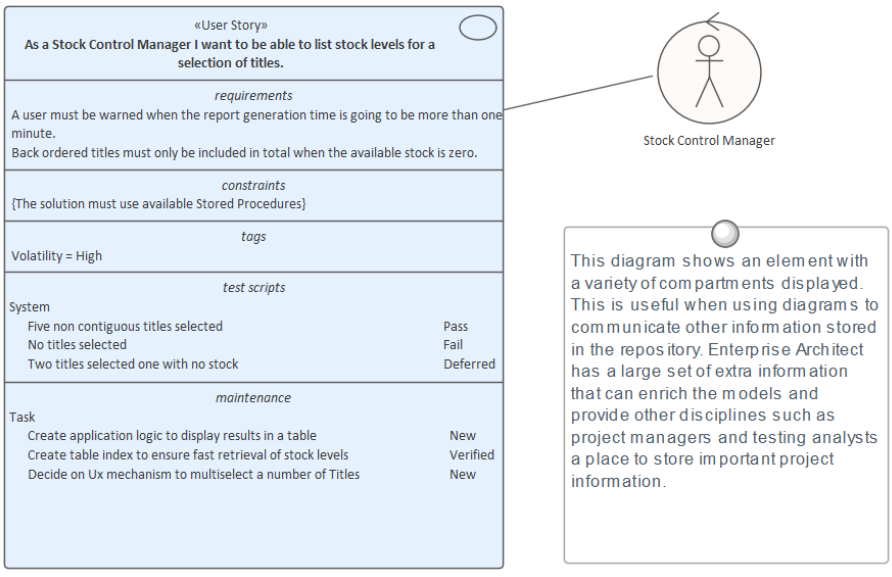
Current	A Requirement must be up-to-date and reflect the current knowledge and project status. Enterprise Architect can assist the analyst by allowing the sources of requirements to be modeled and the requirements themselves can be traced back to these artifacts so when the source is changed all the affected elements could be located.



Independent	The requirements should be independent of each other, and not have overlapping statements that conflict with each other or restate the same need. A degree of analysis will be required as there will inevitably be some overlap, but this can be kept to a minimum by creating requirements in hierarchies and working systematically. Enterprise Architect has a number of features that can assist with this, including the Relationship Matrix, which will help to identify overlap. The practical and flexible search function

	could also be used to identify overlapping or conflicting statements. 
Modifiable	This means that a requirement can be changed without there being the need to modify other related requirements. It also applies to a Software (System) Requirements Specification and requires that it can be changed easily. Enterprise Architect can assist with both these issues; the Requirements themselves can easily be located through the search facility, and the text and properties changed easily. The System Requirements Specification is automatically generated from the model, so by simply changing one or more requirements and regenerating the document it will be updated.
Traceable	A Requirement is a specification of a characteristic or behavior, and does not

	exist in isolation but is typically related to up-process entities such as stakeholders, business drivers and goals, and down-process entities such as Use Cases and components. Enterprise Architect allows elements to be traced in any direction and provides a number of helpful tools to visualize the traces, including the Relationship Matrix, the Traceability Window and the Requirements diagram itself. The Insert Related Elements facility can be used to automatically construct a diagram of traces. 
Unambiguous	A Requirement should only be able to be interpreted in one way. Requirements that are ambiguous can lead to a project being delayed, over budget or having the wrong functionality or behavior. Enterprise Architect can assist with ambiguity by

	helping analysts to record comments about the requirements, using the Discussion facility.
Verifiable	A requirement is verifiable if the implemented system or product can be tested to ascertain that the requirement has been met. Key to being able to achieve this is knowing which test must be run to verify a particular requirement. Enterprise Architect can assist by allowing the modeler to trace Test Cases back to requirements and to visualize their relationship in a number of ways, including the use of the Relationship Matrix. Test results can also be recorded directly inside Enterprise Architect. 
Necessary	Requirements should record a capability or behavior that is really needed or that

	specifies that the system or product should comply with constraints such as standards. Enterprise Architect can assist by allowing the modeler to relate each requirement back to its source and using the Relationship Matrix; requirements that have no source will be obviously identified as unnecessary or needing further investigation.
Feasible	A requirement that cannot be implemented will mean that the need of the stakeholder will not be met. It is best to identify these requirements as quickly as possible so as not to disappoint the owner of the requirement. Enterprise Architect can assist by allowing analysts, architects, designers and developers to discuss the requirement and determine its feasibility using the Discuss & Review window.

Business Context for Requirements

Requirements don't appear in isolation but are usually defined or discovered in the context of a business problem or opportunity that has been defined in one or more business documents. These documents and the information they contain can be included in the models and provide an important anchor point for Requirements.

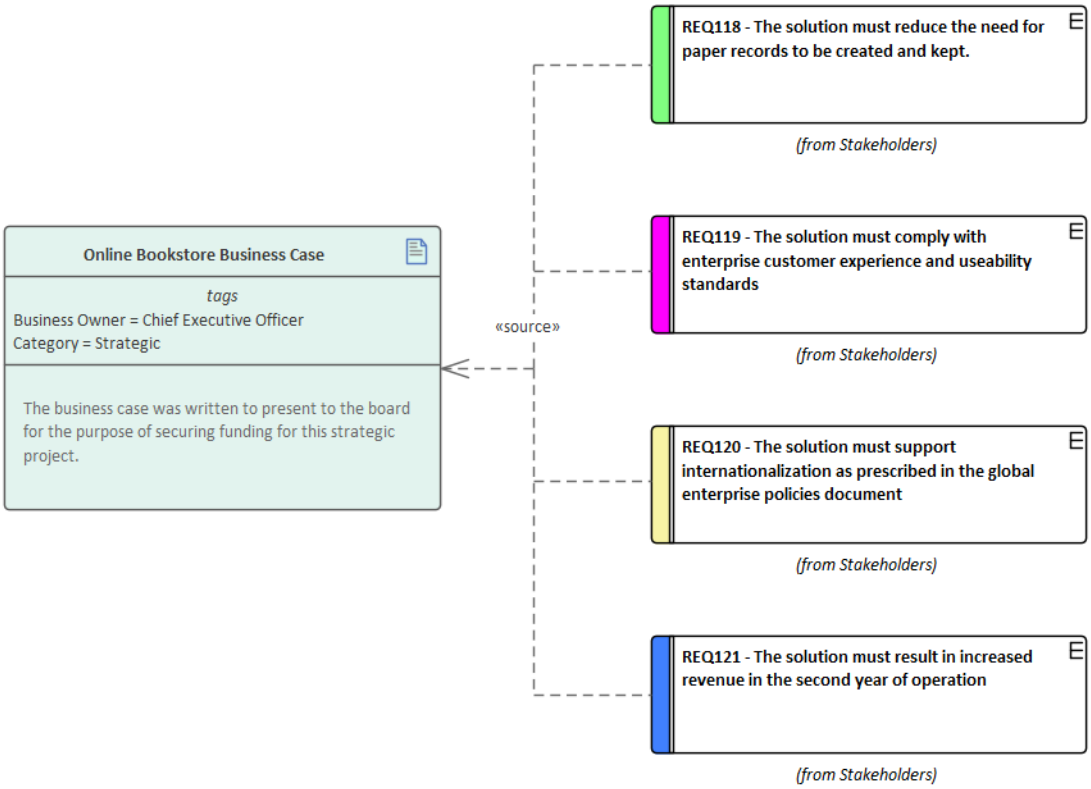
Business Case

The Business Case is a high level document or argument that attempts to articulate the reasons for initiating a project. It is an important artifact for the requirements analyst because it will typically contain information describing business value, drivers and business and technical risks. It places the endeavor in the context of other functions in the business and describes the solution options at a high level. It is an important source for requirements and should be included as an artifact in the model.

Requirements Sources

This diagram shows an external document, "Online Bookstore Business Case", modeled as an artifact. Requirements have been linked back to this artifact, to indicate that the source of the requirement is this document. If the document is subsequently updated, the requirements derived from it are easily located. The Business Case document artifact has a number of Tagged Values indicating properties of the document.

Hyperlinks to external documents can be created by simply dragging and dropping a document file onto a diagram canvas.



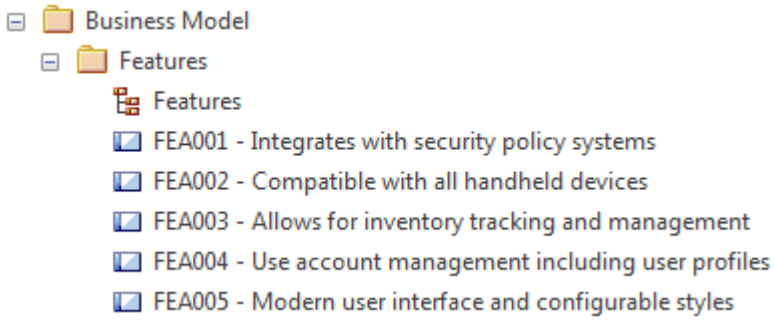
Drivers and Goals

Business Drivers and Goals are often documented by high level strategic thinkers, such as business or enterprise architects. Drivers define resources, processes or constraints that are vital to the operation of the organization, and Goals describe the position that the organization wants to attain.

They are typically enterprise level concerns and so should be modeled above the level of individual projects. They often exist in high level documentation, and even when they aren't clearly articulated at the organization level, an analyst can mine them from previous project documentation such as a Vision document, and model them in an enterprise Package above the project Packages in the repository.

Vision and Concept of Operation

While the Business Case describes the business reason for initiating the project, the Vision typically elaborates the opportunity or problem in more detail, describing the business context, the market position, key stakeholders and requirements, solution choices and constraints. The Vision is more often than not created prior to the team being assembled and can be a great source of requirements information. The required system functionality is often expressed using Features.



Enterprise Architect has a wide range of tools and element types that can be used to model the contents of the Vision document, including Users, Stakeholders, architecturally

significant Use Cases and Requirements, Constraints and Deployment Environments.

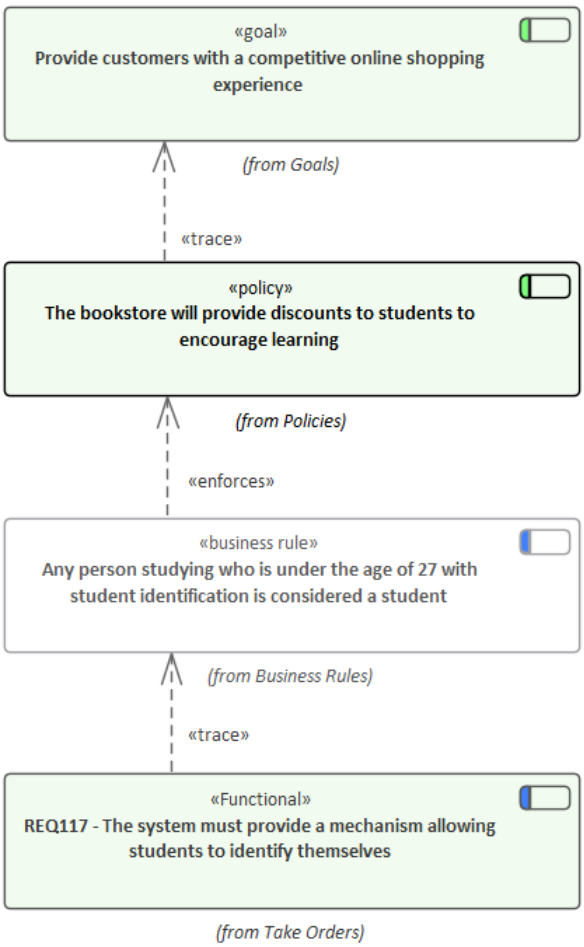
Policies and Business Rules

A Policy is a high level principle or statement of intent typically defined and managed by a governance body; a Business Rule is an implementation of the Policy. They are not strictly requirements and are often defined at the enterprise level rather than the project level, which facilitates their reuse across multiple projects. Policies and Business Rules can be modeled using stereotyped Requirement elements, and business and system requirements can be traced to them from individual projects. There is some overlap with regulatory and safety requirements, which some methods consider to be types of Business Rule. Enterprise Architect supports the modeling of Polices and Business Rules using stereotyped Requirements, but also has a Business Rule Modeling capability that can create executable code for a variety of languages.

- Business Rule Modeling is available in the Unified and Ultimate Editions of Enterprise Architect

Business Rules and Policies

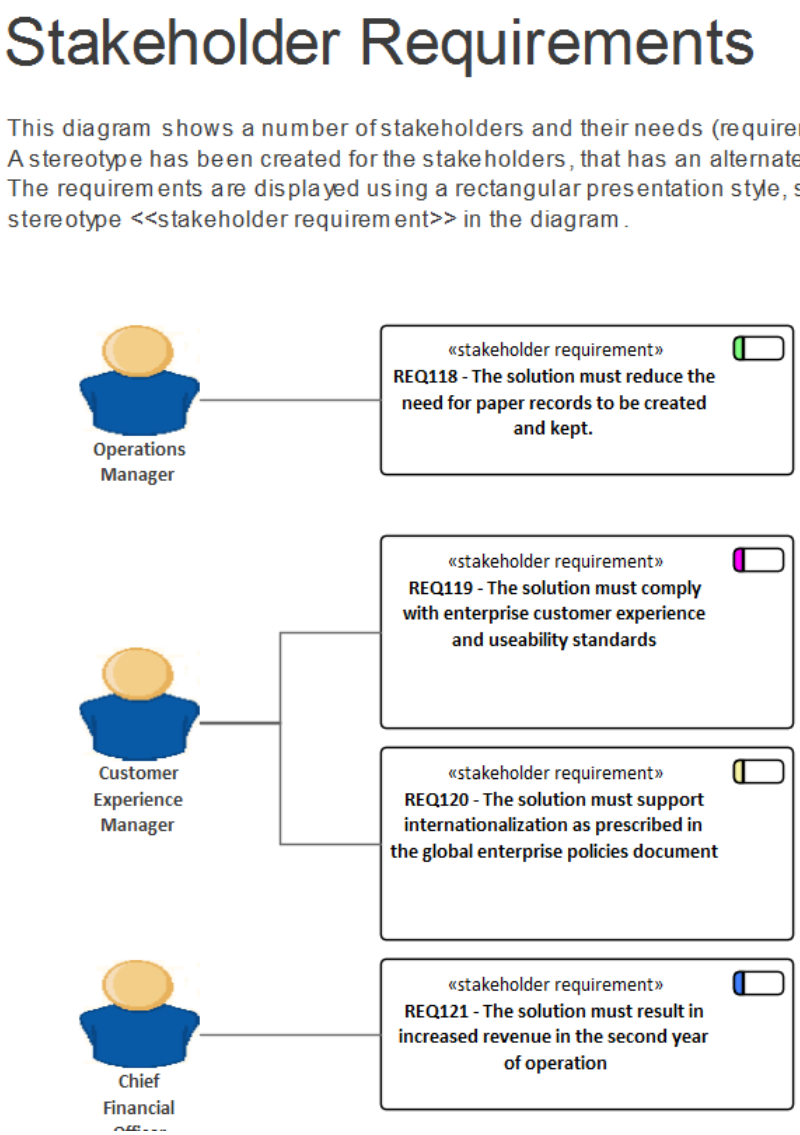
This diagram shows the way that goals, policies and business rules can be modeled using a stereotyped requirement element.
The policies express a guide that cannot be directly enforced while the business rules act to enforce the policy.



Stakeholders and Their Concerns

Stakeholders typically have the same set of concerns regardless of whether projects are running or not. A Security Manager will for example be concerned about the vulnerability of sensitive organizational data, a Customer

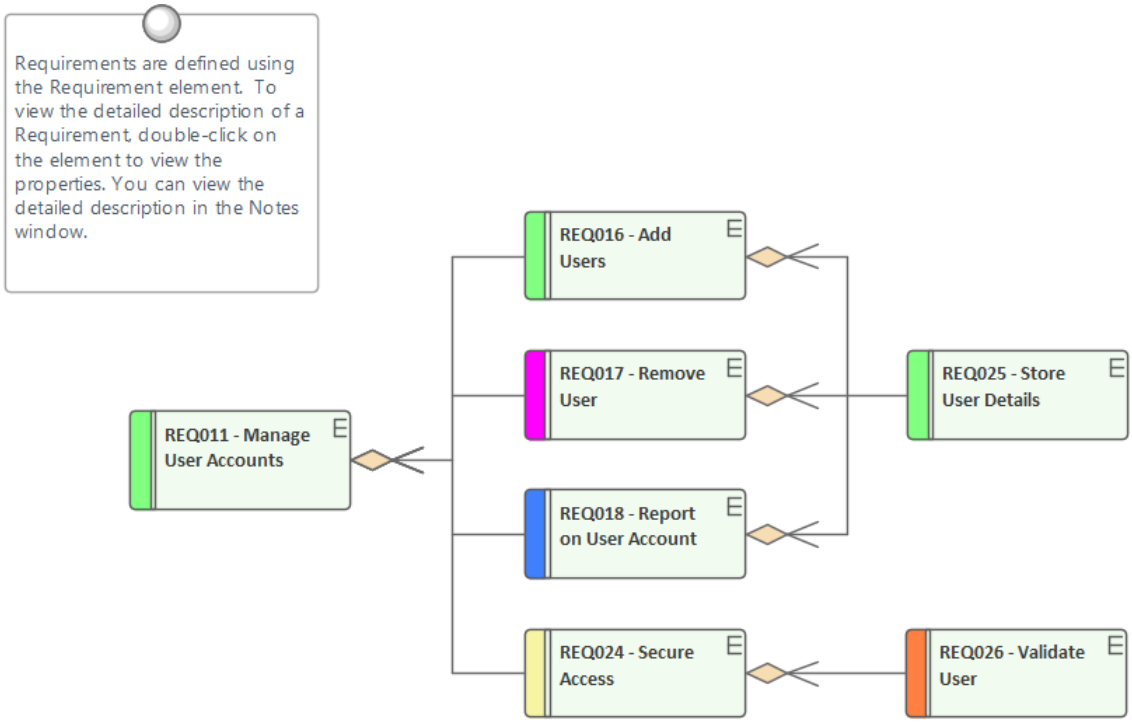
Experience Manager will be concerned about speed of access and a Chief Financial Officer will be interested in return on investment. These concerns can be modeled at the enterprise level as they are generic and independent of individual projects. They will provide a source of understanding for project level requirements and will help identify gaps in the requirements landscape. Enterprise Architect can be used to model Stakeholders using a stereotyped UML Class and these high level concerns can be modeled using a requirement stereotyped as a Stakeholder Concern.



Requirements Diagram

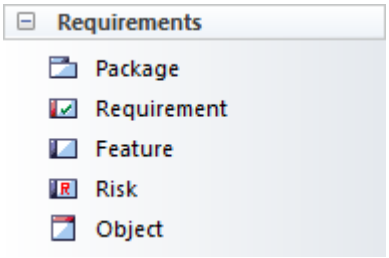
A Requirements diagram is one of Enterprise Architect's extended diagram types. It allows an analyst to model requirements visually, showing how requirements relate to each other and how they connect with other elements in the model such as Business Drivers, Constraints, Business Rules, Use Cases and design Components. The Requirement will be the main element seen on these diagrams; it has a name, a description and a series of properties (called attributes in some literature) such as status, complexity, difficulty and author. Enterprise Architect is designed to be a flexible tool and allows requirements to be created directly in the repository without the use of a diagram, but the diagram has proven to be a useful tool to express the important role requirements play in the development process.

Example Diagram



Requirements Toolbox

You can create elements by dragging them from the 'Requirements' pages of the Diagram Toolbox onto the diagram canvas. Connectors can also be selected from the Toolbox and dragged between elements in the diagram or by using the Quick Linker. This table lists the elements available from the 'Requirements' toolbox but it is important to remember that other elements such as Use Cases and Components can be added to the diagram by opening other Toolbox pages - click on  to display the 'Find Toolbox Item' dialog and specify the element name.



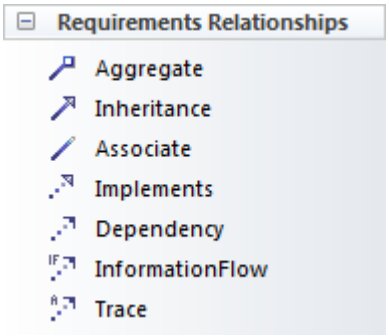
Requirements Toolbox Elements

Element	Usage
Package	Packages are container elements that can be used to group requirements and other elements into sets. They are not requirements themselves but act as a grouping mechanism; analysts should take care that the Package is not a high level requirement.
Requirement	The Requirement element is used for modeling a condition or capability that a system must have. The type of the requirement can be set but there are also a number of types of requirement such as Functional, User and Architectural requirements available from the 'Extended Requirements' page of the toolbox. They are useful for modeling a range of other entities, such as Rationales

	and Assumptions.
Feature	A Feature is a characteristic or property that the system must have to meet its business requirements. They are typically high level properties that represent a group of requirements.
Risk	A Risk is a condition that could cause the disruption, loss or compromise of a system. The element can be used to model both technical and business risks and can be connected to one or more elements.
Object	Objects are useful for modeling any of the entities that are discussed during a requirements elicitation workshop or while reading through project documentation. Formally they are Instances of Classes and when analysis is conducted a Domain Class can be derived from one or more objects.

Requirements Relationship Toolbox

You can create relationships by selecting the corresponding icon in the Toolbox and dragging between any two elements in the diagram canvas, or by using the Quick Linker. This table lists the relationships available from the 'Requirements Relationship' page of the Toolbox but it is important to remember that other relationships such as Composite can be added to the diagram by opening other Toolbox pages.



Requirements Relationships Toolbox

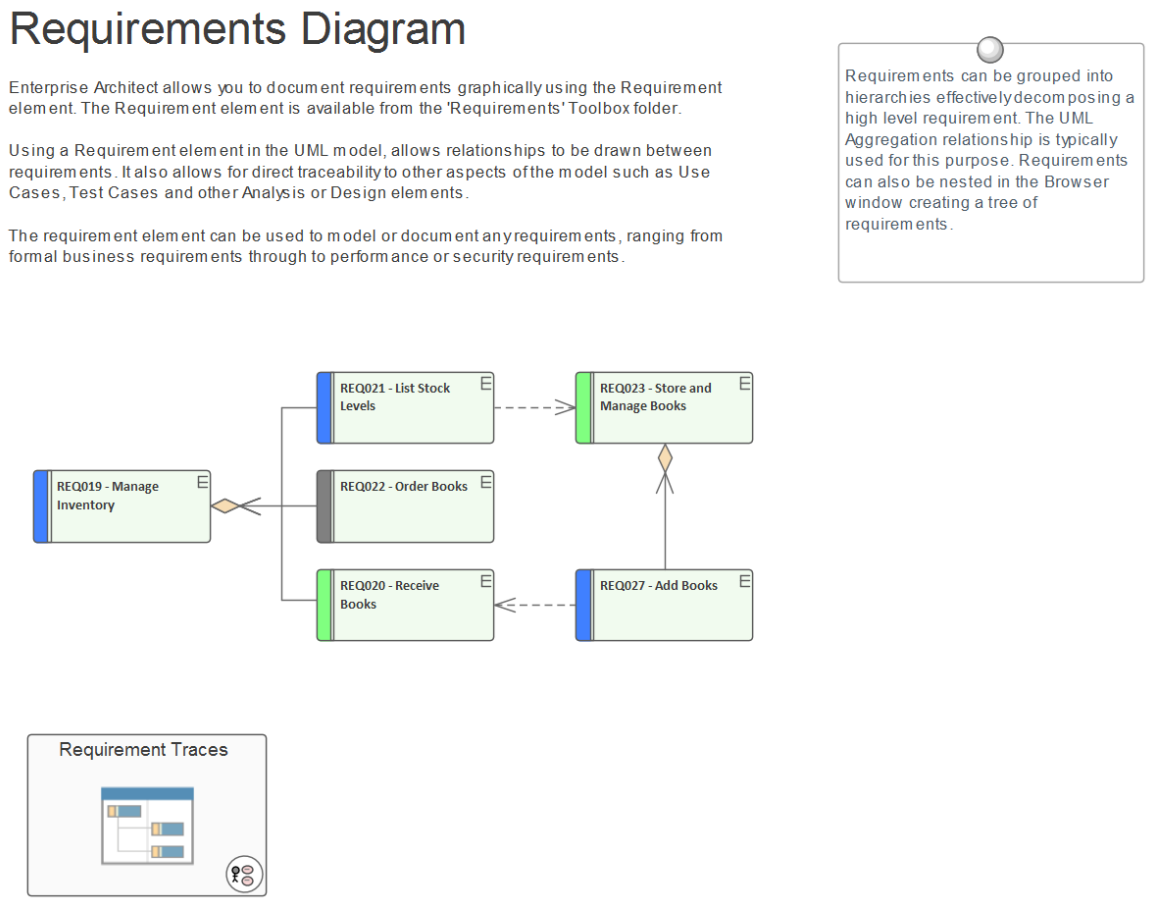
Connector	Use
Aggregate	Used to show that a requirement (diamond end) is made up of another requirement (tail end). This allows hierarchies of requirements to be created.
Inheritance	Used to show that an element (triangle end) is a more generalized version of another element (tail end). The relationship is used between Classifiers such as Use Cases, Classes, Artifacts and

	Components.
Associate	Used to show a semantic or structural relationship between two elements.
Implements	Used to show that a model element implements a Requirement. Typically it would be used by an architect or designer to indicate that the need expressed in the Requirement would be met by a particular module, Use Case or Component in the system.
Dependency	Used to show that a Requirement (tail end) relies upon another element (arrow end).
Information Flow	Used to show that data flows between two elements in a Repository. The type of data can be represented as Information Items that can be selected from any part of the model. They could be used to show the Requirement that information flows between the proposed system and a supplier's system or to represent a Constraint that two Components must communicate via a certain protocol.
Trace	Used to show that an element (tail end) is

	more elaborated in the model than the element at the arrow end. So a User Requirement could be traced to a Stakeholder Requirement or to a Business Goal,

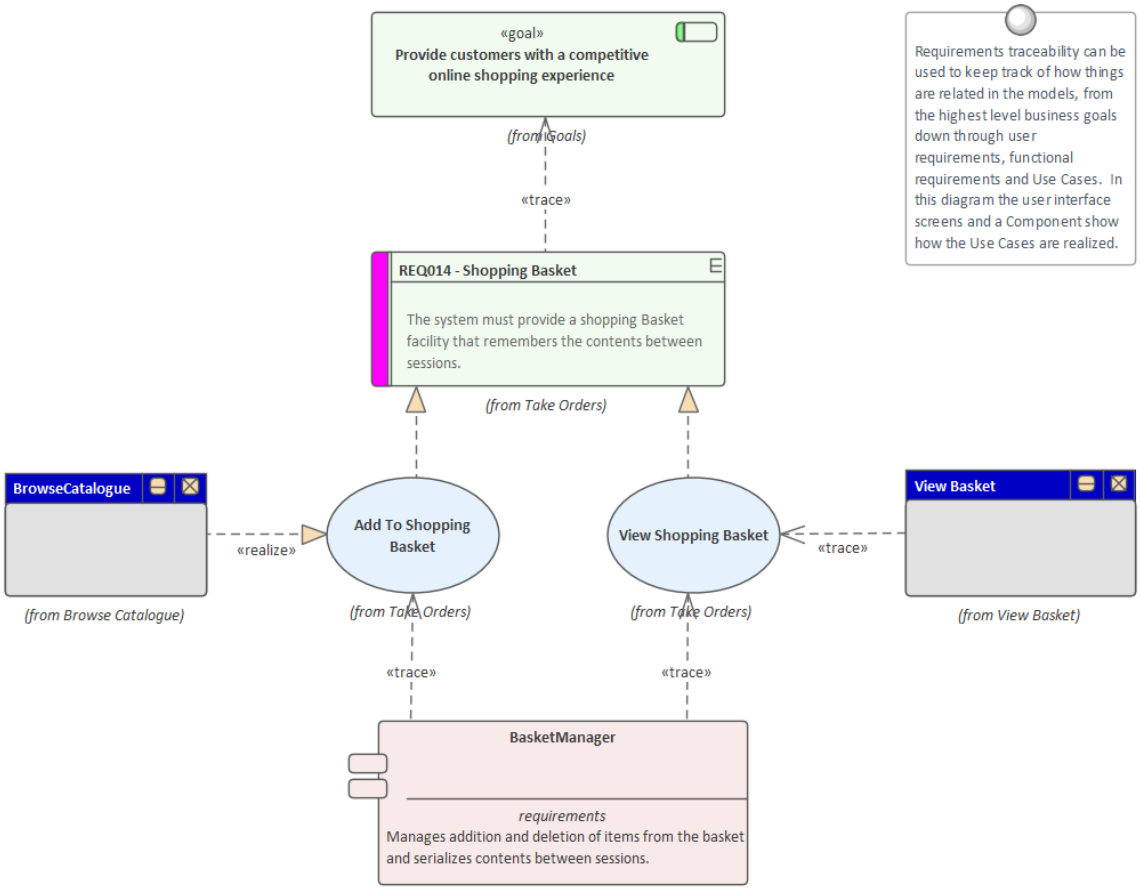
Example Diagram - Hierarchies

This diagram shows how requirements can be connected into hierarchies thus allowing high level requirements to be broken down to verifiable requirements.



Example Diagram - Traces

This diagram shows how Requirements can be connected to other elements in the model, displaying traceability.

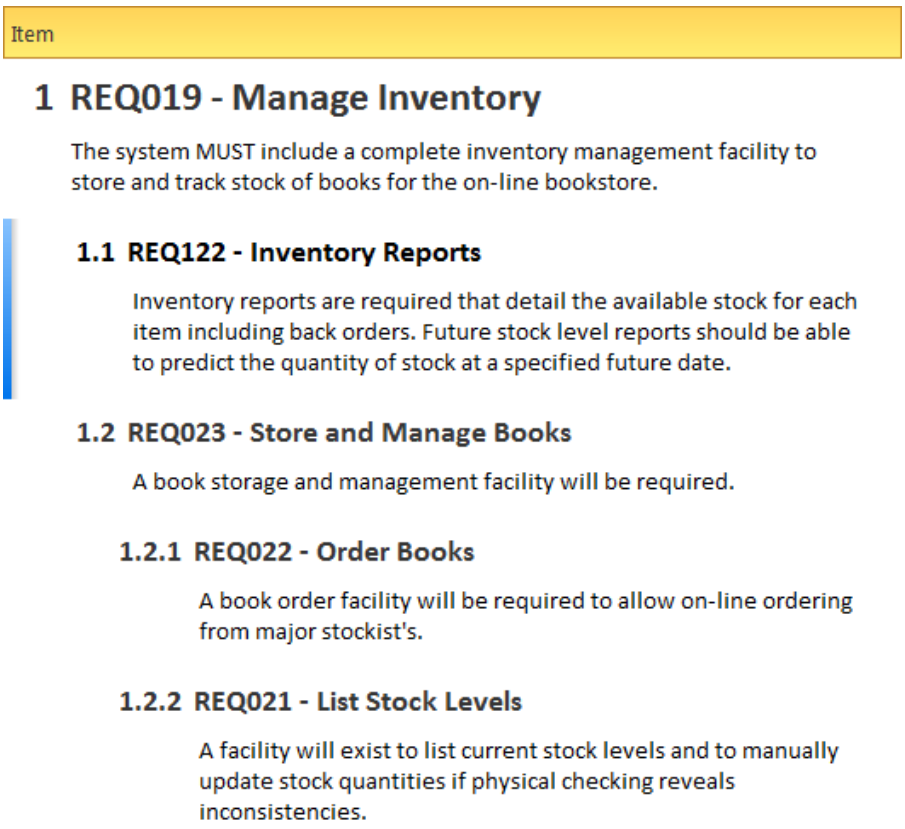
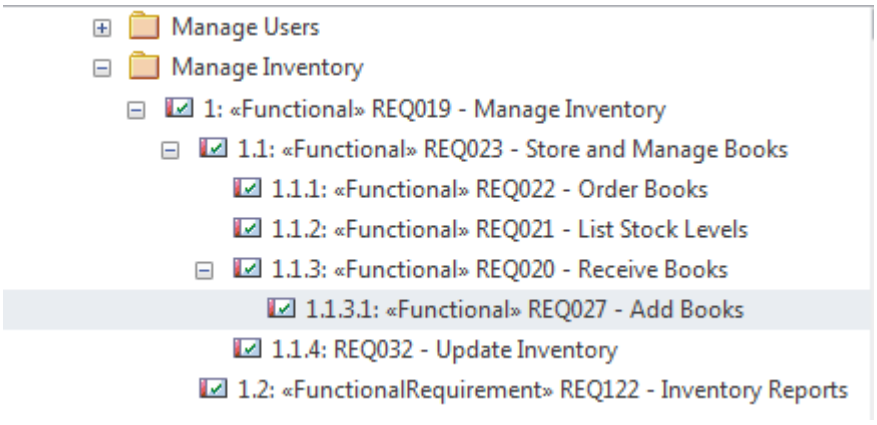


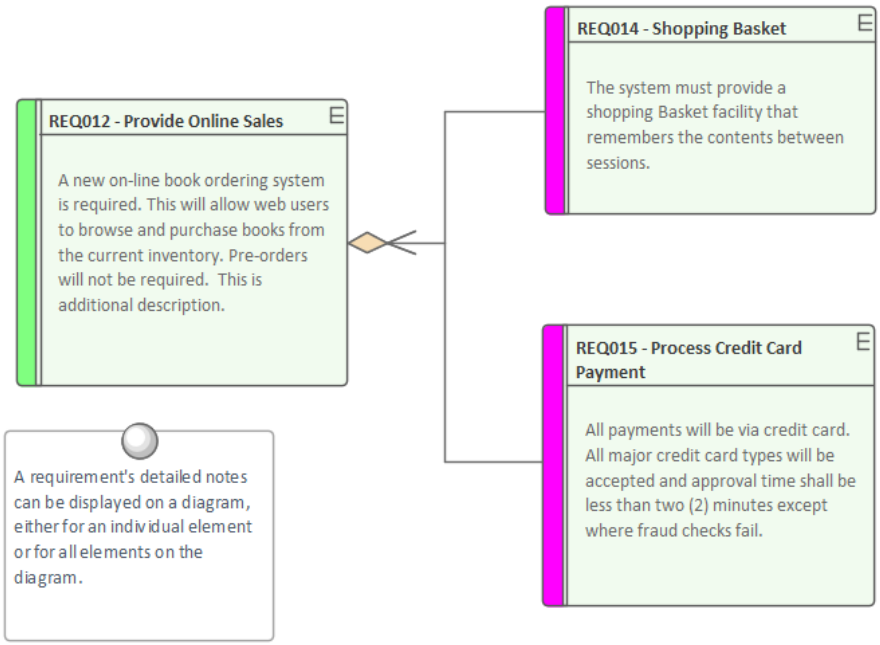
Creating and Viewing Requirements

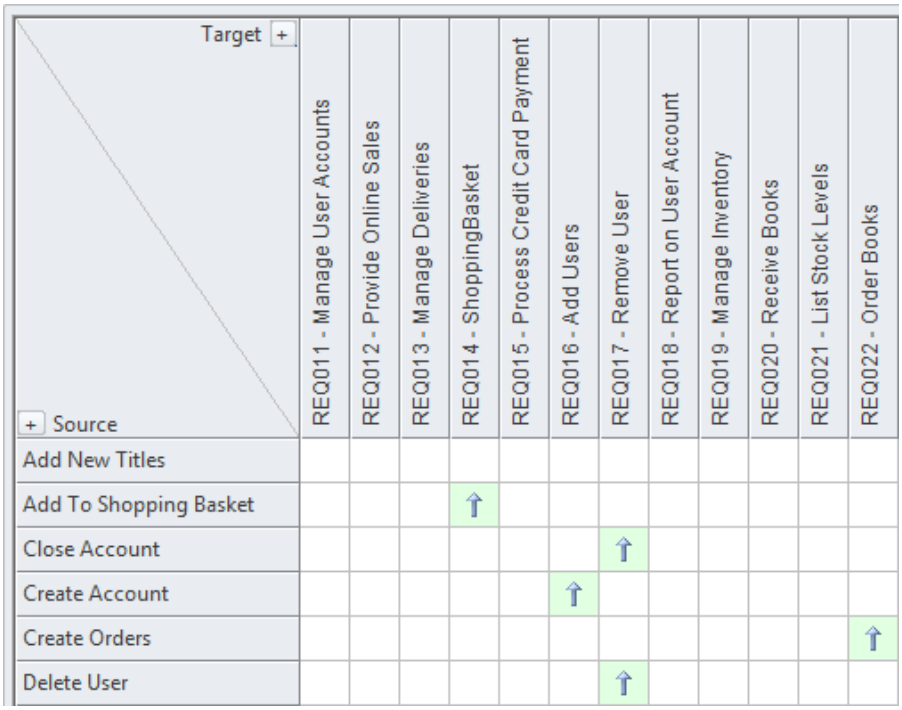
Enterprise Architect is a sophisticated and flexible modeling platform and the tool offers a rich set of features for both the development and the management of requirements for any size project in any domain and using a variety of processes. Requirements can be created in a variety of ways and then visualized in a series of windows and dialogs that make it easy to develop and manage them and to communicate within the team and to the business customers.

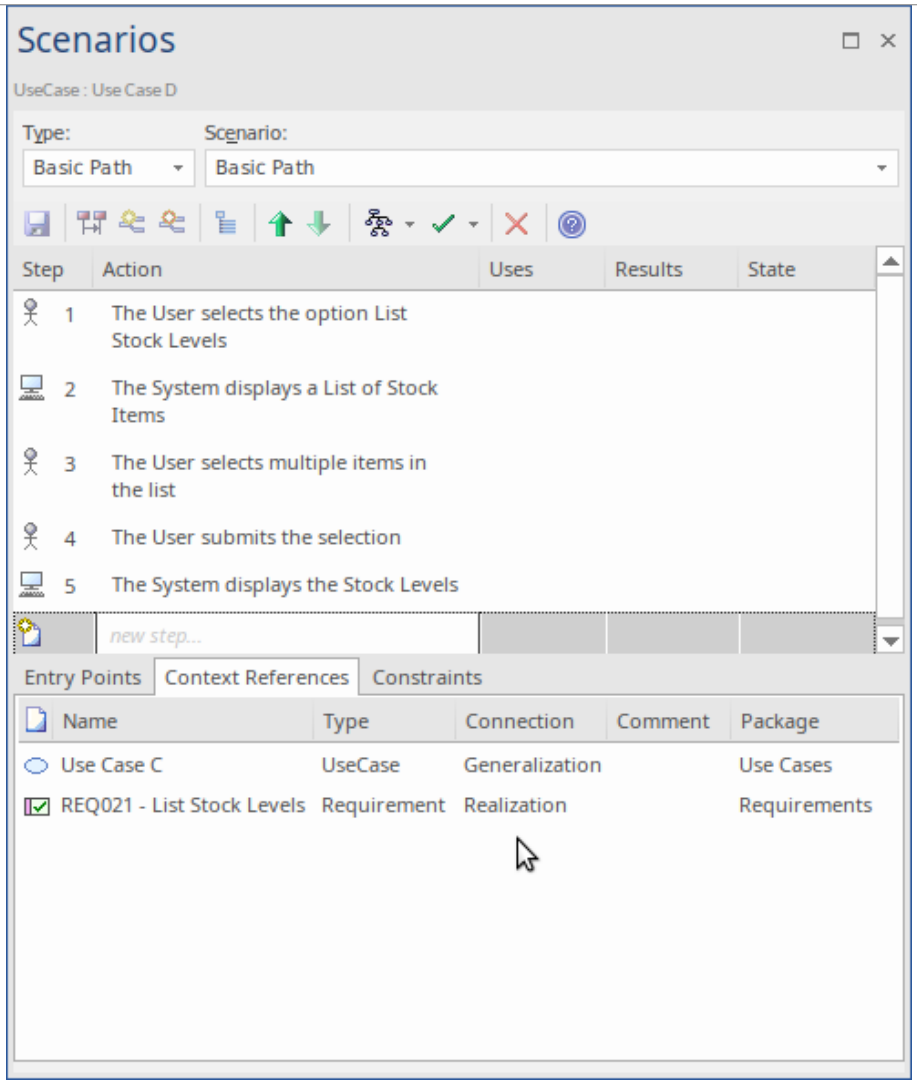
Viewing Requirements

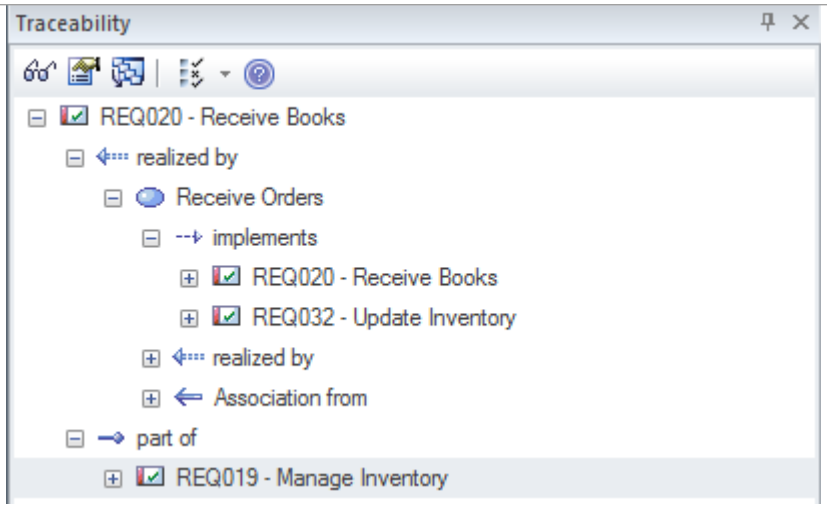
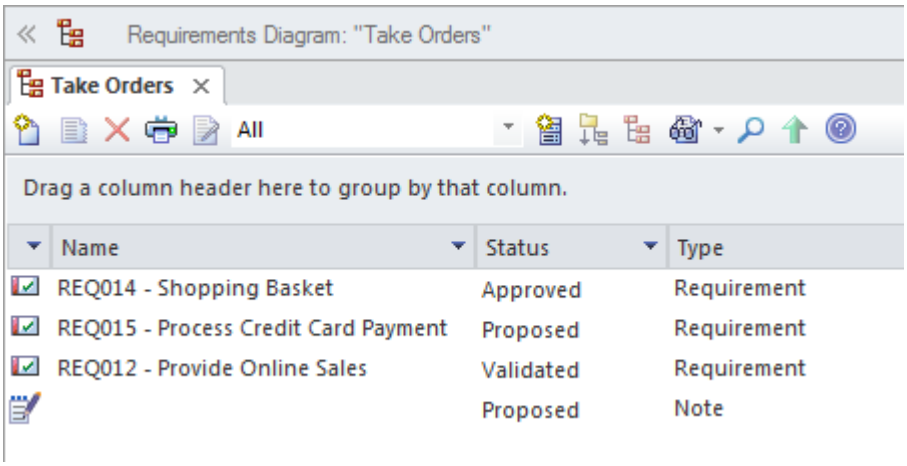
Facility	Description
Specification Manager	Shows Requirements (and other element types) in a simple text format, and helps the modeler to create, edit and manage these elements. The Specification Manager will be the preferred tool for many Requirement Analysts as it allows the modeler to work in a familiar spreadsheet-like interface, to edit requirement properties such as Status, Priority and Difficulty using drop-down lists, and to add notes to the Requirements without needing to draw

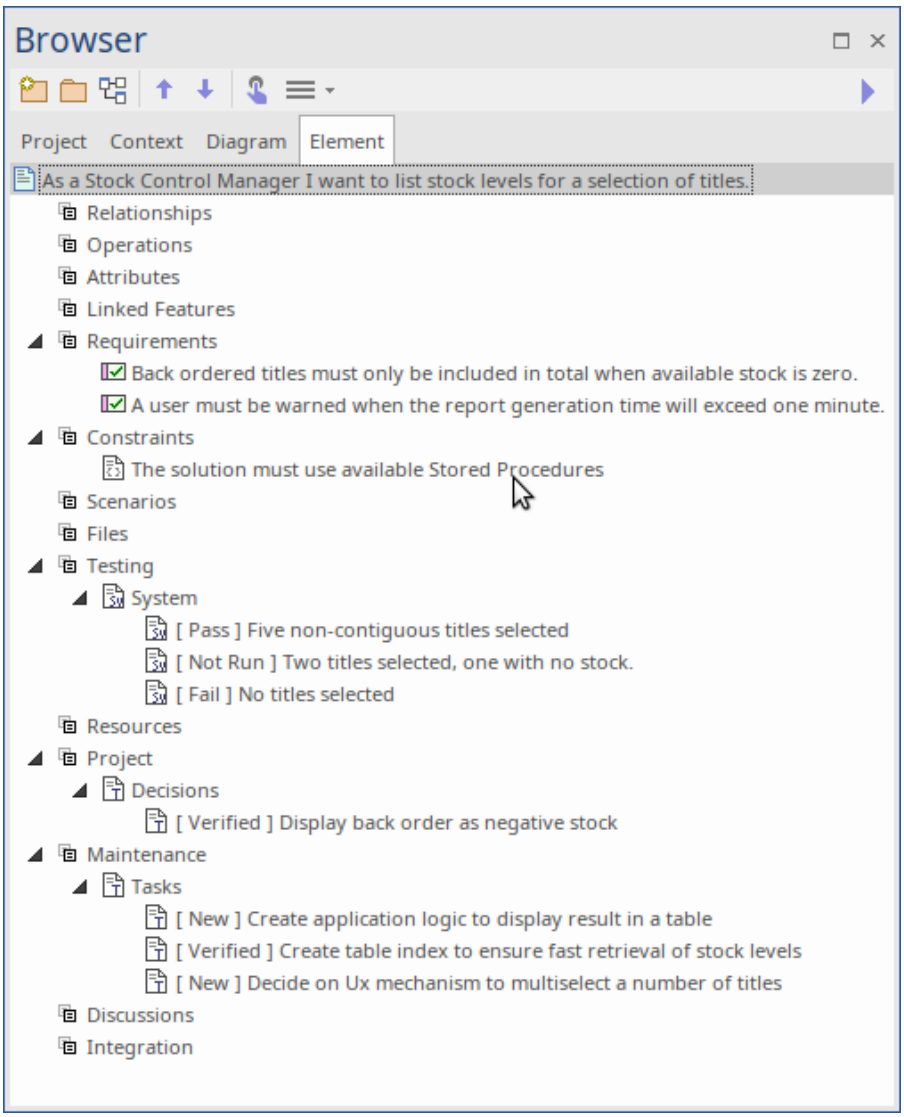
<table>
<tr><td></td><td>diagrams.
 
</td></tr>
<tr><td>Browser Window</td><td>Shows the content and structure of the repository and allows requirements to be viewed in a hierarchy of Packages, elements and diagrams.
 
</td></tr>
<tr><td>Requirements Diagram</td><td>Shows the arrangement of a group of Requirements and other elements, and</td></tr>
</table>

	can show whether the elements are in the same Package or different Packages. It is an effective way of presenting requirements information because connectors can be created to show how a Requirement relates to other elements in the Repository including other Requirements. 
Relationship Matrix	Shows one group of elements on the horizontal axis and another group of elements on the vertical axis with markers indicating if they have a relationship and an arrow showing the direction of the connector. Relationships can be created directly in the matrix and these will be displayed on diagrams containing the source and target elements. The Relationship Matrix is a useful tool

	to quickly visualize gaps or missing requirements and is particularly useful for project managers and business stakeholders who might be less familiar with diagrammatic representations of Requirements and formal languages such as the UML. 
Scenario Builder	Used to create Scenarios, Constraints such as pre-conditions and post-conditions and to detail the steps of Use Case Scenarios. It can also be used to create behavior diagrams that allow the steps in a Scenario to be visualized and used as a target or source for connectors.

	
Traceability Window	Used to display the hierarchy of elements based on their relationship to other elements. While the Browser window will give a structural view based on containment in a name space the Traceability window displays related elements regardless of their location in the repository.

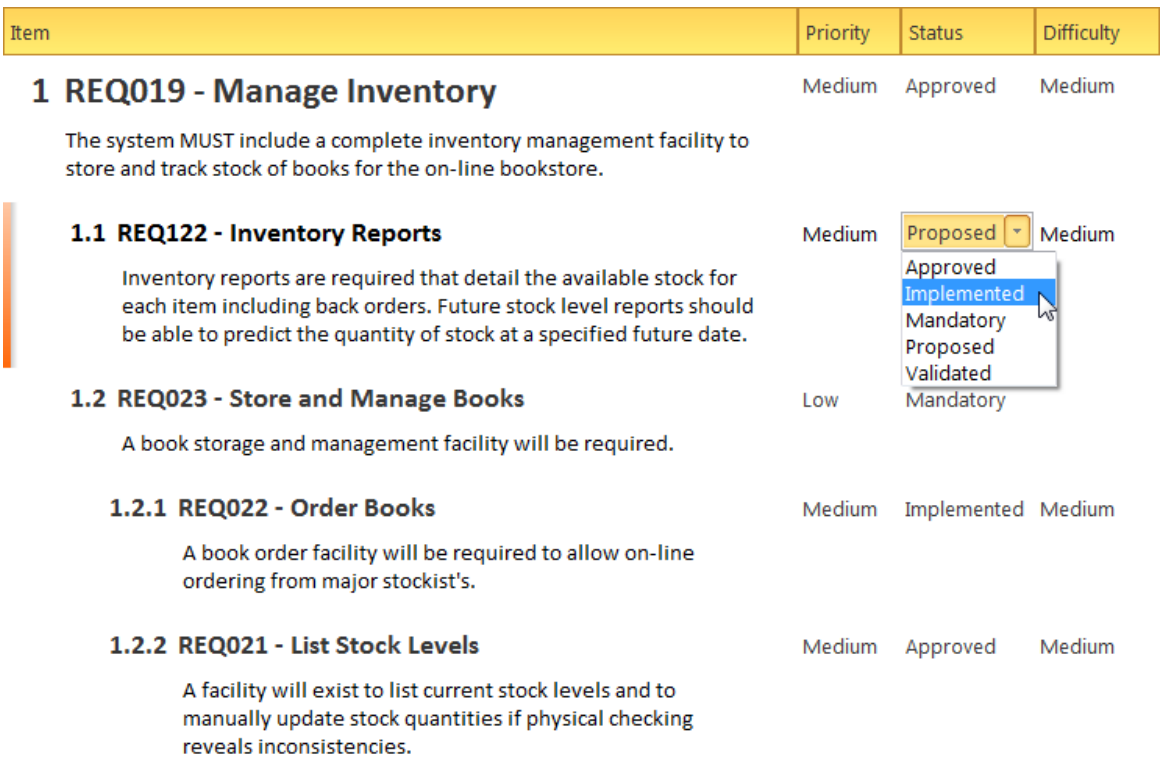
	
Diagram List	Lists the elements in a diagram, filtered and sorted according to the settings you define; shows all or selected default properties of each element. The properties can be edited in-line for each Requirement and new Requirements can be created in the diagram list. 
Package Browser	Lists the elements in a Package, filtered and sorted according to the settings you define; shows all or selected default properties of each element.

Element tab of the Inspector window	Displays a selected element's Attributes, Tagged Values, Constraints, Internal Requirements, Relationships, Maintenance Items, Testing, Project Management items, Files, and more. It is a versatile way of displaying this information in one place without the need to open up other windows. 
Model Search	Enables you to locate Requirements in general in the model, or specific

	Requirement elements, according to the search criteria you use.
Model Views	Enables you to maintain links to commonly-used elements, and to rapidly show developments and changes in (Requirement) Package contents through either reports or slide shows of selected diagrams.

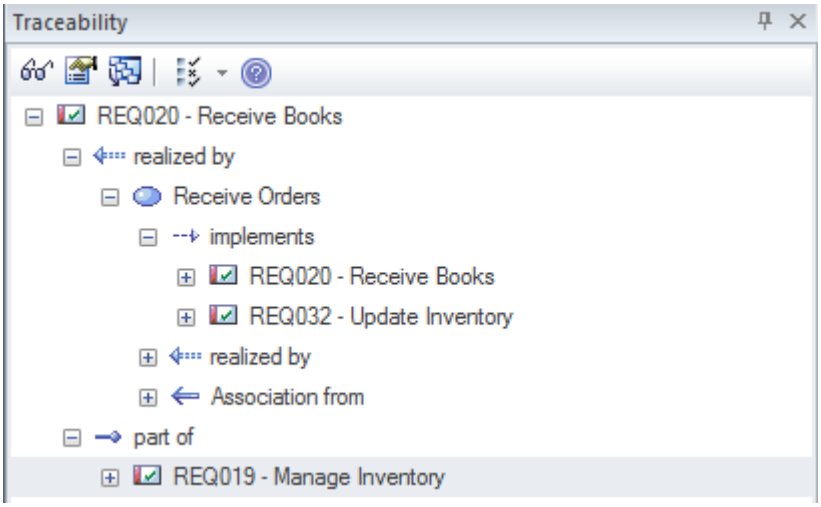
Requirement Development

Requirement development includes all the activities and tasks associated with discovering, evaluating, recording, documenting and validating the requirements for a particular project or program of work. Requirements are discovered, analyzed, specified and verified, and Enterprise Architect has a wide range of tools and features to assist the Requirement Analyst as they develop requirements. The centerpiece for requirement development is the Specification Manager, allowing the Analyst to enter, view and manage requirements in textual form in a spreadsheet format.



The Specification Manager can be used in conjunction with a platform of other tools, such as diagrams, the Traceability window and the Discussions facility. These windows

provide other views of the requirements, giving the modeler and the viewer a deep understanding of how a requirement relates to other parts of the repository, and providing detail not visible through the Specification Manager.



Elicitation

Elicitation is the process of information discovery, the information forming the precursors to requirements. The information will typically be raw and often heterogeneous, and it will not be until the requirements analysis phase is performed that true requirements will be able to be derived from it. Elicitation will take many forms, and all of the skills of the analyst will be needed to determine which documents, tools, people and processes to examine to discover the information. Some of these information source locations are listed in this table.

Location of requirements information

Requirements are not handed to the analyst on a plate but rather will be discovered in a variety of places including: by observing stakeholders performing their work, in business documents and a variety of other locations.

Source	Description
Observing Users	Observing users perform their work is a helpful way of gaining information about Requirements and often reveals details that would not be discovered by user interviews.

Business Documents	A number of business documents such as the Business Case, Vision or Concept of Operation will provide a source for Business Requirements and should be discovered and included as sources of Requirements.
Stakeholder Workshops	Getting all the important stakeholders into a workshop is a useful and productive way to get information that will help with the Requirements definition. Typically there are fertile and robust discussions that provide the basis for deriving Requirements.
Current System Issues	There will often be documented information recording errors, faults and issues with the incumbent system that will provide the basis for Requirements for the replacement system. Care needs to be taken that any Requirements derived from these list are owned by a stakeholder and that there is a business need to include them in the new system.

User Observations

Observing users perform their work is an effective and unobtrusive way of gaining an understanding of the tasks they carry out and how they use information and other software and hardware devices to achieve an outcome. Even if the processes that support the planned system will be different, the observations of the current processes will provide a useful context for discussions. It will also help the analyst empathize with the user and can result in a deeper understanding of the issues they face and provide the basis for the discovery of potential solutions. An analyst will often discover unmentioned documents, checklists and clue cards that can help illuminate the process. Equipped with a mobile phone or camera, it is also useful for the analyst to take photographs of the user working, which will help in the requirements analysis phase.

Enterprise Architect supports the modeler in representing files such as photos and scanned documents directly in the model, creating a rich and expressive representation of the user at work. There is the option to represent these as an Artifact (which, with a single key stroke (F12), will launch the file) or to use a hyperlink or even to include the image itself in a diagram.

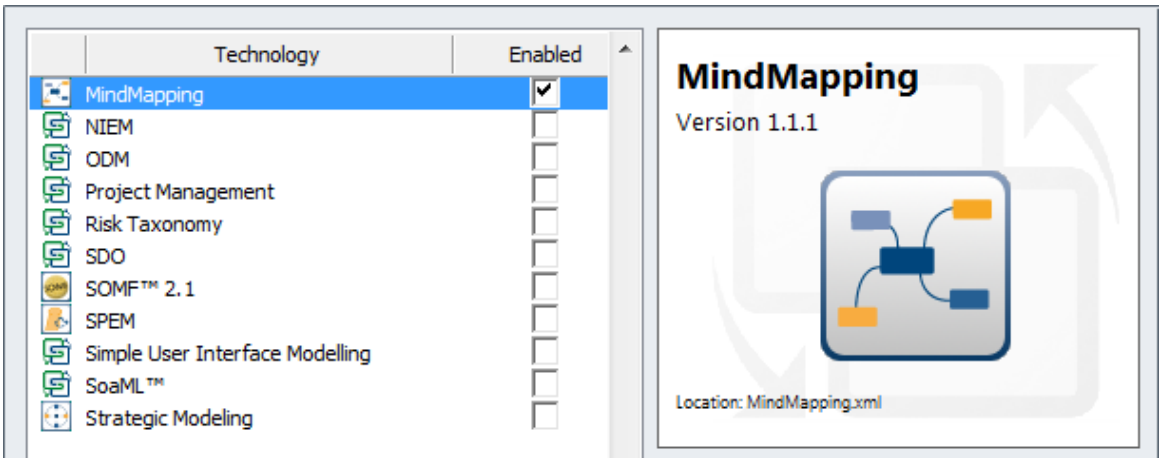
Stakeholder Workshops

The requirements analyst or business analysis is charged with the difficult task of eliciting requirements, which necessitates excellent communication with the stakeholders, including the customer and the analysis team. One very successful way of facilitating the elicitation of the stakeholders' needs is to run a workshop with all the key stakeholders present. The analyst's skills as a communicator, diplomat and mediator are important to create a collaborative and respectful environment conducive to the exploration of the stakeholders' needs and concerns. It is imperative that the analyst uses terminology that the stakeholders understand and displays an understanding or a willingness to learn the elements that make up the domain.

There is sometimes a misconception that what will be articulated is a set of clearly defined requirements that can be entered into the tool as Stakeholder Requirements; this is far from the reality of what happens. Stakeholders will typically articulate a wide range of ideas, including Policies, Business Rules, Data definitions, Project Management Constraints, Functional Requirements, Business Requirements, existing system problems and even suggested solutions. Even when an external consultant is used to run these meetings the analyst will not have time to categorize all of these statements in the meetings. What is needed is a way for the scribe who is tasked with documenting the statements to get them into the tool without any concern for what type of information is being recorded. Having them

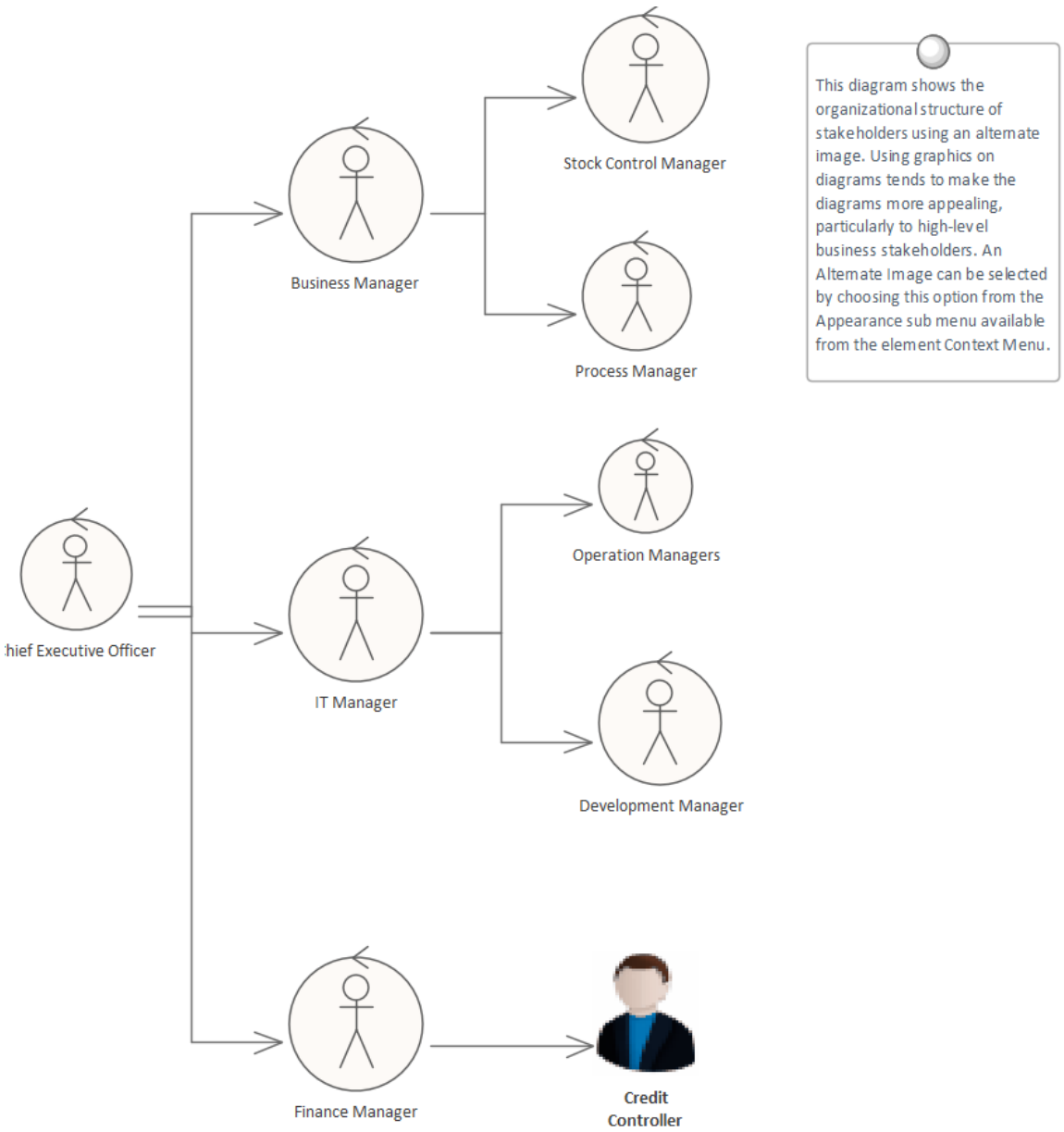
recorded in the tool rather than scribbled in the analyst's notebook is best practice because it allows them to be displayed during the meeting and for stakeholders to see each others' comments.

Enterprise Architect has a number of facilities that can help with these workshops. One method that is very effective is to use the MindMapping diagram to record the stakeholders statements, which is very effective because it is a well known method and doesn't introduce any of the formality that comes with modeling languages such as UML.



As important terms are uncovered they could be entered into the Project Glossary, and even if there is not time to discuss and debate the agreed meaning, the words will act as an initial list of important entities in the domain. Alternatively, the terms could be created in a Domain Model and related to each other with connectors that describe the important relationships between the terms.

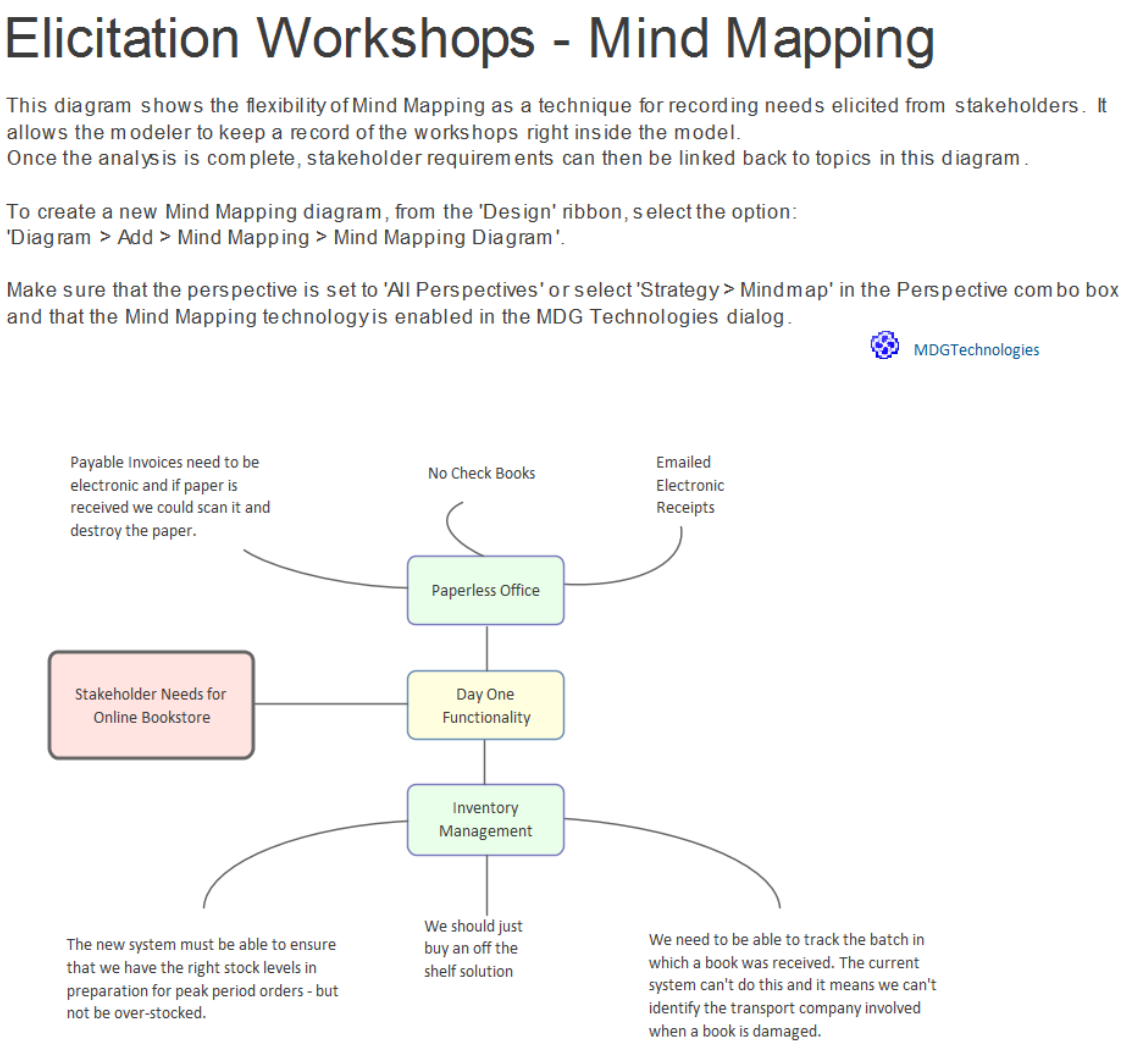
The stakeholders can also be modeled and their organizational relationship to each other can be described in a diagram. This is a useful technique that allows key stakeholders to locate themselves in the models, which creates buy-in.



Mind Mapping Diagram

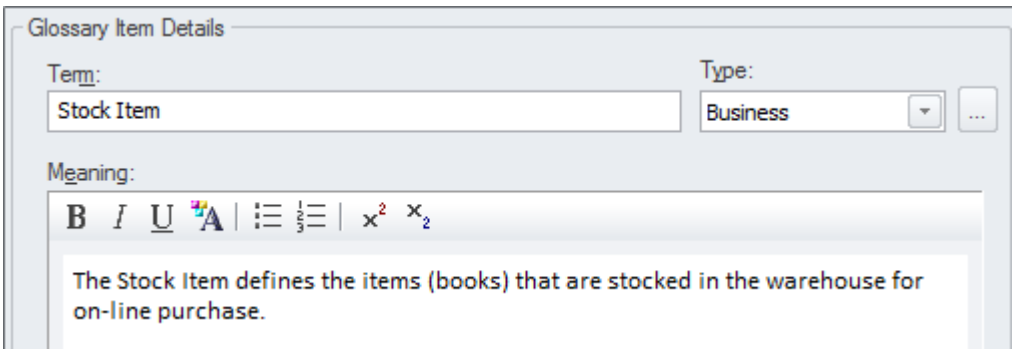
A Mind Mapping diagram can be used to record the stakeholder's statements during an elicitation workshop. The statements are not categorized but simply recorded and later during the analysis phase of Requirement's development they can be converted to the appropriate elements or

retained and the Requirements can be traced back to the topics effectively creating a record of how the Requirement was derived. This is a useful technique that shields the stakeholders from needing to know the modeling languages and allows them to concentrate on articulating their needs, it also frees the analyst up from concerns about which element to use to model the statements. This step is usually performed in the analysis phase of the Requirement's Development process.



Glossary

Prior to a workshop an analyst can populate the Project Glossary with the existing terms and their meanings that have been gleaned from reading project documentation such as a Business Case or Vision Document. During the workshops, as new terms are uncovered they can be added to the Glossary and their definitions can be discussed and entered or deferred until later in the analysis phase.



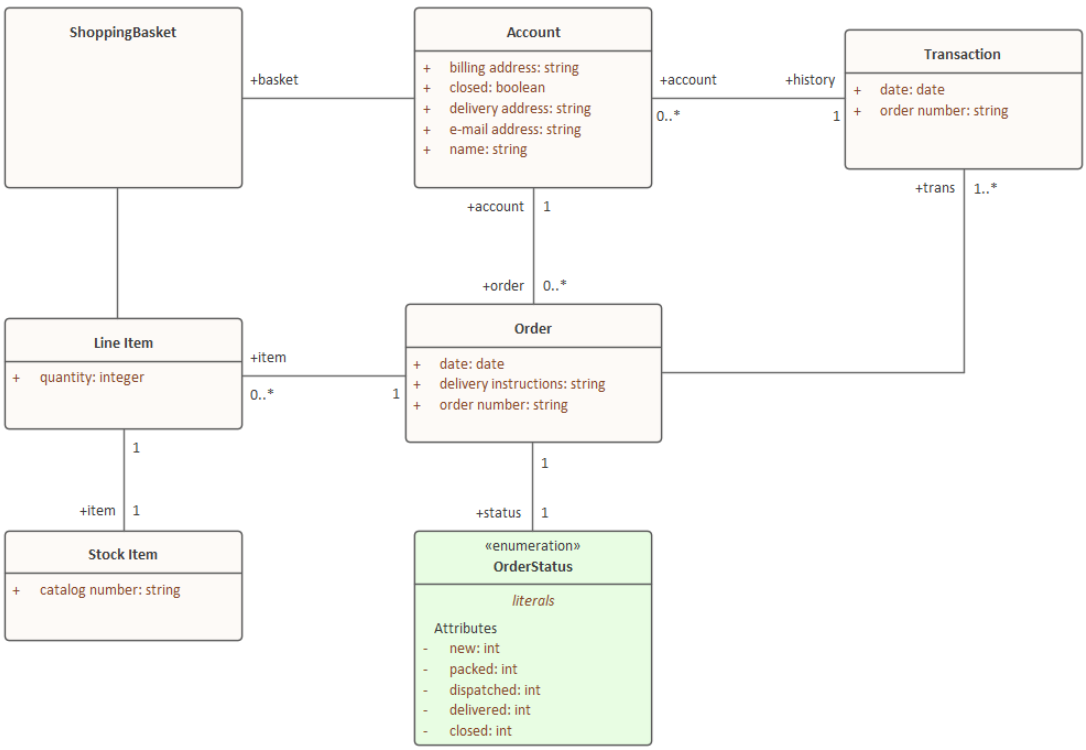
Domain Model

A domain model will act as a guiding model for discussions with many stakeholders and ideally a skeleton model should be created prior to the commencement of any workshops. The Domain Model should be kept simple and domain elements should be given a name and a description or a responsibility and initially only important connections should be made between elements. As the workshop

progresses new elements will be uncovered and can be added directly to the model giving the stakeholders confidence that their needs and concerns are being addressed and managed well. Enterprise Architect allows domain models to be created using the UML Class diagram.

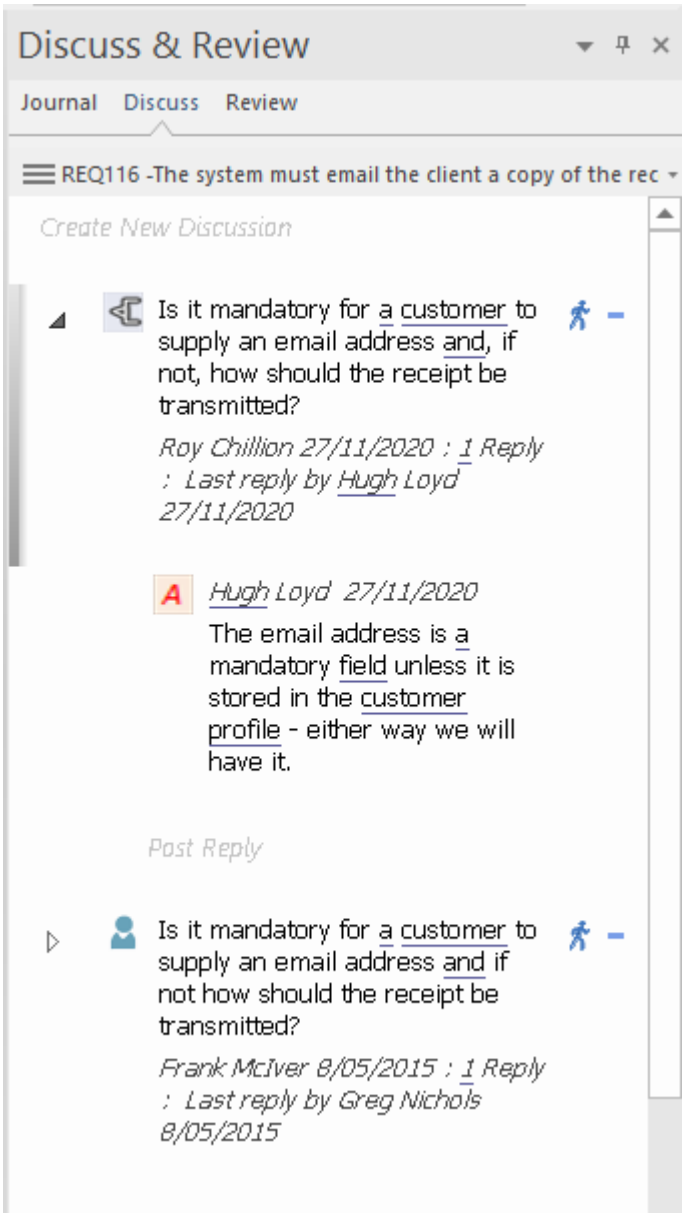
Domain Model

The Domain Model is a useful mechanism for recording and defining business terms that are identified during Requirements analysis. It provides a single definition of the terms and their relationships that can be referenced from anywhere within the model.
One of the advantages of using a Domain model is that the terms are modeled as Class elements, which can be linked to other elements within the Domain model itself or to elements in other parts of the model. They can be used on any number of diagrams, and they can be displayed as a list, using the Package List window.
It is also possible to create hyperlinks within the Notes text of Requirement elements that link directly to relevant Domain elements.



Discussions

The Discuss & Review window is a convenient facility that allows commentary to be made on elements without contaminating the notes with discussions that ultimately don't contribute to the integrity of the model. Modelers often place notes on diagrams or write questions in the element Notes fields, and these are distracting and must be removed when formal documentation is generated from the model. The Discuss & Review window allows a modeler to initiate a discussion and others to reply. It is a perfect way to discuss requirements.



The Discuss & Review window conveniently displays the Discussions for all elements in the repository.

Creating Requirements

Enterprise Architect has extensive support for developing Requirements, and provides a number of specialized tools for this purpose. As with all model content a modeler is encouraged to check whether the Requirements have been entered into the repository by someone else before embarking on the task of creating new Requirements. It is also possible that the Requirements have been defined in another tool such as a spreadsheet and could be imported into Enterprise Architect without the need to create each Requirement manually. Enterprise Architect has two locations for Requirements; they can be created in the model as an element that will appear in the Browser window, or they can be created inside another element as an Internal Requirement or Responsibility.

External and Internal Requirements

Enterprise Architect can support any type of Requirement process and allows Requirements to be defined as elements in the model. These are called External Requirements, but the tool also allows Requirements to be defined for a specific element, and these are called Internal Requirements. An Analyst who wants to define a user requirement such as the *The system must allow bus schedules to be updated*' would use an External Requirement. A modeler wanting to describe how a Component should behave would use an Internal Requirement for the Component, such as *The editor must support Unicode*. There is often contention between Analysts and Developers as to whether a Requirement should be internal or external, and Enterprise Architect provides a facility to move Internal Requirements to be external to the element. When they are moved they are still linked to the original element.

Creating External Requirements

Requirement analysts typically come from varied backgrounds and often have predilections about how they want to work so Enterprise Architect provides a wide range of ways of creating and managing requirements. External Requirements are Requirements that will appear in the Browser window and can be added to diagrams and viewed as separate elements with their own properties.

Methods for creating external requirements

Method	Description
Using the Specification Manager	Using the context menu and selecting 'Add New Element' will result in a new element being created in the grid ready for details to be added
Dragging a Toolbox item onto the current diagram	Dragging and dropping an item from a displayed toolbox page onto the current diagram will result in the element being added to the diagram.
Directly in the Browser window	In the Browser window, choose the location for the element to be inserted and select the 'New Element' toolbar option or 'Add Element' from the context menu.
Directly in a Package List	Display the context menu by right-clicking in the body of the window and select 'New Element'; the element will be added to the list.

Directly in a Diagram List	Display the context menu by right-clicking in the body of the widow and select 'New Element'; the element will be added to the list.
Importing from a variety of sources	Enterprise Architect supports a wide range of ways of importing requirements from external sources.
Moving an internal requirement external	Requirements that are defined inside an element can be moved external to a location specified by modeler thus creating a new element.

Notes

An Internal Requirement will not be displayed on a diagram by default; to ensure it is displayed you must set the compartment as visible either for the individual element or for all elements on the diagram.

Creating Internal Requirements

Internal Requirements can be created from an element's

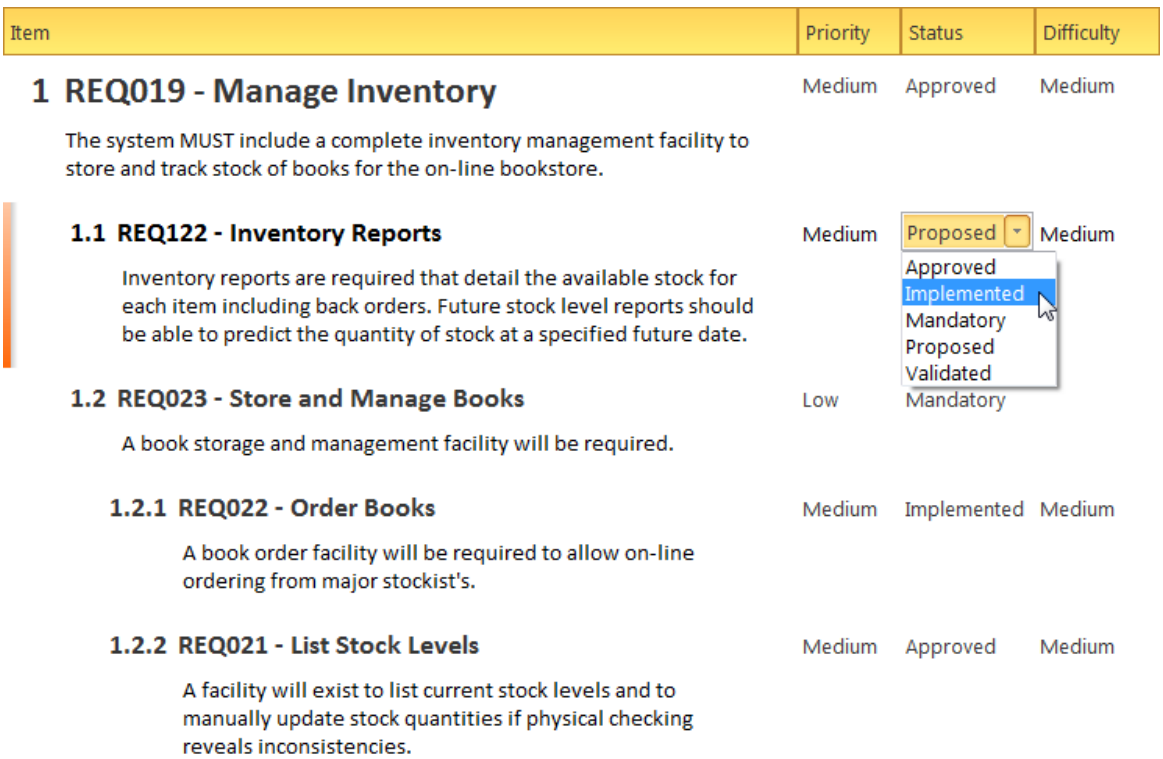
property sheet. This section describes how to do this.

Creating internal requirements

Step	Action
1	Select the element to house the Internal Requirement, and open the Responsibilties window for the element.
2	Select the Requirement section, the Requirement panel will be displayed.
3	Enter the name of the Requirement in the 'Requirement' field and any details into the notes field. Other properties such as Status and Priority can be added.
4	Select 'Save' to save the Requirement. The Requirement will be added to the element and will be displayed in the bottom section of the Window. Repeat the steps to enter another Requirement.

Using the Specification Manager

Enterprise Architect has a rich and fully featured tool called the Specification Manager for creating, visualizing and managing specifications, which is particularly useful for working with requirements. Requirements analysts come from varied backgrounds and often have strong predilections about the way they work and the tools they prefer to use, so Enterprise Architect has functionality to suit a wide range of work styles. Some analysts prefer to work with elements and can use the diagramming interface; others prefer to work with hierarchies, so they can work in the Browser window; but for those who prefer to work with text through an interface such as tables and spreadsheets, the Specification Manager would be their tool of choice.



When you create, delete or update elements in the

Specification Manager, they are automatically updated in the Browser window and any diagrams the element appears in.

Requirement Properties

Requirements development and management is critical to the success of any project and the properties of the requirements are important to the prioritization, and the way they will be elaborated and used within an implementation or development team. All Enterprise Architect elements have standard properties such as Status, Author and Phase but the requirement element has additional properties such as Difficulty and Priority. Some requirements processes will specify specific properties such as Owner and Volatility (Stability) and these can be configured by using Tagged Values that can be applied to each requirement. The 'Notes' field for a requirement has special significance as it often contains a formal and contractual description of how the system must behave or perform.

Access

On a diagram or in the Browser window, select an element and:

Ribbon	Design > Element > Editors > Properties
Context Menu	Right-click on element | Properties
Keyboard	

Shortcuts	Alt+Enter
Other	Double-click on element

Use to

- Document requirements
- Set requirement properties such as Type, Difficulty and Priority
- Set other element properties common to both Requirements and other model elements

Reference

Field	Action
Short Description	The name of the Requirement, which could include numbering, a natural language description, or some other formal specification. If you have set up Auto Names and Counters for Requirements and they are active, and you already have some text in this field, it would be over-written by the

	auto-counter text.
Alias	An alternative name (alias) to be used for this requirement. If you have set Alias autonaming and autonumbering, and you have already have some text in this field, it is over-written by the auto-counter text.
Status	The current status of this requirement.
Difficulty	An estimate of the difficulty in meeting this requirement; select from: • Low • Medium • High
Priority	The relative importance of meeting this requirement compared to other requirements; select from: • Low • Medium • High
Author	The modeler who created this requirement.
Key Words	A set of user-defined words that could be

	used to index or define the subject of this requirement.
Type	The type of this Requirement, typically used as a category for the Requirement. Possible values are defined on the 'Requirements' tab of the General Types window. This field displays a single value. You can click on the drop-down arrow and select a different value if necessary. However, be aware that you can define *multiple* values for this field, including stereotypes that you create or that are used in integrated or imported MDG technologies. You assign these multiple values using the Properties window for the selected Requirement element, in the 'Stereotype' field. This has two impacts on the 'Type' field: • The value displayed in the field might have been set on the Properties window, and might not be shown in the drop-down list (if it is a stereotype and not a General Type) • If you select a different value from the 'Type' drop-down list, you change only the first of the multiple values (the one displayed in the field); you do not

	change any of the other multiple values, which remain set
Phase	The project phase of this requirement.
Version	The version of this requirement.
Last Update	Read-only field specifying when this Requirement was last changed.
Created	Read-only field specifying when this Requirement was first created.
Notes	The description of this requirement, typically providing a more detailed explanation of the requirement. Some requirement processes prescribe that only a statement of the requirement be provided, and the 'Notes' field in these cases would remain blank. Novice modelers sometimes make the mistake of using this field for analyst discussions about the requirement; this commentary is best entered in the purpose-built Discuss & Review window.

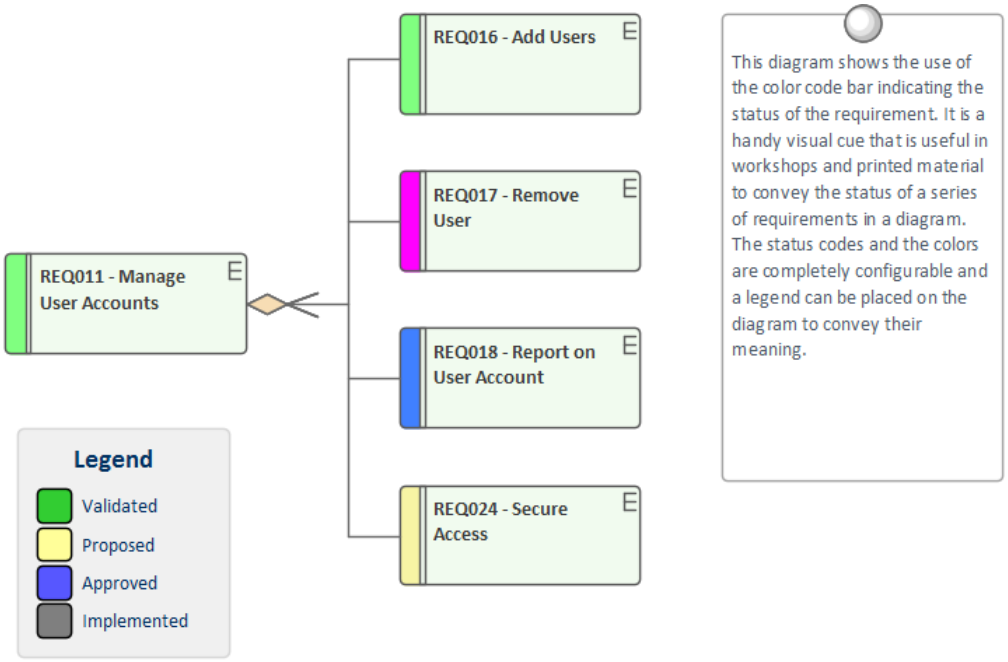
Notes

- In Requirement Management tools and texts, the characteristics of a requirement are commonly called 'Attributes'; however, in UML the term 'Attribute' refers to a different type of feature, and the Requirement characteristics are defined as properties - in this Enterprise Architect documentation, the term properties is used
- In a project, it might be necessary to define more information in a Requirement than is provided by the standard properties and Tagged Values can be used for this purpose

Color Coded Requirements Status

The status of a Requirement is an important property for project managers and other team members. Enterprise Architect provides a way to display the status of each Requirement as a color code in a diagram. The color codes act as a compelling visual cue, allowing team members and other stakeholders to get a quick view of the status of a set of Requirements. The color codes are pre-configured in Enterprise Architect but can be customized to suit any team, including deleting or adding additional codes and changing the color of existing ones. The default color codes are:

- Yellow for Proposed
- Blue for Approved
- Green for Validated
- Orange for Mandatory
- Black for Implemented



A modeler can choose whether to display the status color codes on diagrams in a repository, by changing this setting in the 'Objects' page of the 'Preferences' dialog.

Access

Ribbon	Start > Application > Preferences > Preferences > Objects > Show status colors on diagrams

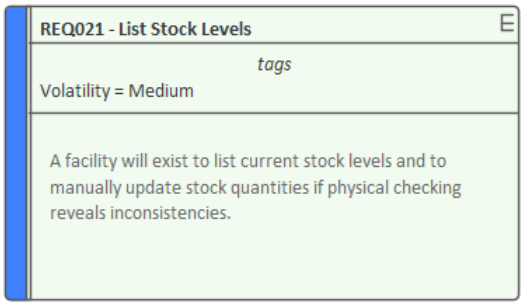
Enable color coded requirements

Step	Action
1	Select the 'Show status colors on diagrams' checkbox to enable the status of requirements to be represented by color coding in a diagram.

Displaying Properties on a Diagram

A diagram is a effective means of communication. For many stakeholders it will be the preferred method of viewing the contents of a repository and each stakeholder will typically want to see different information. Enterprise Architect provides great flexibility, allowing the modeler to tailor what is shown in a diagram. This includes whether to display the detailed notes of a requirement, the extended properties represented by Tagged Values, constraints, testing details and more. This feature is not only available for requirements but can be used with any diagram object in a repository. The customization can occur at two levels:

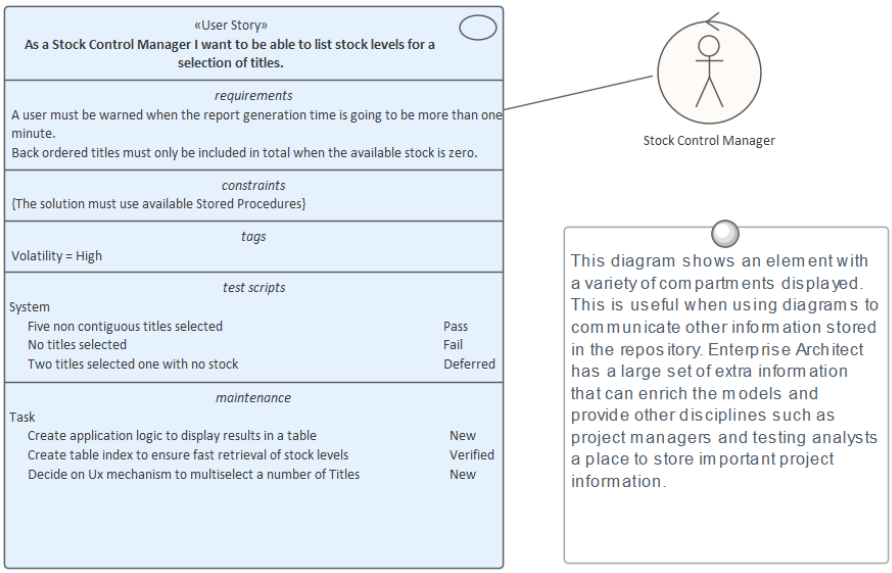
- Customize what is displayed for individual elements in a diagram
- Customize what is displayed for all elements in a diagram



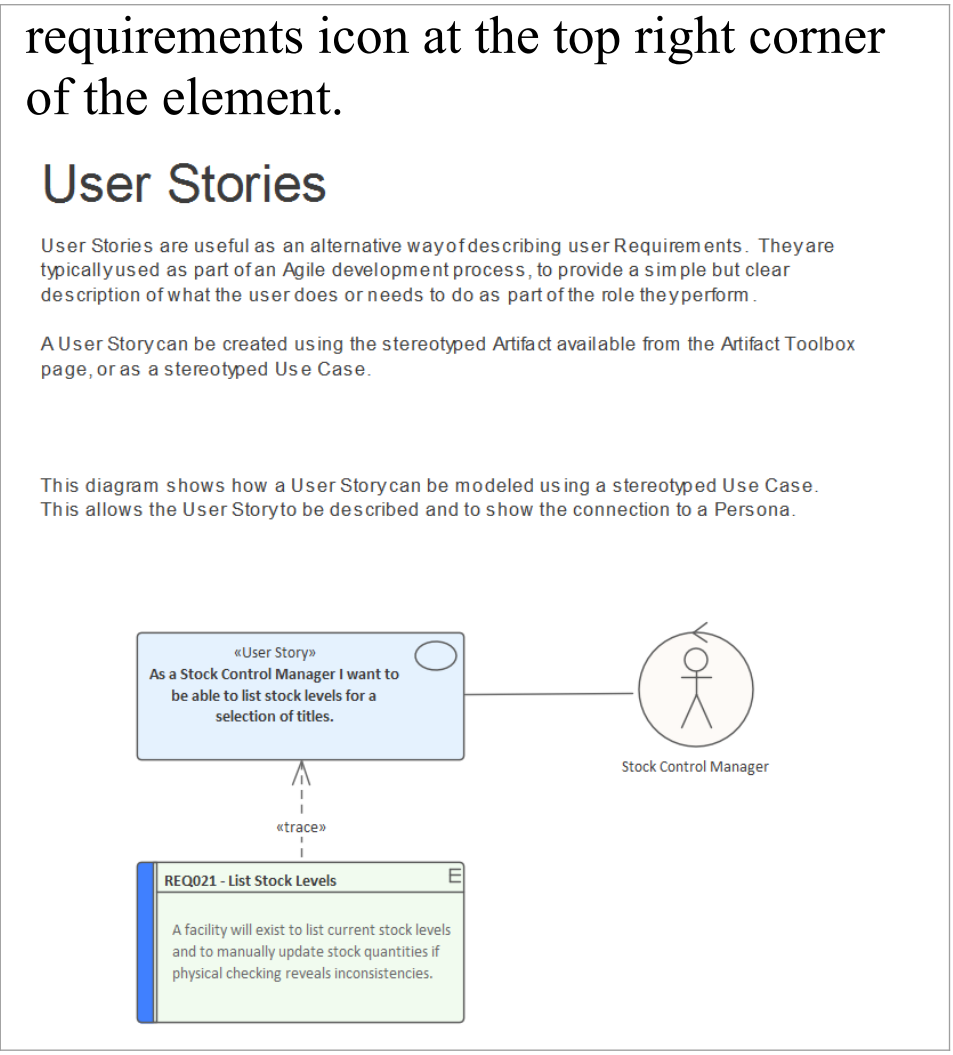
Customize what is displayed in a diagram

Action	Description
Customize	By default, requirements' extended

visible compartments	properties, notes and other element information are not displayed in a diagram, but it can be useful to display this in the diagram elements, particularly when conducting requirement workshops or creating documentation. Enterprise Architect supports element compartments, and any number of compartments can be displayed in diagram elements, including Notes, Tagged Values (extended properties), Constraints, Tests and Maintenance items such as Features, Changes, Documents, Issues, Defects and Tasks. There are two options to do this: • To display the additional compartment on all elements in a diagram, double-click on the diagram background and select the 'Elements' tab of the 'Properties' dialog for the diagram; select the compartment checkbox for each compartment to display and click on the OK button • To display the additional compartment on a specific element in a diagram, from the element context menu select the 'Compartment Visibility' option; select the compartment checkbox for each compartment to display in the

	'Show Element Compartments' panel of the 'Compartment Visibility' dialog, or for notes in the 'Element Notes' panel and click on the OK button The additional compartments are then displayed in the Requirement element(s) on the diagram. 
Displaying a stereotype in a diagram	By default requirement stereotypes are not displayed in a diagram. Enterprise Architect allows the modeler to display the element in a rectangular notation which displays the stereotype and an icon. They can be displayed for individual elements using the element context menu 'Appearance | Use Rectangle Notation' option. If the elements status colors were configured to be shown in diagrams the status color will be indicated in the

requirements icon at the top right corner of the element.



Notes

- When the display is customized at the level of the diagram these settings will take precedence over the element level settings
- When the display is customized at the level of the requirement the length of the notes text displayed can be further tailored to a particular number of characters and formatting options can be set

Import Requirements

A requirements analyst has a busy schedule of work, from running stakeholder workshops to sifting through project documentation and updating project managers with the status of the requirement development, so any opportunity to save time and work more efficiently is usually welcomed. One of the most difficult tasks that the analyst faces is how to get a central repository of requirements when the source documents are often in a variety of formats and tools. Enterprise Architect has a range of mechanisms to import requirements from disparate sources, including:

- Enterprise Architect models using copy and paste
- A text based document such as a word processor file
- A CSV file that can be exported from a Spread Sheet or similar tool
- An XMI file that has been exported from another model
- The Rational Doors requirement management tool
- Any file source using a script to process the file
- A reusable asset server that has a register of requirements

Import Requirements by Copy and Paste from another Model

This method of creating Requirements allows you to copy them from another Enterprise Architect model by simply

opening both models and copying the elements from one model to the other. You can copy a number of elements in the same Package or, if it is easier, you can copy an entire Package and all the elements it contains. When you copy elements they will be created as new elements in the target model and will be assigned new GUIDs.

Step	Action
1	Using the Browser window select an individual requirement or multi-select a number of requirements in the same Package and right-click to display the context menu.
2	Select: • For a single requirement, or a complete Package of requirements, the 'Copy/Paste | Copy to Clipboard | Full Structure for Duplication' option • For a selection of requirements, the 'Copy to Clipboard | Full Structure for Duplication' option Enterprise Architect copies the elements to the clipboard.
3	Select the target location in the Browser window and right-click to display the context menu.
4	Select 'Copy/Paste | Paste Element(s) from Clipboard' (or, if appropriate, 'Paste Package from

	Clipboard') from the context menu. Enterprise Architect creates the new elements in the target location, assigning new GUIDs to the elements.

Import Requirements from Text

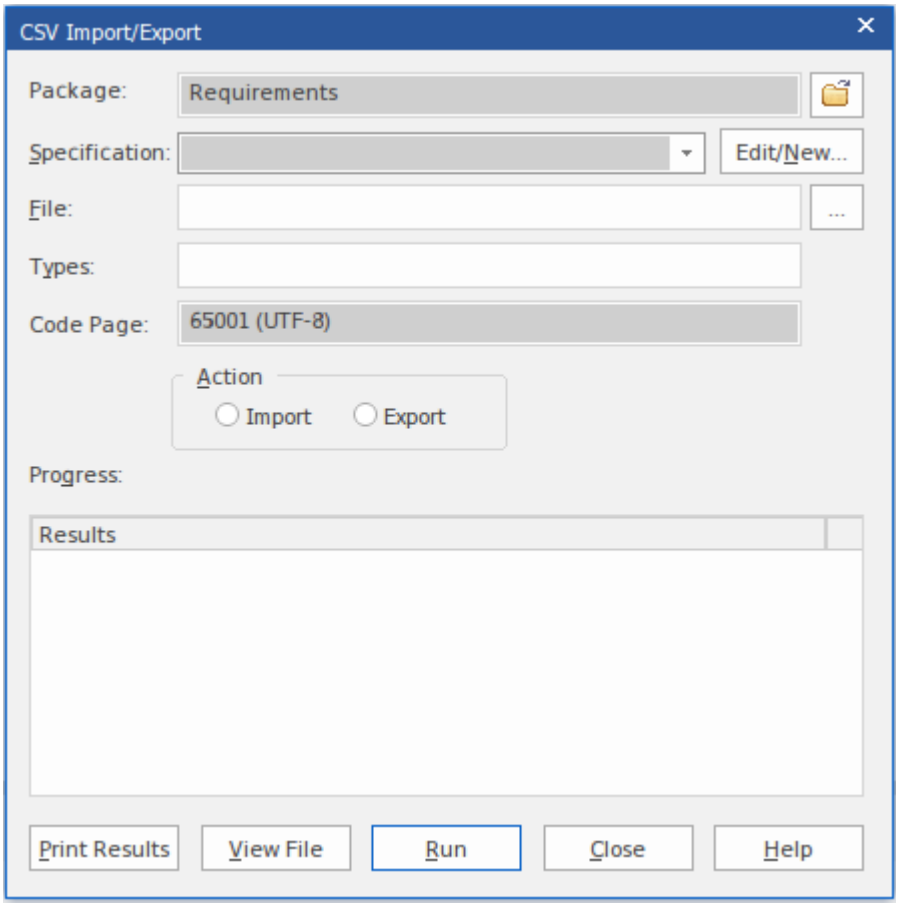
Word Processors and other text tools are commonly used to develop requirements when a team is not equipped with a purpose built requirements modeling platform such as Enterprise Architect. It is quite common to arrive at an elicitation workshop and find a customer or another stakeholder holding up a document saying that they have already started documenting their requirements. Fortunately Enterprise Architect has a convenient way of importing these requirements. This procedure creates a new element in Enterprise Architect by converting a text heading into an element's name and the text under the heading into the element's notes. You can use this method to generate any type of element; however, it is particularly useful for importing requirements from a requirements specification document. If there are a large number of requirements it might be more appropriate to use the Scripting method to import the requirements or, if they are in a table, to export them to a spreadsheet and use the CSV import mechanism.

Steps to Import Requirements from Text

Step	Action
1	Ensure you have a Requirements diagram open.
2	Open the document file containing the text you want to generate Requirement elements from (this can be opened in any common text editing tool).
3	Highlight the required heading and associated text and drag them from the text file into the diagram. The 'Toolbox Shortcut' menu displays.
4	Navigate through the menus and select the relevant element type, in this case Requirement. (If the diagram you are dragging onto is not a requirement diagram you will have to navigate to the requirement.)
5	Enterprise Architect creates a Requirement element in the diagram, and displays the 'Properties' dialog with the section heading in the 'Name' (or equivalent) field and the text in the 'Notes' field; the element is also added to the diagram's parent Package or element in the Browser window.

Import Requirements from a CSV file

Spreadsheets are often the default tool that analysts will use for Requirement Development, when they are not equipped with a more sophisticated tool such as Enterprise Architect. So it is quite common to find that a number of members of a team, including customers, have been entering requirements into a spreadsheet before Enterprise Architect has been installed. Fortunately, Enterprise Architect has a built-in facility to import such requirements so that they can be managed in a purpose-built requirements modeling platform, and the spreadsheets can be decommissioned. Columns in the spreadsheet will typically contain the name, description and additional properties of the requirements, which will need to be mapped to fields inside Enterprise Architect using a specification.



Access

Ribbon	Publish > Model Exchange > CSV > CSV Import/Export

Steps to Import Requirements from a CSV file

Step	Action
1	The Package defaults to the one selected in the Browser window into which to import the Requirements. If you want to change this Package, use the  icon to browse for the new Package.
2	In the 'Specification' field, click on the drop-down arrow and select the specification to use from the list. Alternatively, click on the Edit/New button to create a new one.
3	In the 'File' field, type in or browse for the source CSV file that contains the Requirements to be imported.
4	Click on the 'Import' radio button.
5	Click on the Run button to import the Requirements. The progress of the import is displayed in the 'Progress' panel.

Import Requirements from XMI

Enterprise Architect supports the XML Metadata

Interchange (XMI) exchange format which is governed by the Object Management Group (OMG) and provides a convenient way of exchanging models or model fragments. It is quite common for requirements that have been developed for one system to be applicable to another system; this is particularly true of non-functional requirements. Each project typically needs a unique set of requirements that have their own lifetime so these are best exported from one model and imported to another. XMI provides a suitable exchange format for this purpose and the XMI can be easily exported from one model and imported into another.

Access

Select your target Package in the Browser window, then:

Ribbon	Publish > Model Exchange > Import Package > Import Package from Native/XML File
Keyboard Shortcuts	Ctrl+Alt+I

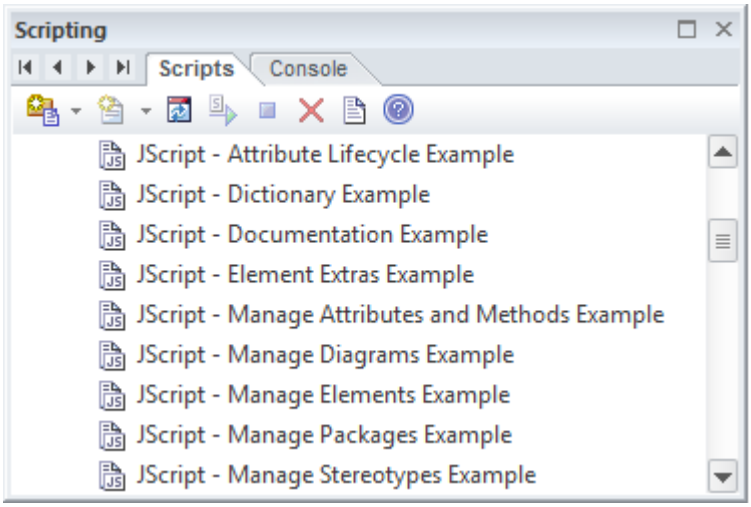
Import Requirements from XMI Steps

Step	Action
1	Select the location in the Browser window where you want the content inserted, and access the 'Import Package from XMI' dialog.
2	Select the filename of the XMI file to import, and select the appropriate options.
3	Click on the Import button to import the Package Enterprise Architect will prompt you to confirm the import; click on the Yes button to import the file. If there are unsaved diagrams you will be prompted to save diagrams before continuing. If any elements in the import file exist in the model you will be warned, and will not be able to import the file unless you select the 'Strip GUIDs' option.

Import Requirements from Any Format

Enterprise Architect can import requirements from a wide range of file formats, but on occasion the structure of the requirements in the source document, or the number of requirements, can make using one of these built-in

mechanisms difficult. Fortunately the requirements can be imported from any file format using the convenient and flexible scripting facility - available in the core product - or by developing an Add-In.



Import Requirements from Rational Doors

The Model Driven Generation (MDG) technology for Doors provides a lightweight bridge between Enterprise Architect and IBM Rational Software Architect (formerly Telelogic) DOORS. This allows the analyst to import the Requirements from Doors into an Enterprise Architect repository and to keep them synchronized with Doors. The entire hierarchy of Requirements will be imported and individual requirements can be linked to model elements such as Use Cases and Components. The Enterprise Architect model can be synchronized with Doors by re-importing the requirements. Any deleted items will be added to a 'Trash Can' Package but will not be deleted from the model or the diagrams.

Move Requirement External

Elements in Enterprise Architect can have internal requirements (responsibilities) that define what the element must accomplish. These can overlap or duplicate more formal requirements that the system in general must meet, so a modeler could decide to make an element's internal requirement into an external Requirement element. This is often done to allow a number of elements to implement the need expressed in the requirement. This can be achieved using the 'move external' function.

Access

On a diagram or in the Browser window, select an element and:

Ribbon	Design > Element > Responsibilities > Requirements
Context Menu	Right-click on element | Properties | Responsibilities > Requirements
Keyboard Shortcuts	Alt+Enter > Requirements Shift+Alt+R
Other	Double-click on element > Requirements

Change an element's internal requirement into an external requirement element

Step	Action
1	Right-click on the internal requirement to change to an external requirement. A short context menu displays.
2	Click on the 'Move External' menu option. The 'Find Package' dialog displays.
3	Locate and click on the Package to place the new Requirement element in.
4	Click on the OK button. A new Requirement element is created in the target Package, with a Realization connector from the current element to the Requirement. In the 'Properties' dialog, the Requirement is now marked with a 'Yes' in the 'External' column, and the dialog fields are disabled. To edit its details, double-click on the Requirement. The Requirement

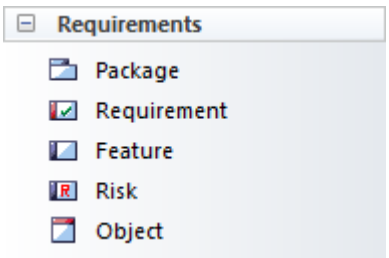
	element's own 'Properties' dialog displays.

Notes

- When an internal requirement is made into an external requirement element it is still viewable from within the element; any change to the external requirement will be reflected in the internal requirement and vice versa
- When an internal requirement is made into an external requirement element, the 'Stability' field and its value for the internal requirement are translated into the Stability Tagged Value in the external requirement

Recording Requirement Types

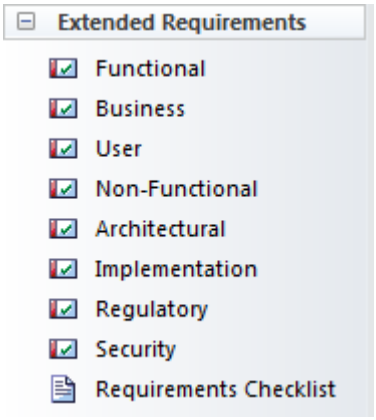
Enterprise Architect supports a wide range of element types and, in addition to the elements defined by the various standards, Enterprise Architect has added a series of extended elements that can be used for documenting a wide range of requirement types. When a Requirements diagram is created or opened, Enterprise Architect will display the Requirements Toolbox, which contains a number of standard and extended requirement elements. These elements can be dragged and dropped onto the diagram to create new elements, including a Package that acts as a container for Requirement elements and Feature, Risk and Object elements.



Extended Requirements

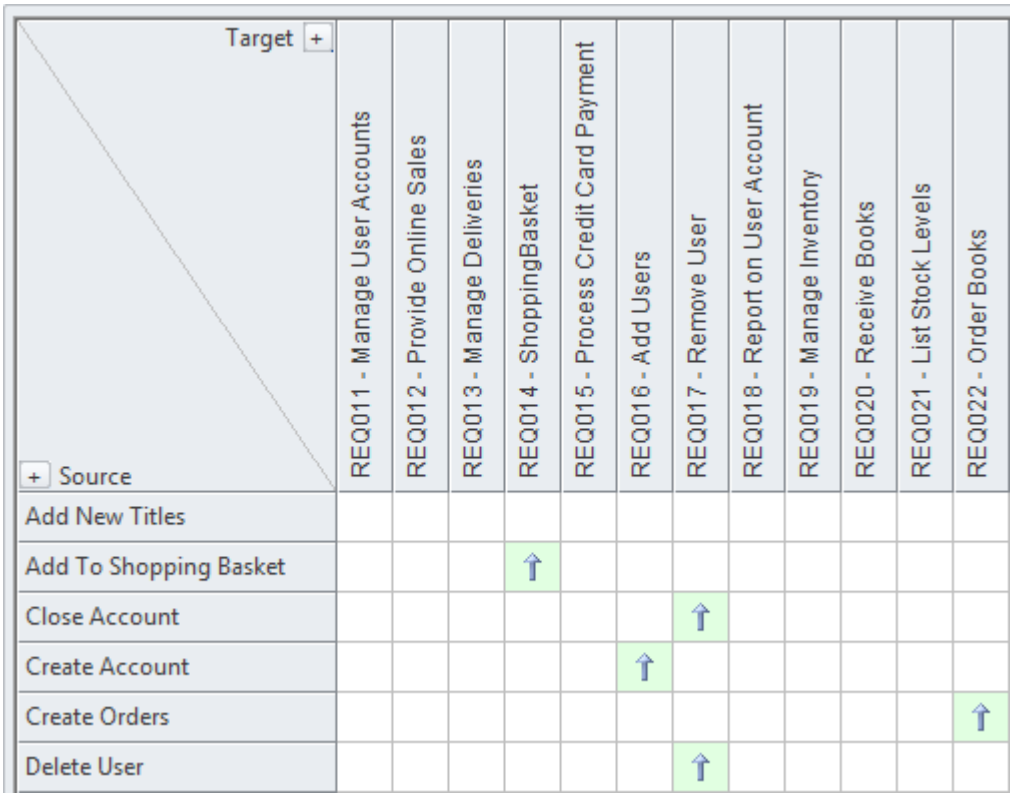
There is also a convenient set of extended requirements that can be used to model a variety of diagram types ranging from Functional, Business and User to Architectural and Non-Functional requirements. These come pre-configured with Tagged Values that allow you to specify additional

properties of the requirements.



Analysis

The analysis phase of Requirements development ensures that the Requirements discovered in the Elicitation phase have been articulated correctly and have the correct format, level of detail and properties and form a cohesive and correct set. As a result of the disparate sources and methods of elicitation the Requirements recorded in the elicitation phase will need some massaging and balancing - it is quite common for example to find repeated or overlapping requirements or for an analyst to have omitted to record the concerns of one or more stakeholders. Tools such as the Relationship Matrix and the Traceability window will reveal omissions and issues with requirements. The Discuss & Review window and the Chat & Mail window - incorporating the Model Mail facility - also provide mechanisms for discussing the Requirements with other team members.



Target / Source	REQ011 - Manage User Accounts	REQ012 - Provide Online Sales	REQ013 - Manage Deliveries	REQ014 - ShoppingBasket	REQ015 - Process Credit Card Payment	REQ016 - Add Users	REQ017 - Remove User	REQ018 - Report on User Account	REQ019 - Manage Inventory	REQ020 - Receive Books	REQ021 - List Stock Levels	REQ022 - Order Books
Add New Titles												
Add To Shopping Basket				↑								
Close Account							↑					
Create Account						↑						
Create Orders												↑
Delete User							↑					

Models Used to Document Requirements

One of the most important aspects of the requirements engineering discipline is to communicate knowledge and ensure that all stakeholders have a clear and unambiguous understanding of the problem and the proposed solution. This can be challenging because the stakeholders typically cross organizational boundaries and have a myriad of backgrounds spanning from senior business executives to low level engineers. This heterogeneous audience will need a variety of communication devices to ensure the various stakeholders are able to engage with the requirements and are able to understand them. Enterprise Architect is a modeling platform with a formidable range of tools and features that can be used to model requirements in almost any way. These include modeling stakeholders, requirements, user stories, user interfaces and a wide range of other models.

Requirements models

Model	Description
Textual Requirements	Textual requirements can be modeled using the Requirement element, and users can choose to work with the elements in a

	text tool such as the Specification Manager, directly in the Browser window in a hierarchy, or visually in a diagram. The Requirement element can be connected to other elements to describe a hierarchy of requirements, or to business goals or Use Cases and User Interface models. Through the Specification Manager Enterprise Architect allows the modeler to create, analyze and manage requirements in a text tool that resembles working in a spreadsheet but which is more effective by giving the analyst access to other models, including the glossary and the domain models.
Stakeholders	Stakeholders can be modeled using UML Classes and descriptions can be added that describe the stakeholder. Stakeholders are possibly the most important entities in the requirements engineering discipline and creating elements to represent them in the model allows them to be used as the owner of requirements and business rules. They can be placed onto diagrams allowing them to be visible in elicitation and prioritization workshops.
	A Glossary can be created and managed

Glossary	using the Project Glossary, ensuring that important project and domain terms are accessible right inside the model. These terms can be inserted into the Notes fields of elements including Use Case and User Story descriptions.
Use Cases	Use Cases can be modeled in a Use Case diagram and can be connected to a range of other elements including user interface models, User Requirements and Components. The Use Cases can be kept light-weight by just completing the description or they can be fully-dressed using the Scenario Builder Tool. Use Cases often present a problem for the requirements analyst because the diagrams are typically drawn in a diagramming tool and the text is written in a word processor, making it inaccessible to other model elements. Using Enterprise Architect's Scenario Builder the Use Case descriptions can be completed inside the Use Case itself inside the modeling tool. The tool can also produce behavior diagrams that represent the Use Case Scenarios automatically from the model.
	User Stories can be modeled using a

User Stories	stereotyped Use Case element and the text of the story can be completed in the description field. The Users and Personas can also be modeled and related to the story. Enterprise Architect allows the modeler to work with the stories in text form or in diagrams. Functional requirements can be added in preparation for handing to the development team for an iteration and these can be managed inside or outside the user story.
Domain Models	Domain models can be modeled using a UML Class diagram. The important entities in the business domain can be recorded, detailed and related to other elements. Creating domain models early in a project helps stakeholders make sense of all the important entities in a domain and the models can be used to generate a Data Dictionary. The domain elements can be created as links in textual statements of requirements, creating an articulated model that facilitates communication and understanding.
Process Models	Business processes are a useful way of recording the activities of a business including the events that trigger them to happen, the information that is produced

	or consumed, the outcomes and the roles that carry out the work. Enterprise Architect supports BPMN, UML and SysML Activity diagrams that can be used for this purpose.
Storyboards	Storyboards can be modeled using graphic elements in diagrams and a slide-show can be created to walk through the story.
Wireframes	Wireframes can be modeled using the Wireframing diagram, which has built-in support for popular hand held devices such as Apple and Android phones and tablets, and also for modeling dialog windows and web pages. Using Enterprise Architect's Wireframing tool, an analyst can create effective models of the arrangement of the application's content, describing interface elements and navigational mechanisms. Analysts and experienced designers have typically worked in isolation from other disciplines, but using Enterprise Architect the models can be created and maintained inside the same tool as the other requirement models, allowing traces to be created between other elements and the

	controls and content in the Wireframes.
User Profiles and Personas	User Profiles and Personas can be modeled using a stereotyped Actor element which allows descriptions and properties to be added that describe the persona.
System Events and Responses	A system will typically respond to a number of events and can also be responsible for creating events such as raising an alert or sending a data stream. These can be modeled in Enterprise Architect using BPMN or UML and SysML Activity diagrams.
System Interfaces	System Interfaces can be modeled using Provided and Required Interfaces and Ports on a Component diagram which describe how the software or hardware system interacts with other systems or how the internal Components of a system communicate. Enterprise Architect has rich support for modeling the interfaces and error codes and other behavior can be modeled. The interfaces can be linked to data definitions and Application Programmer Interface (API) specifications and a range of model

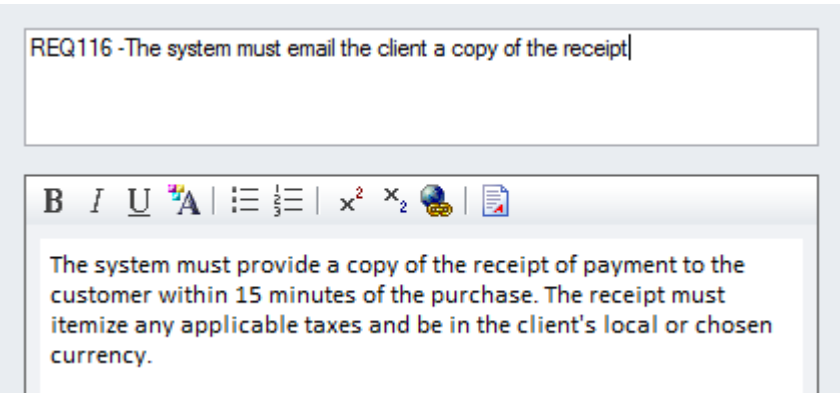
	elements including Use Cases and Requirements. The Interfaces can be added to documentation such as the System Requirements Specification and this document can be automatically generated from the model.

Requirements Naming and Numbering

Requirements are fundamental to the definition of a problem (or opportunity) and the solution must be traced back to this definition.

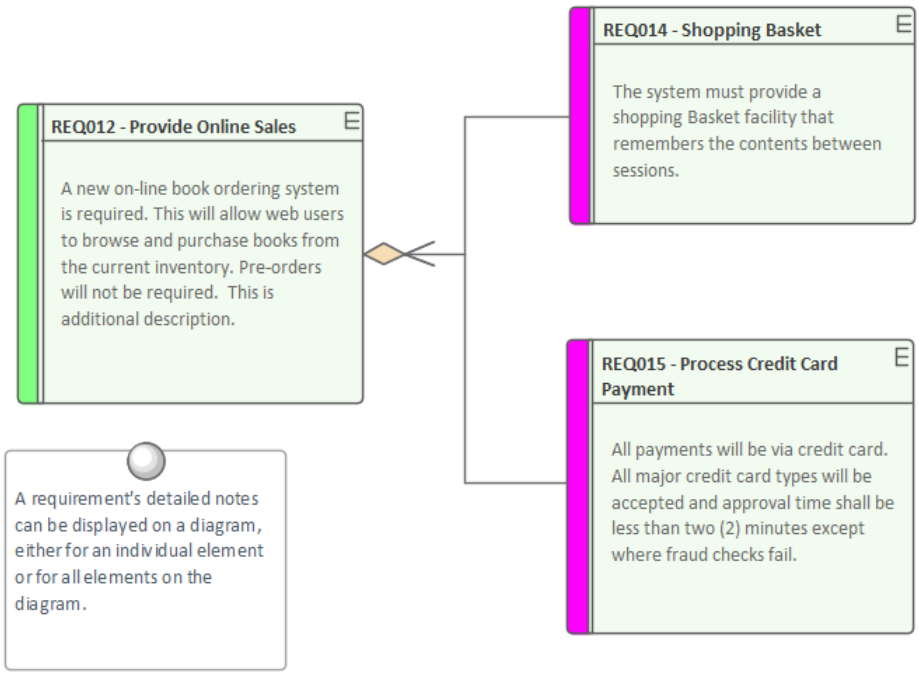
Requirement Names and Descriptions

There are many schemes that are used to name requirements and Enterprise Architect is flexible enough to support any scheme that is used. There are two locations you can add textual information to a Requirement: the element name, which has a limit of 255 characters, and the 'Notes' field, which is effectively unlimited. Some schemes specify that a single definition of the requirement is entered and no notes are needed. Other schemes prescribe a short name and specify that the requirement is clarified with detailed text. If notes are not used it is common practice to use some type of numbering system so the Requirement can be referred to unambiguously.



When Requirements appear in diagrams the name will by

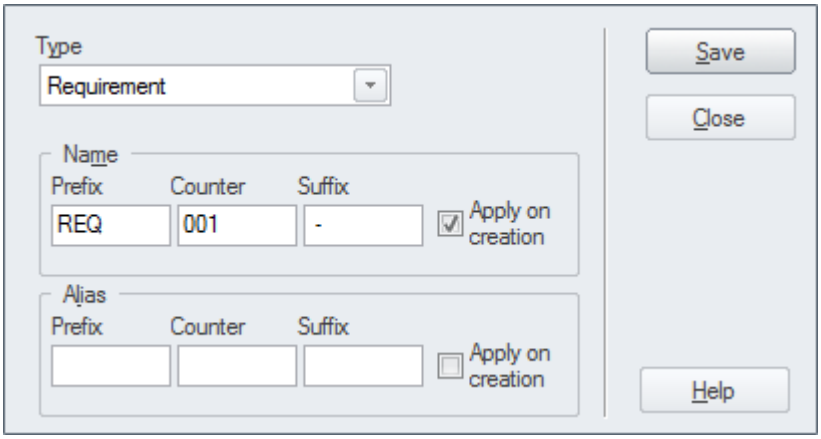
default be displayed but a modeler can choose to display any one of a number of the requirement's compartments including the notes. This technique creates expressive diagrams that reveal the details of the requirement and help the reader or reviewer to understand the Requirement more fully.



Sequential Numbering

Good practice often recommends that Requirements are given a sequential number when they are created so they can be referred to in stakeholder workshops, change requests, conversations with System Integrators or implementation teams. Using a name in this situation is often unwieldy and subject to error so a sequential number is preferred. Enterprise Architect has a facility called Auto Names and Counters for this purpose that can be used to assign a

sequential number to any element type including Requirements. It includes a prefix definition, a counter and a suffix definition allowing numbers such as: 'REQ007 - Manage Inventory' to be created. These can be further refined to numbering systems such as this architectural requirement: 'ARR134 - Payloads for internal component interfaces must use an XML format'.

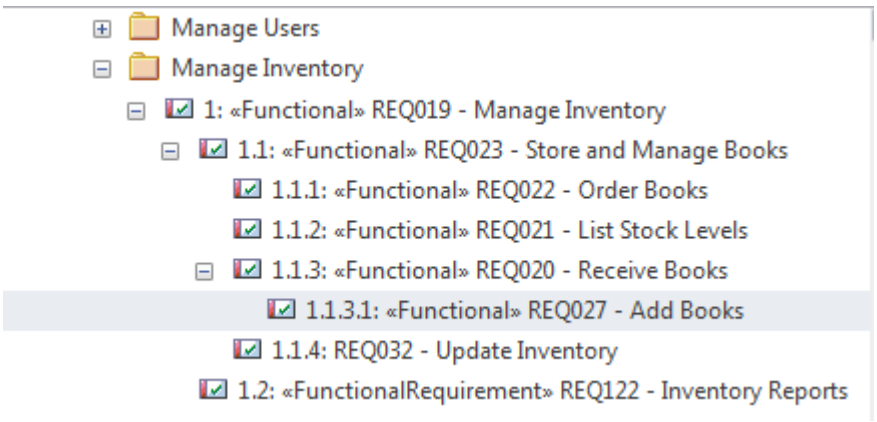


The counter is added to the name and is displayed in all views of the repository including the Browser window, Relationship Matrix, search results and diagrams.

Numbered Hierarchies

When Requirements are written in word processors they typically use a numbering scheme called Outline Numbering, which assigns a number to the first level heading such as: '4 Inventory Requirements' and then a sub-heading is numbered by adding a period and a number such as '4.1 Stock Levels' and again down another level '4.1.1 List Stock Levels'. Enterprise Architect has a facility called Level Numbering that applies hierarchical numbering

to the elements in a Package. This is a useful mechanism that is displayed in a number of locations, including the Browser window, the Specification Manager, Diagram List and Package List. It must be remembered, however, that if the order or the level of the elements in the Package is altered they will be assigned new numbers based on their new position; this makes this mechanism unsuitable if immutable numbers are needed.

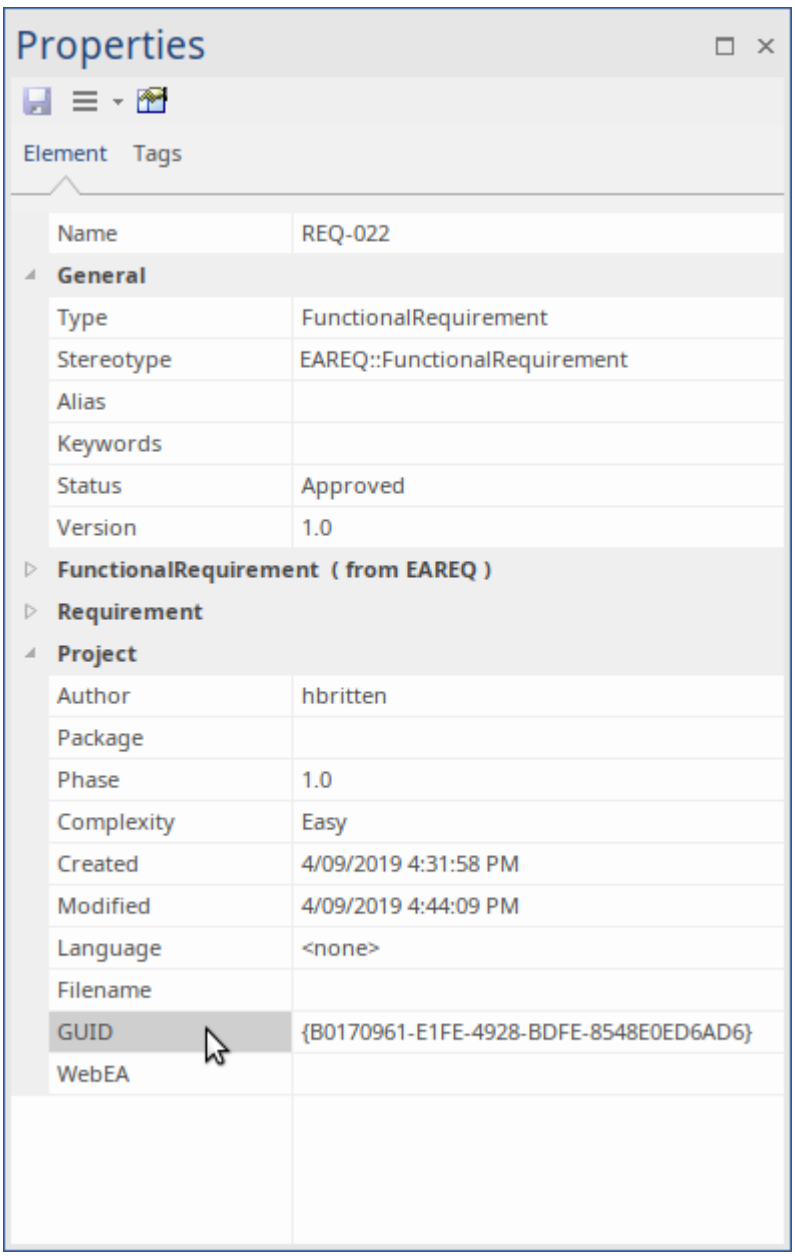


Numbered Packages

This is a hybrid method where Packages are used to create a high level naming and numbering structure and the Requirements in each Package are numbered using the Package identifier and a number to identify them. So Requirements for the Fulfillment of Orders could be contained in a Package named '2.4 Fulfill Orders' and an individual requirement in this Package could be named 'FO-01 Process Credit Card Payments'. This would be manually maintained or a Script could be written to ensure that the numbers were correctly assigned.

Globally Unique Identifier

Every element, diagram and connector in an Enterprise Architect repository is given an immutable and unique reference in the form of a Globally Unique Identifier (GUID). The GUID is assigned to the element when it is created and is guaranteed to be unique across time and space. Thus requirements can ultimately be referred to by this unique identifier. While the GUID is a useful and irrefutable way of referring to a Requirement it is not practical to use it in discussion with stakeholders because of its length and the fact that it is not memorable. The GUID's purpose is to be able to track and manage a Requirements provenance particularly when Enterprise Architect is used to generate Requirements to other tools. It is also used as the identifier in the XMI exchange format.



Proprietary Numbering Systems

There might be projects or programs of work that will, for regulatory or commercial reasons, specify a proprietary numbering system that must be used with Requirements. For this reason one of Enterprise Architects in-built schemes might not suffice; in this situation the user can create their

own numbering scheme using the Scripting facility in combination with Tagged Values.

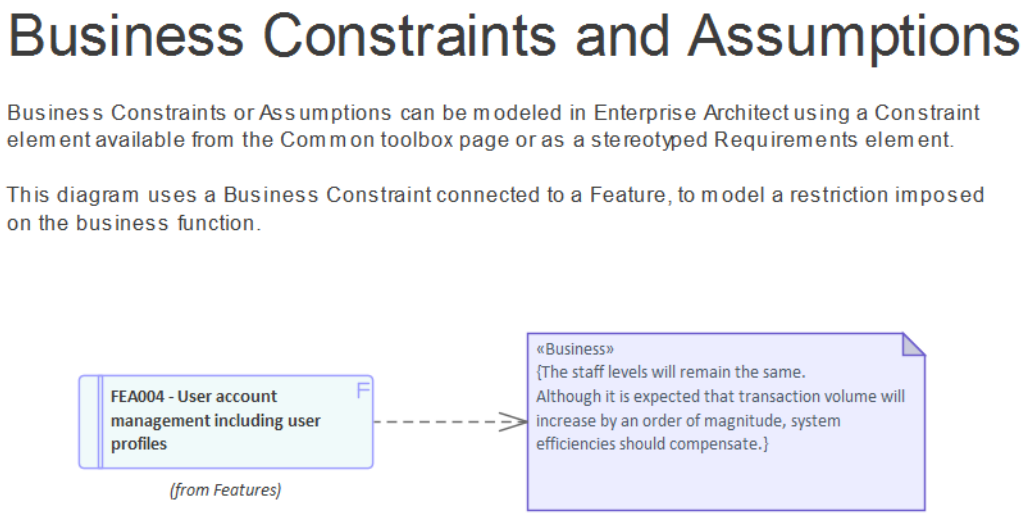
Model Assumptions and Constraints

When an analyst is working through the information acquired from the elicitation process they will typically come across statements or conditions that are best described as Assumptions that have been made or Constraints that will restrict the solution is some way. These are not Requirements but have an important role in the requirements development process because they have the ability to affect the solution and must be understood.

Business Constraints

A Business Constraint is a business restriction or limitation imposed on the choices that can be made for the design, implementation or deployment of the solution. They are typically restrictions of budget, time and resources, but can be any type of limitation such as the context of the business deployment where the solution must not change the way that branch staff currently work. A Business Constraint might also limit the access or presentation of information such as 'Only the last four digits of a credit card number can be displayed in reports.' There is some overlap with business rules and the analyst should be careful to separate the two notions. Business Constraints can be modeled in Enterprise Architect using a Constraint element available from the Common toolbox page or a stereotyped Requirements element. They can be related to one or more model elements

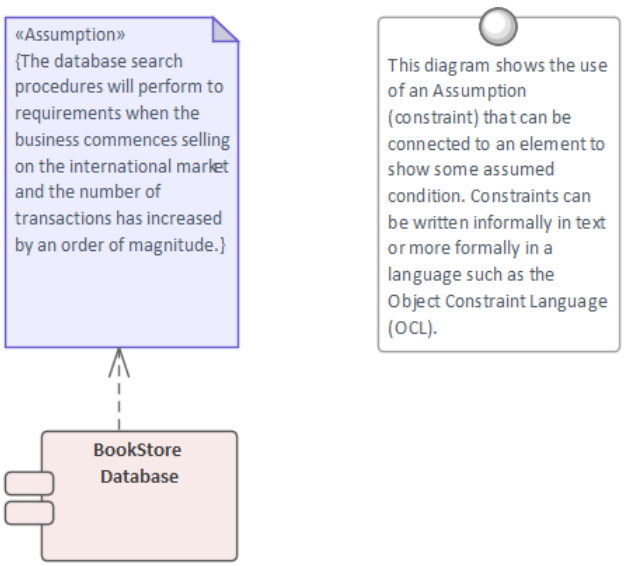
using a Dependency relationship. Constraints can also be created as a property of an element using the Properties window.



Assumptions

An assumption is a statement that is believed to be true but that has not yet been verified. It is imperative that assumptions are modeled and attempts are made to verify them as they have the potential, if proved to be false, to significantly change in the definition of the problem and therefore the solution. They can be statements made about the way things are currently done or they could pertain to a future process or solution. Assumptions are similar to Risks, and good practice would prescribe them as being managed in a similar way to Risks. Attempts should be made to verify them as early as possible in the requirements development phase. An example of an assumption is: 'The User will know the meaning of toolbox icons as used in other Windows

applications'. Based on this assumption the solution designer might plan not to implement tool-tips for the icons. Assumptions can be modeled in Enterprise Architect using a Constraint element, available from the 'Common' Toolbox page, or as a stereotyped Requirements element. They can be related to one or more model elements using a Dependency relationship.



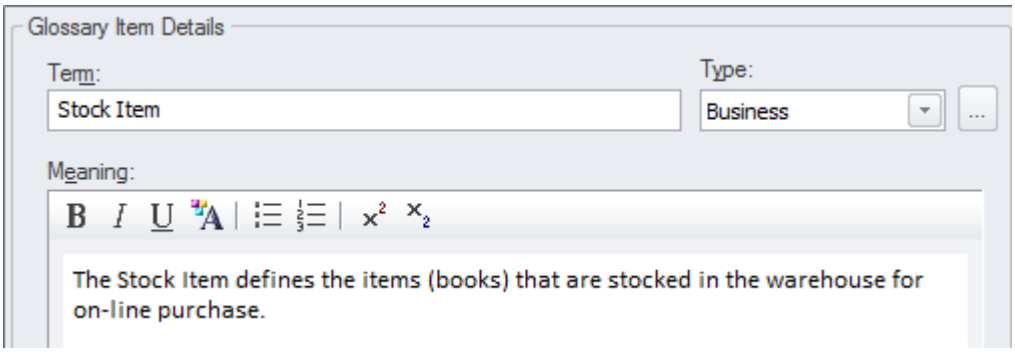
Technical Constraints

A technical constraint is any restriction on the choices that can be made for the architecture, design, implementation or deployment of the solution. They can start with principles defined in the enterprise architecture that restrict the types of platform, programming language and decision to buy or build part of the solution. They could also be restrictions on the type of protocol or standard that the solution must implement or comply with. Restrictions on file sizes and

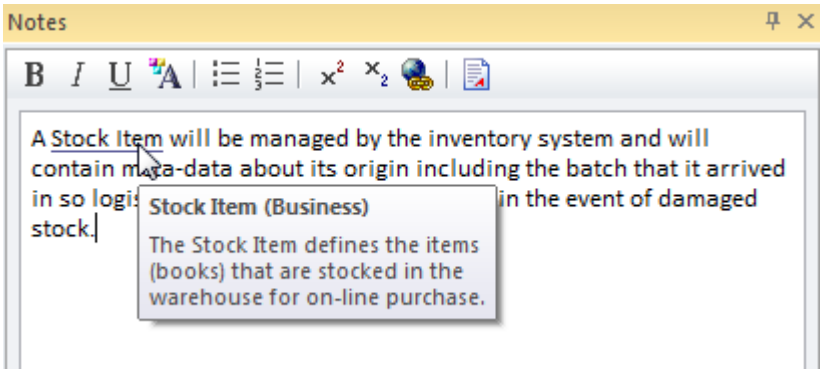
formats can also place limitations on the solution choices. There is some overlap with non-functional requirements and the analyst should be careful to separate the two notions. Technical constraints can be modeled in Enterprise Architect using a Constraint element available from the 'Common' Toolbox page or as a stereotyped Requirements element. They can be related to one or more model elements using a Dependency relationship. Constraints can also be created as a property of an element using the Properties window.

Create a Glossary

One of the fundamental precepts of requirements engineering is to promote understanding and convergence of thought and to remove ambiguity. It is quite common for members on a project team to have quite disparate understanding of domain concepts. This can be easily rectified by creating a Project Glossary early in the project and ensuring the glossary is accessible to the project team. Enterprise Architect has a built in Glossary that allows the requirements analyst to define terms and meanings, and to create categories (types) of terms.



One of the features that makes the glossary useful is the ability to link from text fields such as element notes to terms in the Glossary, and a rollover in the text displays the meaning.



Notes

Experienced modelers tend to define the terms of a domain in a Domain Model and reserve the glossary for project and process terms and their definitions such as Use Case, Software Requirements Specification, Metric etc. This allows relationships between domain terms to be defined and data elements (Attributes) can be added including datatypes creating a more expressive representation than would be possible in the Glossary.

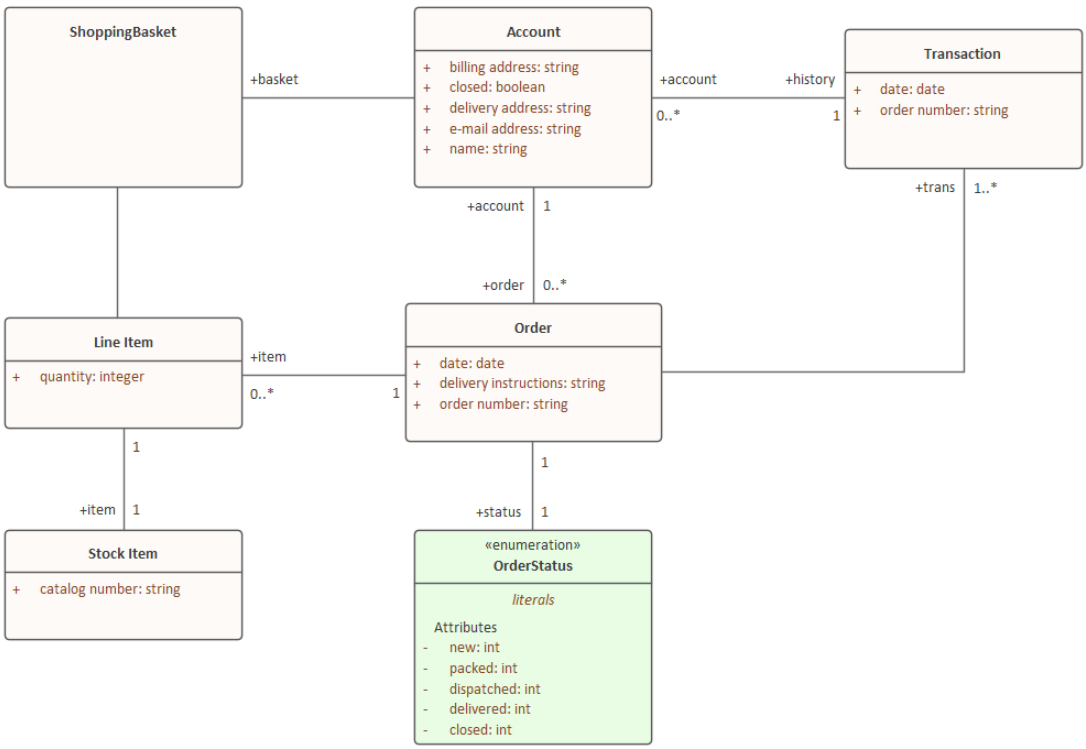
Create a Domain Model

Requirements analysis will reveal a number of business terms that must be defined if the requirements are to be understood and clarified. There are a number of options for recording these terms, including the Project Glossary, which is a purpose built lexicon through which you can list, define and categorize terms. The Domain Model (business object model) is another useful mechanism for describing the important terms of the business, providing a single definition of the terms and their relationships that is accessible to all project staff, from high level business managers to low level engineers. One of the advantages of using a Domain Model is that the terms are modeled as elements, allowing them to be linked to other elements within the Domain Model itself or to elements in other parts of the models.

Enterprise Architect has extensive support for modeling a domain using a UML Class diagram, or the Classes can be created directly in the Browser window and displayed in a list using the Package List window.

Domain Model

The Domain Model is a useful mechanism for recording and defining business terms that are identified during Requirements analysis. It provides a single definition of the terms and their relationships that can be referenced from anywhere within the model.
One of the advantages of using a Domain model is that the terms are modeled as Class elements, which can be linked to other elements within the Domain model itself or to elements in other parts of the model. They can be used on any number of diagrams, and they can be displayed as a list, using the Package List window.
It is also possible to create hyperlinks within the Notes text of Requirement elements that link directly to relevant Domain elements.



The Domain Model elements can be added to other diagrams to make them more expressive and to show important information, such as what data is being consumed or produced by a business process, or what data is being passed between application interfaces.

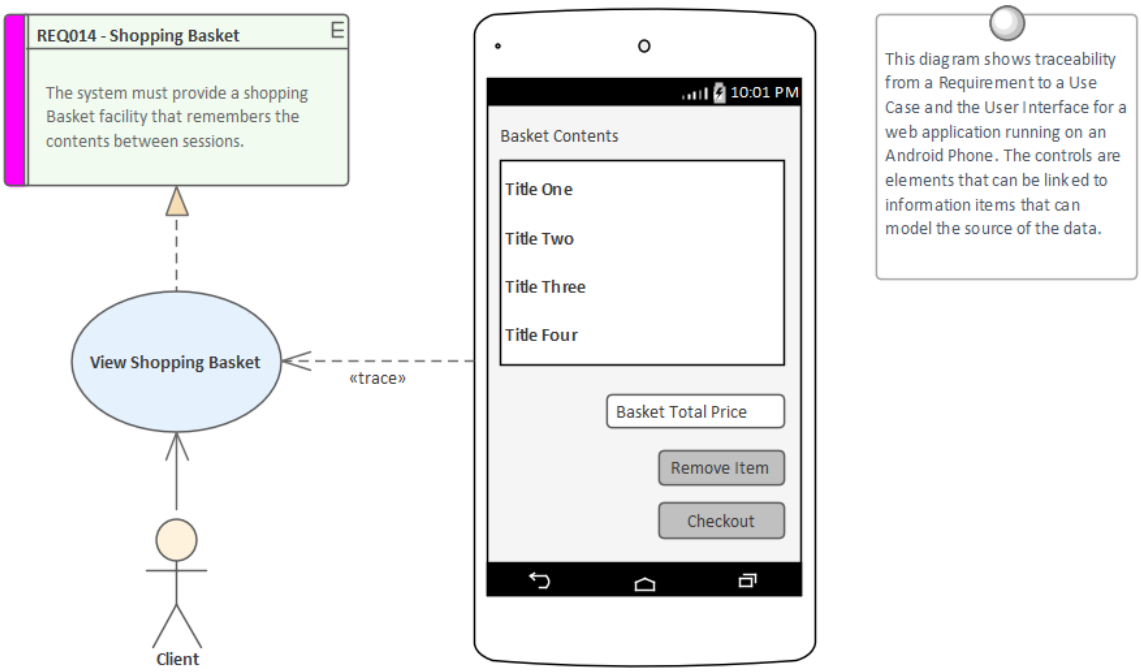
Enterprise Architect allows you to link to the domain elements (or any element) from the text of a requirement's notes. (This facility is available from any element.)

Notes

Experienced modelers tend to define the terms of a domain in a Domain Model and reserve the Glossary for project and process terms and their definitions such as Use Case, Software Requirements Specification, Metric etc.

Model the User Interface

The User Interface for a system can be modeled in a variety of ways depending on whether the interface is for the Web or for Win32 applications or for Web Applications targeting mobile devices.



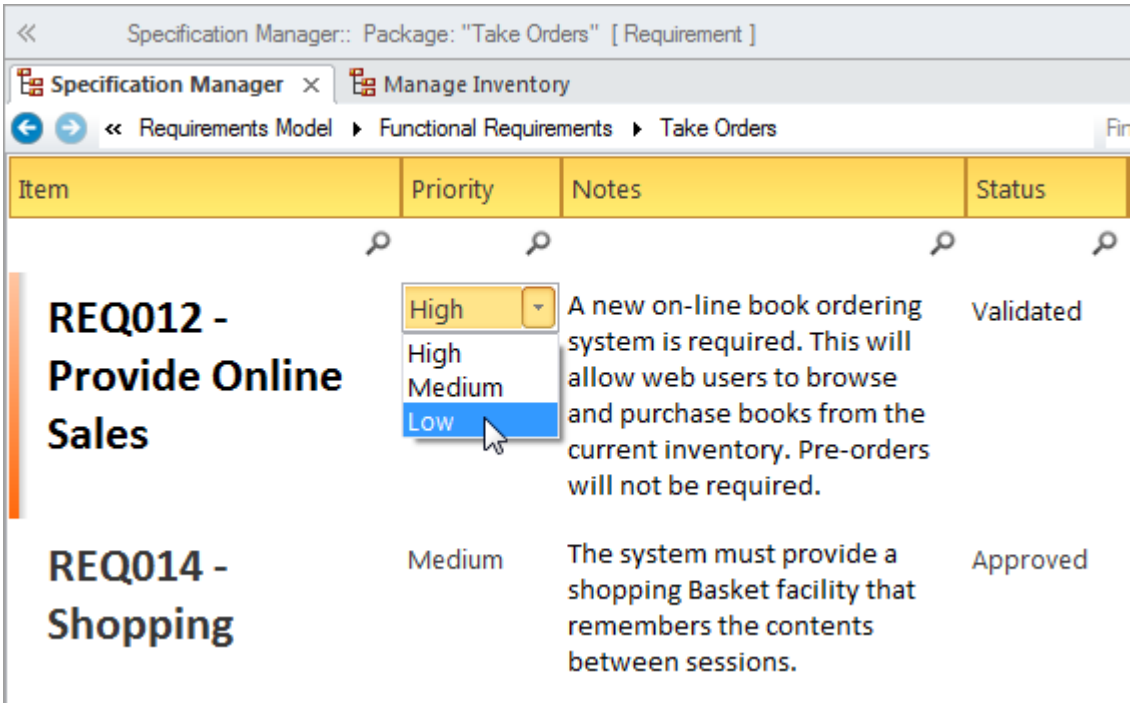
Prioritize the Requirements

Prioritization of requirements is imperative to the success of a project as it ensures that analysis, development, testing and implementation resources are focused on the most critical aspects of the system. Prioritization is a decision process that allocates a priority to the each requirement, the most common criteria for the categorization is business value. Business value is typically determined by the cost-benefit analysis of the value the implemented requirement will produce for the organization or its customers. Other factors might be policy or regulatory compliance, urgency, business or technical risk and the likelihood of success.

Changing the Priority Collaboratively

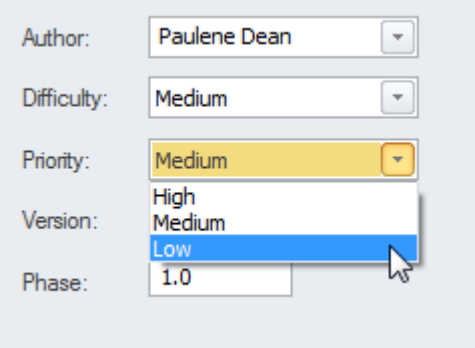
The process of selecting criteria and assigning priority is typically collaborative, and is often done in a workshop with stakeholders or their representatives debating the categorization. In previous eras this was a laborious and difficult process, but Enterprise Architect has features for working with requirement properties, including priority. There are a number of windows that support working with the requirements in a list and editing the priority in-line, automatically filtering or sorting the list of requirements based on the newly assigned priority, including the Package List and the Diagram List. The Specification Manager is a

useful tool for this purpose and provides a text based interface where the requirements and their notes can be viewed and priorities can be selected from a drop down list. The interface also displays a number of other useful properties that are typically useful for prioritization, such as Status and Complexity.



Requirement Priority Property

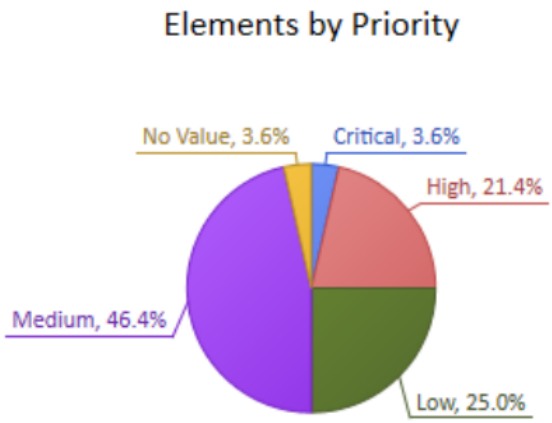
A wide range of criteria can be used for prioritization, and each organization and project will typically use some type of weighted average to determine the priority. Enterprise Architect has flexible and complete support for requirement prioritization, as each element has a built in 'Priority' property that can be set to indicate its priority, by selecting the appropriate value from a drop down list.



The list of priorities is conveniently pre-loaded when you install Enterprise Architect, but these values can be edited or completely revised to suit your organization or project. They can even be imported as reference data from a previous project or, if the current project was created based on a template, the organization's priorities could be pre-loaded from the base model.

Dashboard Diagams

Enterprise Architect has a series of dashboard diagrams that can be used to create a compelling view of the Priority of requirements in a Package with the option to include sub-Packages. There are a number of pre-configured charts that can be used to display the ratio of Priority values for Requirements in a part of the model. Filters add another level of user configuration allowing a modeler to, for example, exclude requirements of a particular Status or ensure only requirements for the current phase are displayed.



This diagram shows a Pie Chart element depicting element priorities for all the Requirements in a selected Package. It provides a useful summary for a Requirements Manager and is dynamically updated when the priority changes and the diagram is reopened. A range of other pre-defined Charts and user-defined Charts can also be added. A filter has been added to exclude all elements other than Requirements.

Extension for Setting Priority

When there is a large number of Requirements and the basis for the prioritization has been well defined, a script or Add-In could be developed that could assign the Priority automatically based on a specified algorithm. Additional properties such as business value could be assigned using Tagged Values, and any Requirements that did not have the requisite properties set could be written to a log or presented as a search result list. The script or Add-In could be used on multiple project,s giving a consistent and unbiased result.

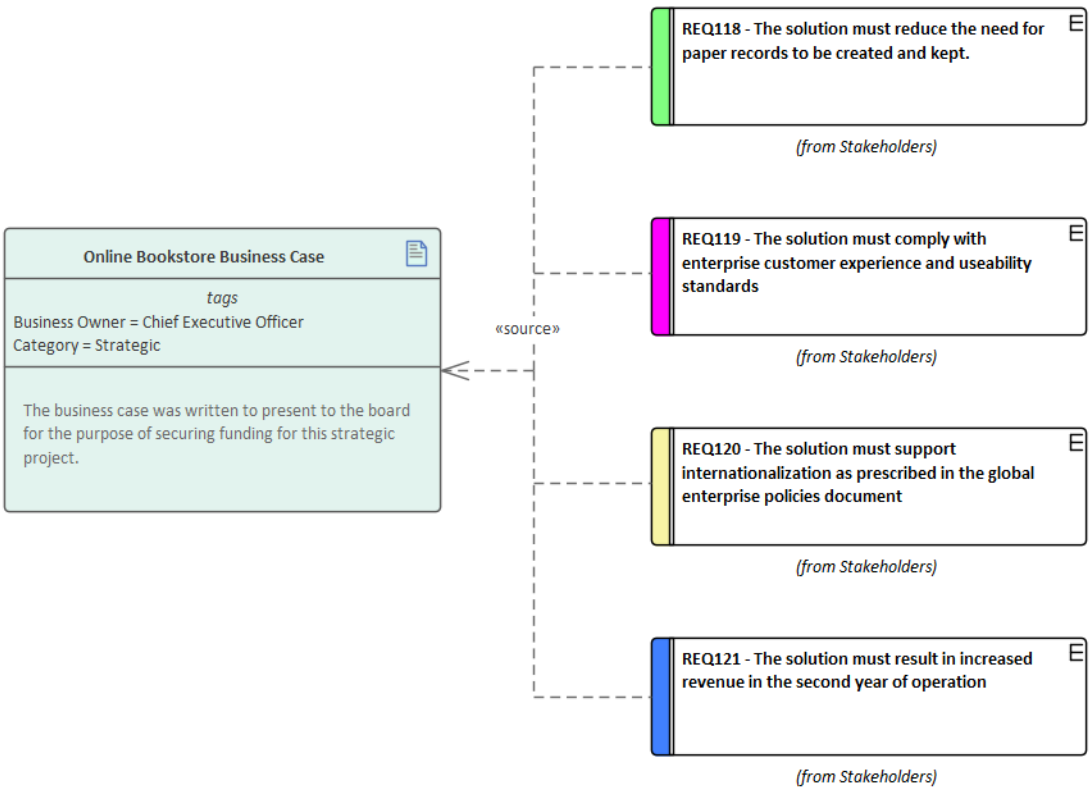
Specification

Requirements are typically derived from a wide range of sources, and there are often a number of requirements analysts working on a project. This will tend to make the Requirements disparate. Ensuring the Requirements are consistent and of high quality is critical to the success of a project.

Requirements Sources

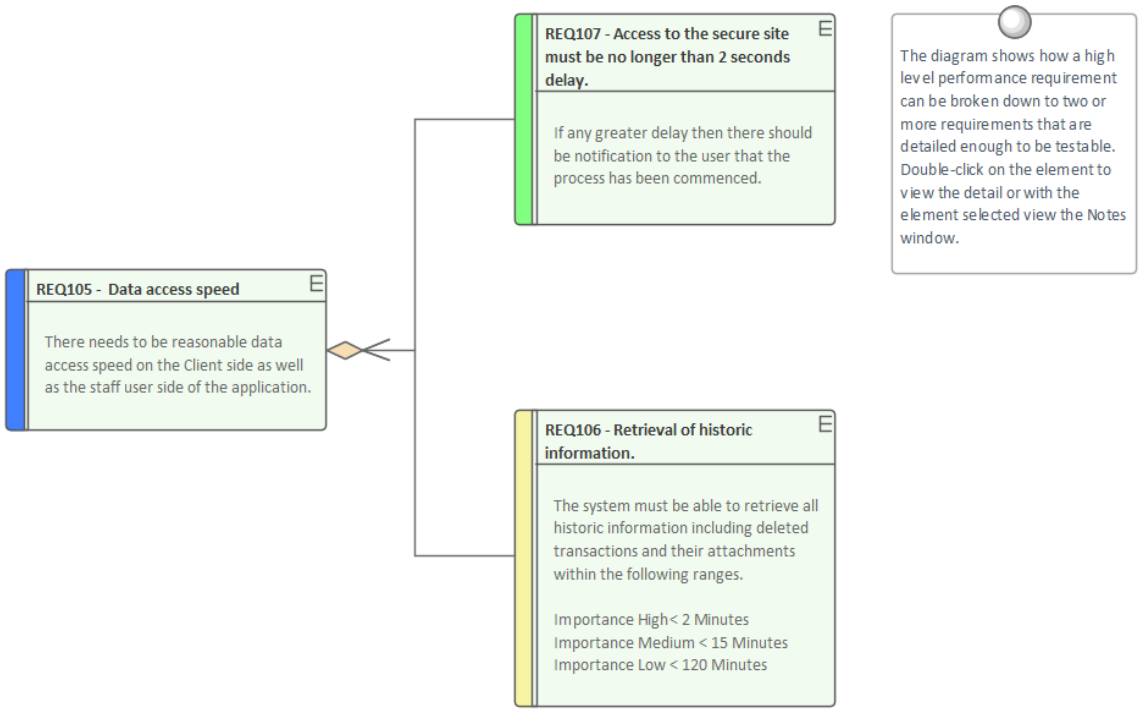
This diagram shows an external document, "Online Bookstore Business Case", modeled as an artifact. Requirements have been linked back to this artifact, to indicate that the source of the requirement is this document. If the document is subsequently updated, the requirements derived from it are easily located. The Business Case document artifact has a number of Tagged Values indicating properties of the document.

Hyperlinks to external documents can be created by simply dragging and dropping a document file onto a diagram canvas.



Specify Quality Attributes

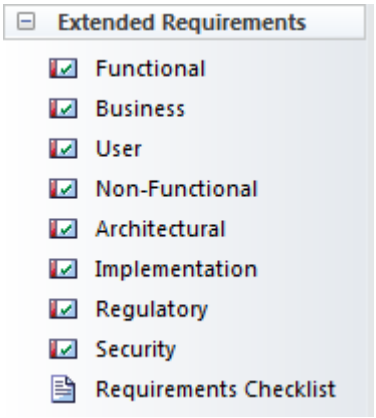
Non Functional Requirements and Quality Attributes are important inputs into the Architecture of a system and provide the criteria that define what the stakeholder's expectations are about how well the system will operate. Regardless of how well the Functional Requirements have been defined and implemented, if the Quality Attributes aren't built into the architecture and implemented the system is unlikely to be satisfactory to its stakeholders and users.



Adding Non Functional Requirements

Enterprise Architect conveniently has a number of Requirements such as the Non Functional, Architecture, Implementation and Security Requirement types available

from the 'Extended Requirements' page of the Requirements Toolbox. These can be dragged onto a diagram or created directly in the Browser window or Specification Manager.

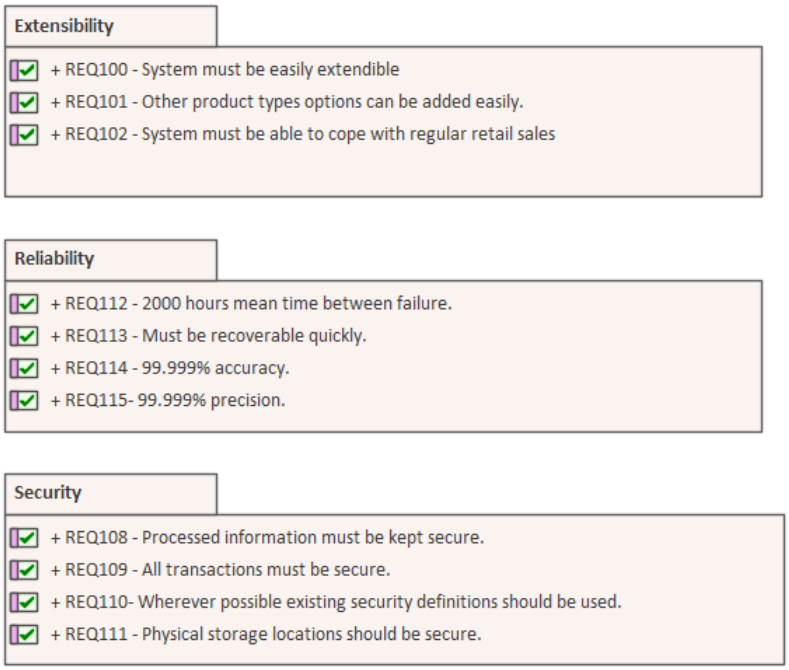


Requirements Packages

Quality Attributes are often grouped into sets, such as those pertaining to security or performance, or one of the '-ilities' such as Reliability. A requirements Package can be used to group these elements together which provides a convenient categorization for reporting. The Packages can be seen in the Browser window and can also be added to a diagram with the option to display the individual Requirements inside the Package.

Non-Functional Requirements

The Non Functional Requirements have been defined using a number of separate packages. The packages can then be displayed on a diagram, showing the Requirements they contain.



Requirement Sources

Much of the literature on requirements development talks of 'gathering' requirements, which implies that the analyst is walking through an orchard picking low hanging fruit. In practice the process is quite the opposite and the requirement analyst needs to have all the skills and tenacity of a detective to discover the sources of requirements. The requirement sources and the documents and artifacts that are discovered are first class citizens and can be modeled in Enterprise Architect to provide a register of the requirement sources. This is a list of common requirements sources:

- Interviews with users and other stakeholders
- Observations of users performing tasks
- Business Case or Proposal
- Concept of Operation or Vision document
- Procedure manuals and user task lists
- Enhancement Requests for the existing system
- Marketing material and product definitions
- Analysis of a market leader or competitor's products

Modeling Requirement Sources

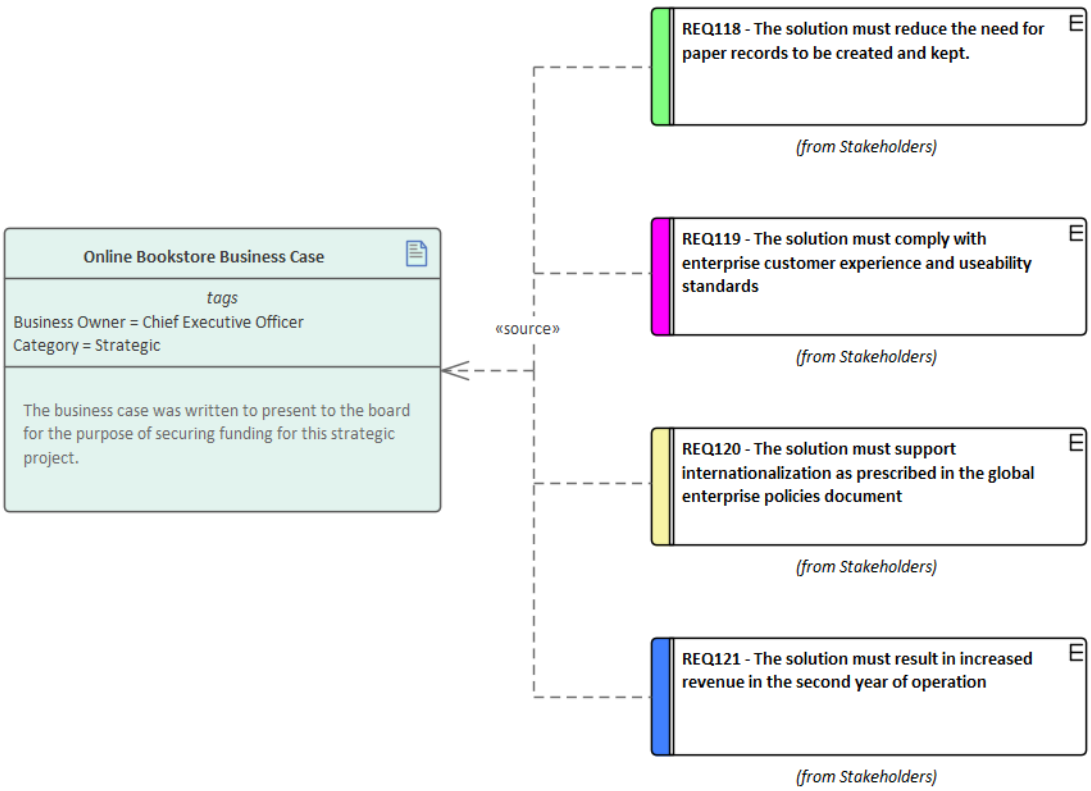
Requirement sources are typically documents or other file based artifacts and are best modeled as UML Artifacts using Tagged Values to add metadata to the element to record

such information as its type, location, who it is governed by and any other relevant properties. The document and files will typically be stored in a variety of places across an organizations networks, but so long as there is a file path or a URL they can be opened from within Enterprise Architect by a single key stroke. The artifact will be opened either inside or outside Enterprise Architect using the appropriate editor for the file type.

Requirements Sources

This diagram shows an external document, "Online Bookstore Business Case", modeled as an artifact. Requirements have been linked back to this artifact, to indicate that the source of the requirement is this document. If the document is subsequently updated, the requirements derived from it are easily located. The Business Case document artifact has a number of Tagged Values indicating properties of the document.

Hyperlinks to external documents can be created by simply dragging and dropping a document file onto a diagram canvas.



Creating artifacts to model requirement sources

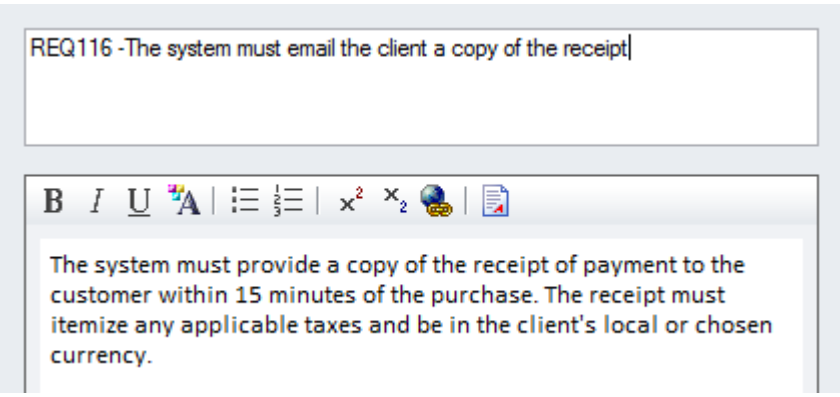
Step	Action
1	Ensure you have a diagram open that will be the canvas where you will create the elements. Open a file system browser such as MS Windows Explorer and drag-and-drop the file to the diagram canvas. A window will be displayed.
2	Select Artifact (External) from the window. Enterprise Architect will add a new Artifact to the diagram and prompt you to enter a name.
3	Enter an appropriate name for the requirement source or leave blank to use the file name for the element name.

Elaborate the Requirements

Once the information that acts as a precursor for the Requirements has been analyzed and the Requirements have been created in Enterprise Architect they must be elaborated with properties and additional information to make them meaningful to the stakeholders who own them and the implementation teams that will use them to design and build the required system.

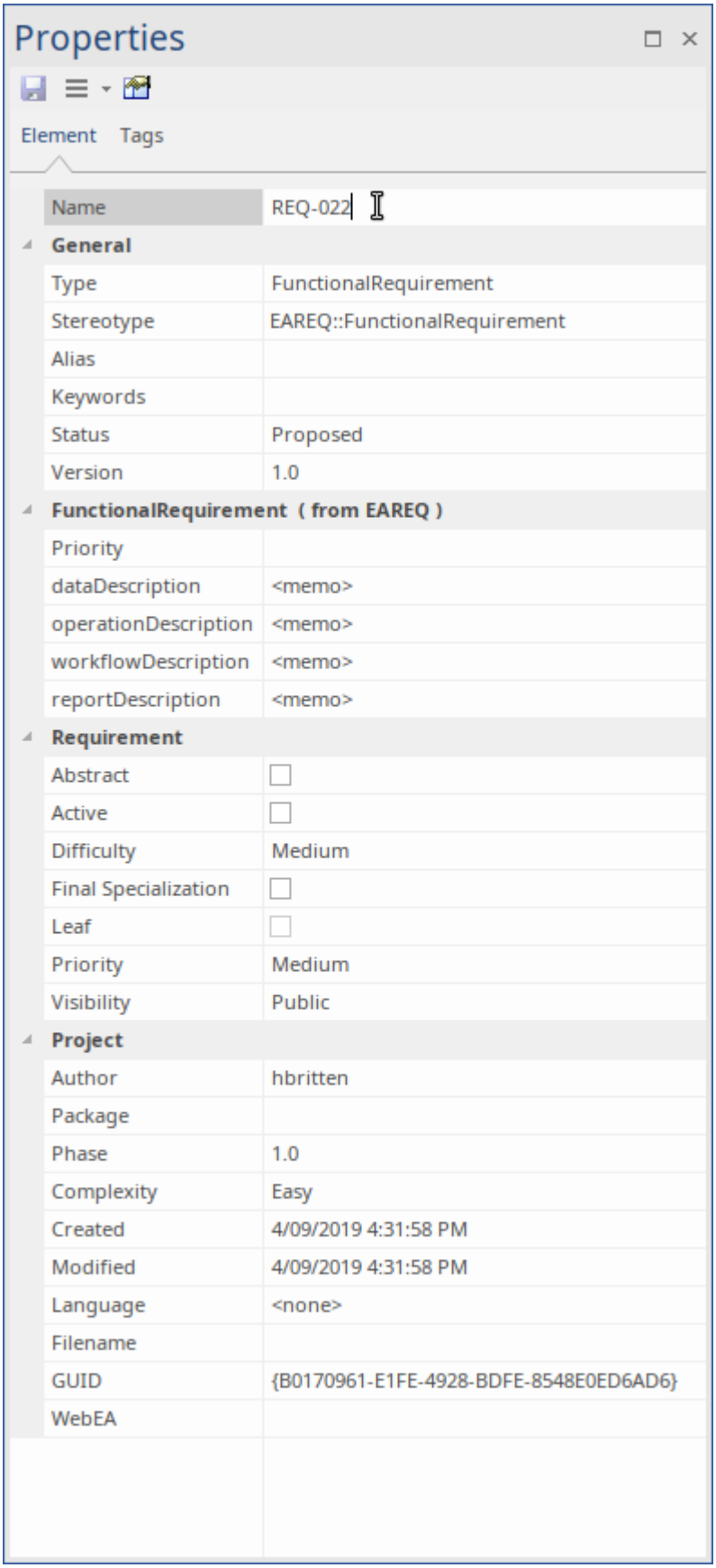
Setting Requirement Properties

When Requirement elements are created they will be given a name, but will often benefit from having additional information recorded about the requirement that will add clarification and details needed by the Stakeholders or the implantation team. These details can be entered into the 'Notes' field for the element.



In addition there are a number of built-in properties that can be set for each Requirement, the most important of which are listed in this table; other user-defined properties can be

added using Tagged Values.



Important Requirement properties

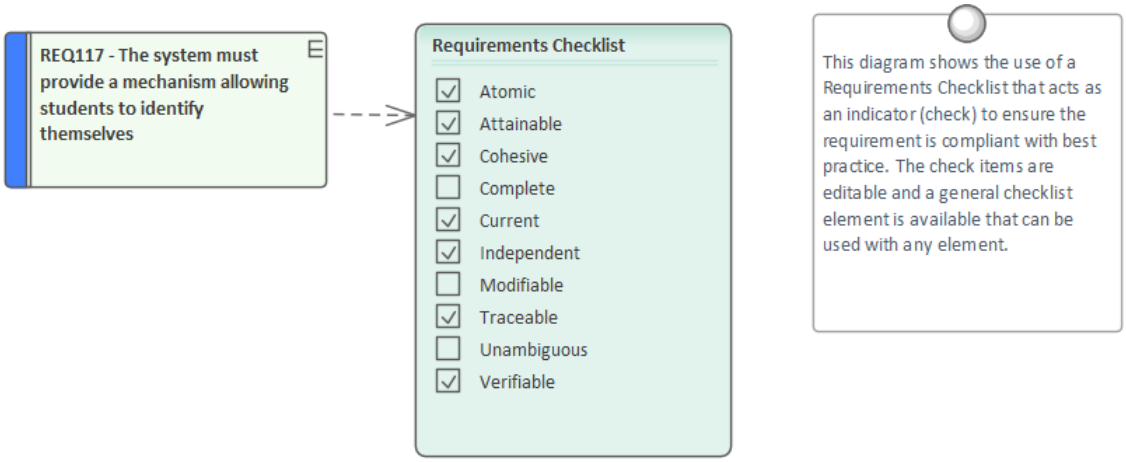
Property	Description
Status	The Status is used to record the condition of the Requirement; the modeler can select it from a drop-down list of predefined values. Setting the Status helps project managers and others involved in planning determine if a Requirement is ready to be included in a development Package of work or iteration.
Alias	The Alias can be used to record a sequence number or identifier for the Requirement. A modeler can choose to display the Alias in a number of locations including diagrams and reports.
Priority	The Priority is used to set the precedence or relative importance of a Requirement; the modeler can select it from a drop-down list of predefined values. Setting the Priority helps project managers and others involved in planning to determine those Requirements that

	should take precedence over others for implementation.
Difficulty	The Difficulty is a measure that attempts to describe the relative effort that would be required to realize the Requirement in a system implementation. The modeler can select a value from a drop-down list of predefined values. Setting the Difficulty helps project managers and others involved in planning determine the effort needed to implement the Requirement.
Phase	The Phase is a value that can be set to indicate the stage of development at which the Requirement will be implemented. The modeler can enter any value into this field. Setting the Phase helps project managers and others involved in planning determine which Package of work or iteration the Requirement should be included in.

Validation

Requirements validation is necessary to make sure the Requirements are of a high standard, suitably define the Customer's problem (or opportunity) and are sufficient for the implementation teams to design and implement the product. It is imperative that the requirements have the desired level of quality and are complete and necessary. There are a number of ways that Requirements can be validated, but probably the two most common ways are to perform team reviews and to assign test cases to the requirements.

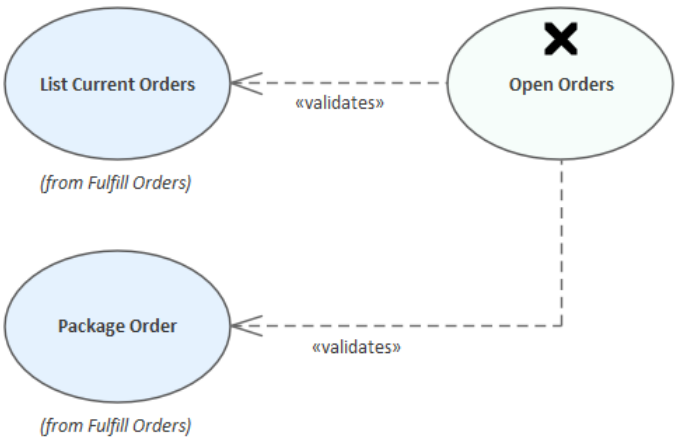
The team reviews are typically conducted by team members or other analysts who have some familiarity with the domain, but were not themselves responsible for the requirements development or management. Enterprise Architect has a handy tool to assist with this process, called the Formal Review, which works across the entire model and allows reviewers to record their findings in discussion documents and to reference model elements. There is also a Requirements Checklist element available from the 'Extended Requirements' page of the Requirements Toolbox, which provides a useful mechanism for checking the quality of Requirements.



Test Cases can be defined at a number of levels from User Acceptance tests down to Unit tests. Defining the test cases early in the requirements development process creates a double check on the Requirements, because when test cases are defined issues with the Requirements are often uncovered. Enterprise Architect has a number of facilities to define test cases and a modeler can select whichever is the most appropriate for the endeavor.

Derive Test Cases

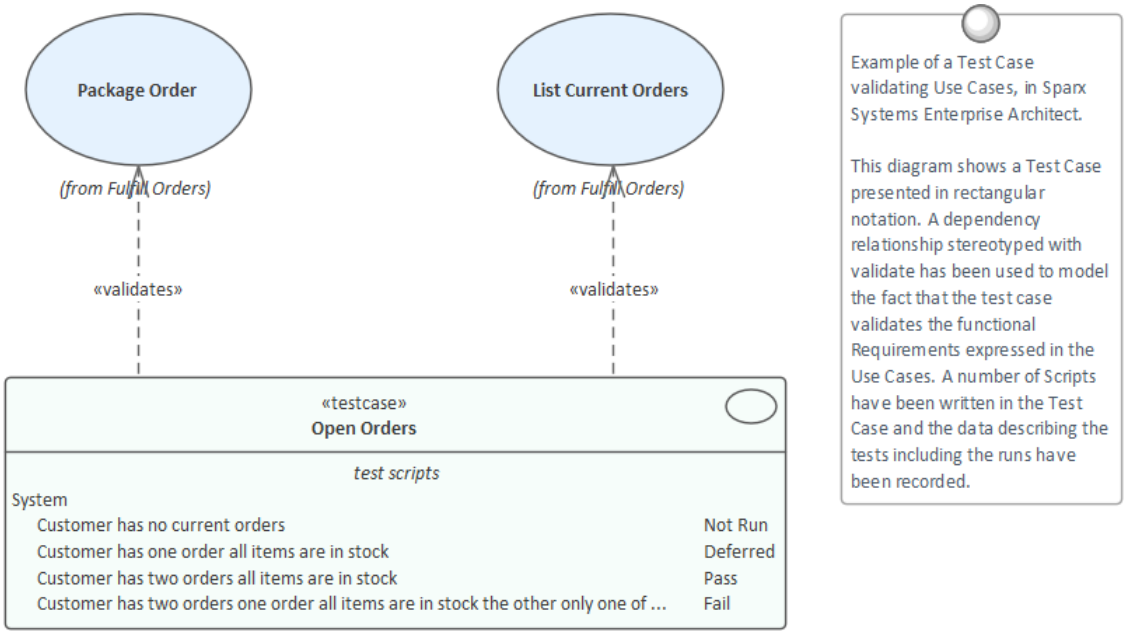
The definition of Test Cases acts as a double check on the quality of Requirements as test designer bring their own unique and often orthogonal view of the requirements. Best practice suggests that the testing team should be independent and isolated from the requirements team thus providing a fresh set of eyes on the Requirements. It is quite common for one Test Case to test a number of requirements or for a Requirement to be tested by one or more Test Cases. These relationships can be modeled in Enterprise Architect using the Test Case element which can have detailed test scripts defined.



Detailing the Test Cases

Any element in Enterprise Architect can have tests defined, which can be one of a number of types - such as Load, Regression or Standard - and from one of a number of classes of test - such as User Acceptance, System and Unit.

The details of the tests, including when they were run and the status of the test indicating whether it passed or failed, can be recorded.

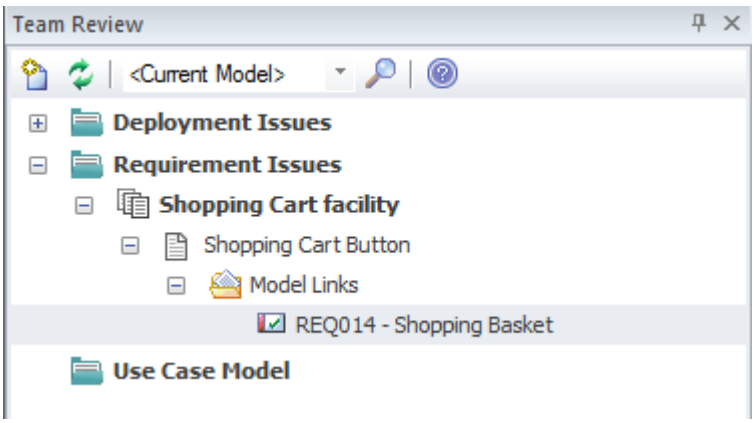


Review Requirements

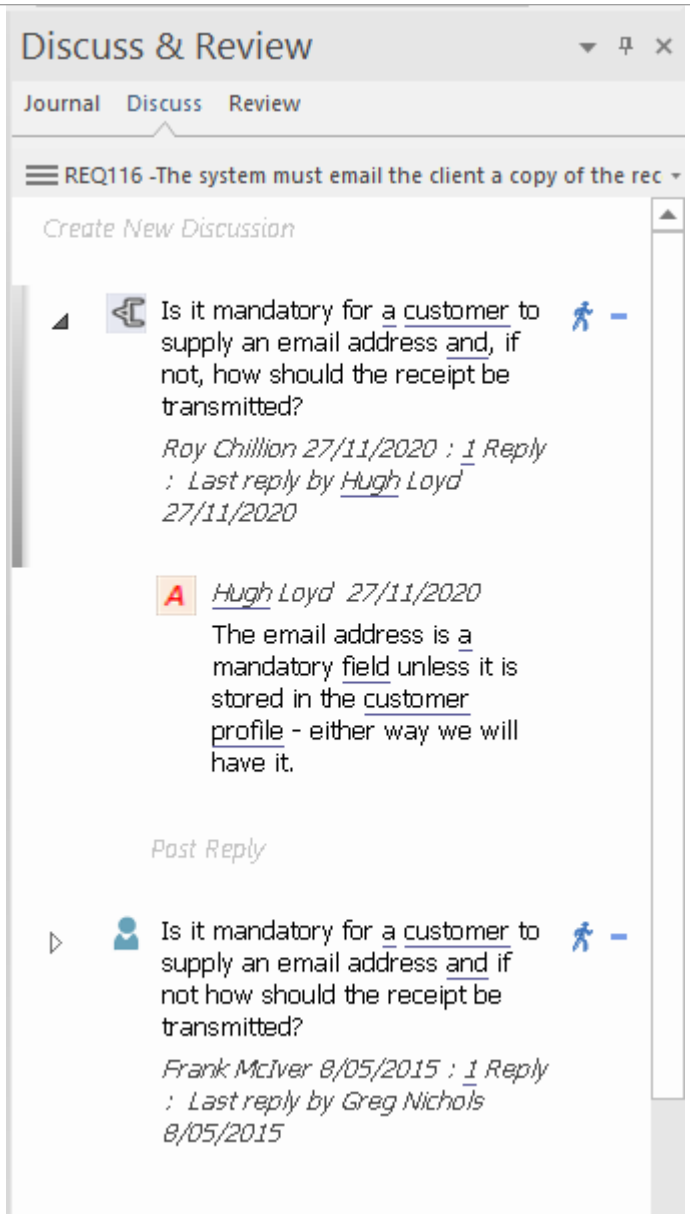
Requirements are typically discovered and written by a variety of team members, and are commonly sourced from a variety of documents and stakeholders at disparate times. All of these factors lead to a tendency for the Requirements to lack quality, be inconsistent and to contain issues. Best practice encourages early and continuous reviews of Requirements to detect and remove defects before the Requirements reach the design and implementation teams. Enterprise Architect provides sophisticated tools for reviewing Requirements including the Model Library tool, Discussions, and Maintenance items.

Requirement Review Tools

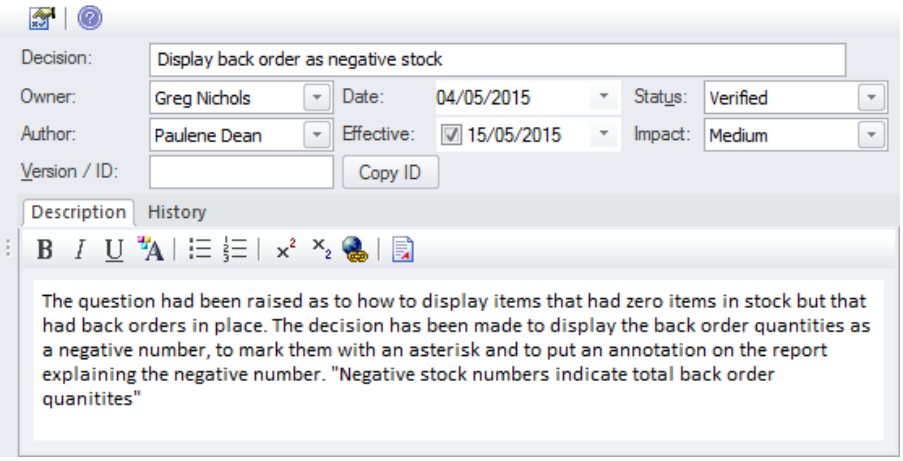
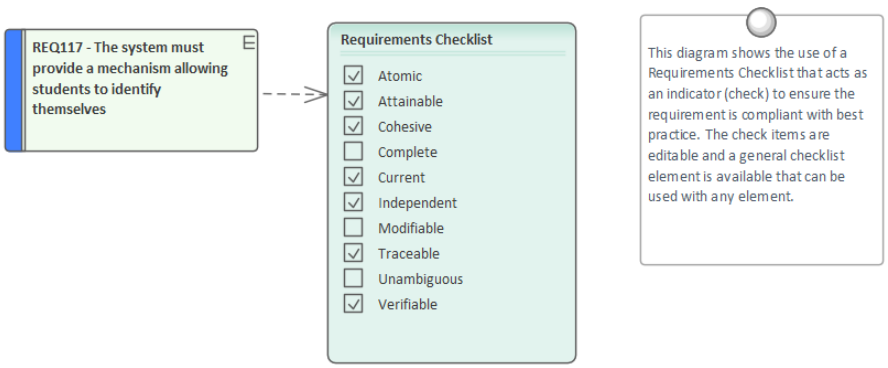
Review Tool	Description
Model Library	The Model Library is a useful tool that allows team members to collaborate and review the contents of the repository. It is particularly useful with requirements validation as it allows a set of requirements to be discussed and reviewed. Each review could be set up as a category and then different types of review could be set up as topics, with posts being used to enter review items.

	Model elements can be added to a post, allowing - for example - a post to report two or more conflicting requirements. Team members would have the chance to post replies and view the linked elements. Statuses can be applied to indicate the progress through a workflow. The Model Library tool can be used in conjunction with the Discuss & Review window (where individual elements are managed) and the Discuss & Review - History window (which contains a composite list of all element discussions in the repository). 
Discussions	During the progress of a Requirements Review there will inevitably be ambiguities, issues and defects detected. Before these are entered against the Requirement as maintenance items, it is common for analysts conducting the review to want to discuss these items with team members or stakeholders. The

	Discuss & Review window is a useful place to record this information, allowing other team members to post replies. This is a particularly convenient facility because in many requirement tools these discussions are typically written in the text of the element, making it difficult to produce customer-ready documentation until they are all removed. It is quite common for modelers to enter discussions even outside a formal review and these will assist the reviewers to resolve Requirement defects or issues.

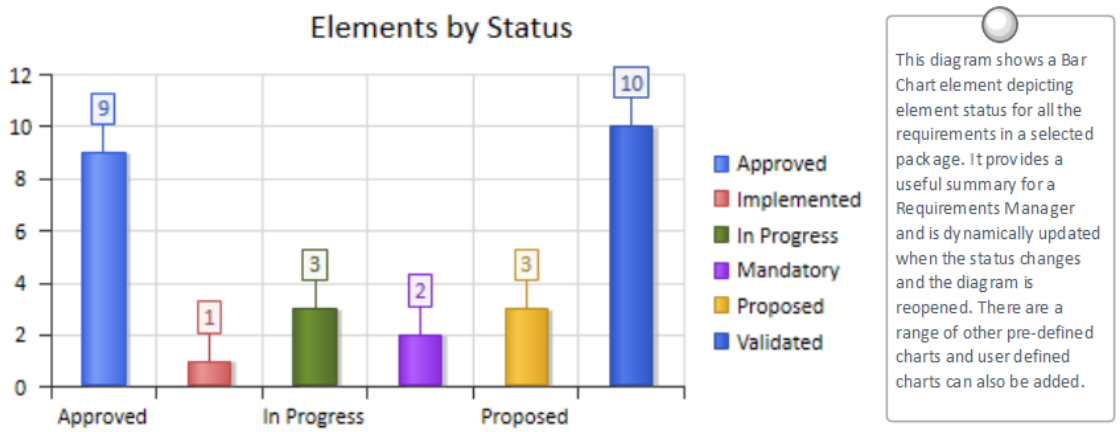


Element Maintenance Items	As Requirements are developed Changes will inevitable occur, Issues will be raised, Events will happen, Defects will be found, Tasks will need to be assigned and Decisions will need to be made. Enterprise Architect has a useful set of facilities to manage these items through the Project Maintenance facility. There is a window tab for each of these

	maintenance items where details of the item such as its Name, Description, Author, Completion Date and more can be assigned. 
Requirement Checklist	Enterprise Architect has a convenient Requirements Checklist element available from the 'Extended Requirements' page of the Requirements Toolbox. This is a convenient way of recording visually whether a Requirement is of high quality and is a useful mechanism for team reviews. 

Requirement Management

This consists of the activities to maintain a set of requirements that represent an accord or agreement between the project team and the customer. It also involves ensuring that the requirements are acceptable to the design and implementation teams and that they are sufficient so that what they specify can be implemented into working business, software or hardware systems. Enterprise Architect is a sophisticated platform for managing requirements, and regardless of the domain, the size of the project or the method being followed, there are tools that will make it straightforward to manage even large repositories of requirements in complex projects.



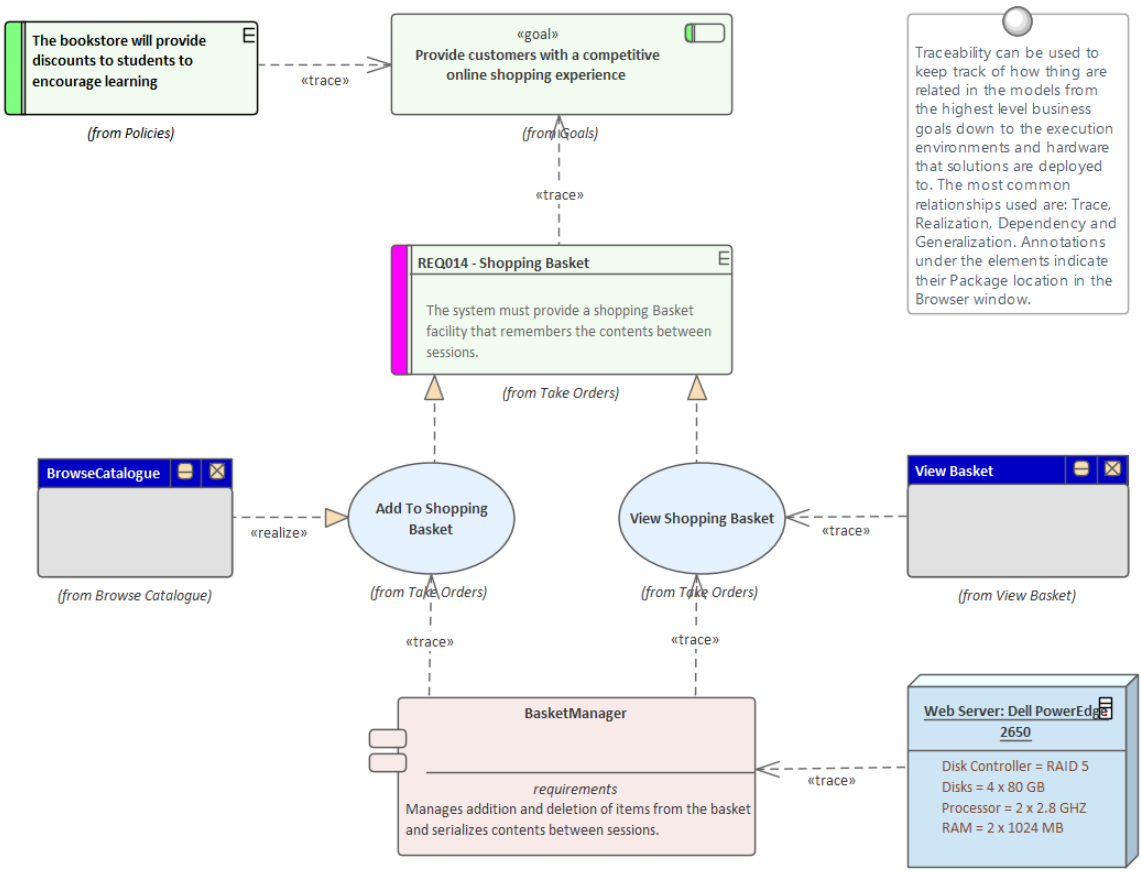
Tracing Requirements

Most Requirement processes mandate that Requirements are traced from high level concepts such as Business Drivers, Visions and Goals down to the parts of Components that implement them. For many projects this is an intractable problem because much of the information lives in a set of heterogeneous tools such as word processor documents, spreadsheets, diagram tools, corporate presentation tools and more. Some Project Managers attempt to solve the problem by creating a spreadsheet that acts as a register of all the disparate information but the management of this file takes up considerable project resources and the file is almost impossible to keep up to date. With Enterprise Architect there is the ability to model all of this project information in the one tool and to create easy-to-maintain and analyzable traces between all the elements, from the organization's mission statement right down to the level of programming code, if required.

Visualizing Traces in diagrams

Regardless of whether you have entered the project's Requirements using a diagram, using a text-based tool such as the Specification Manager, or by importing them from another tool, viewing the requirement traces in a diagram gives an easily understood view of their relationships. The diagrams can be created easily by dragging and dropping

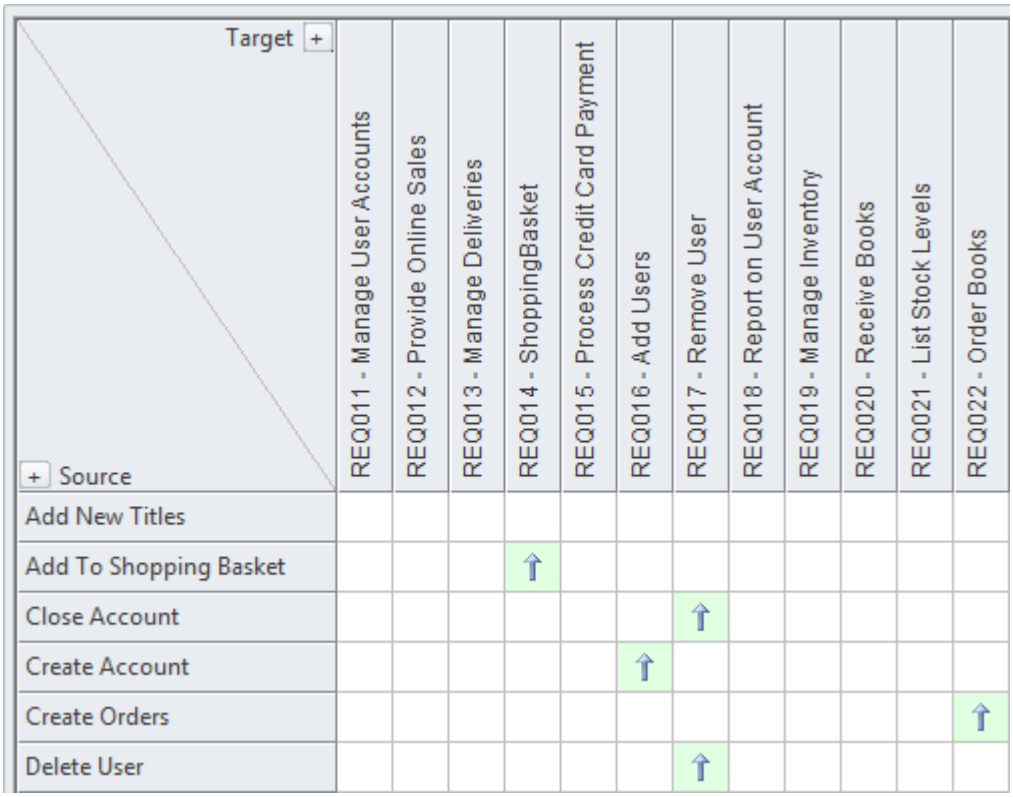
elements from the Browser window, or automatically by using the 'Insert Related Elements' option. This function can be configured and used to draw a graph of elements to any depth, and can be restricted to selected types of element and connector. It is a handy productivity tool in a team environment, and even modelers with deep knowledge of the domain and the repository are surprised at the connections that are displayed in the diagrams.



Visualizing Traces using the Relationship Matrix

The Relationship Matrix provides an alternative way of

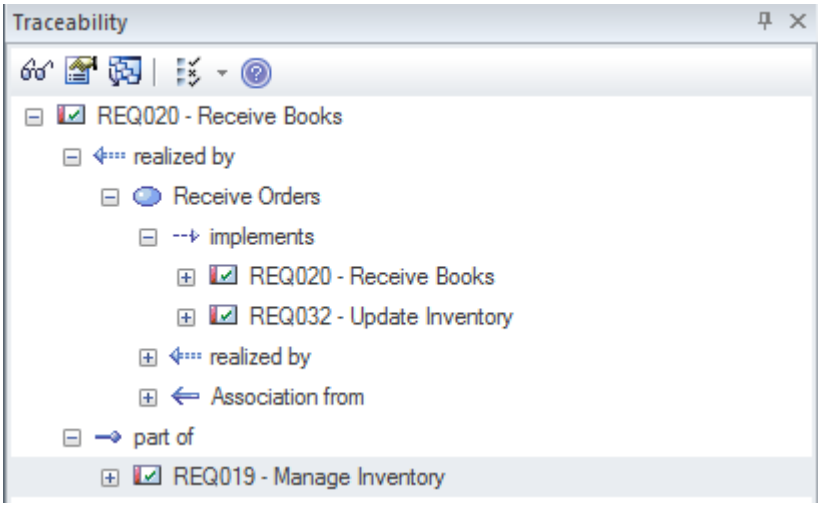
visualizing the relationships between Requirements and other elements, or even between different levels or types of Requirement. It is quite common for some stakeholders to prefer a spreadsheet-like view of the Requirements and their relationships, and the Relationship Matrix provides an excellent way of presenting the relationships without resorting to a diagram. In Use Case driven requirements methods, Use Cases are said to realize one or more Requirements, and these relationships can be displayed visually in the Relationship Matrix. The list of Use Cases would appear on one axis of the matrix and the Requirements would be listed on the other axis. A marker in the intersection of a row and column would display if a relationship exists, indicating that a particular Use Case realizes a Requirement. Relationships between elements can be created or deleted using the Relationship Matrix, and the Matrix can be saved and reopened at any time or saved to a CSV file so it could be opened in a spreadsheet. Documentation can also be created that includes the Relationship Matrix, providing a useful communication tool for people who do not have access to the model.



Source \ Target	REQ011 - Manage User Accounts	REQ012 - Provide Online Sales	REQ013 - Manage Deliveries	REQ014 - ShoppingBasket	REQ015 - Process Credit Card Payment	REQ016 - Add Users	REQ017 - Remove User	REQ018 - Report on User Account	REQ019 - Manage Inventory	REQ020 - Receive Books	REQ021 - List Stock Levels	REQ022 - Order Books
Add New Titles												
Add To Shopping Basket				↑								
Close Account							↑					
Create Account						↑						
Create Orders												↑
Delete User							↑					

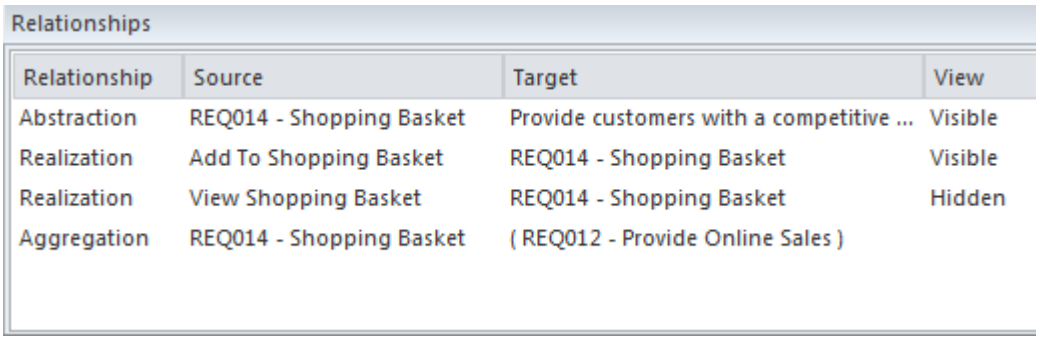
Visualizing Traces using the Traceability Window

While diagrams and the Relationship Matrix allow modelers to view traces between requirement elements it is possible that the creators of these views of the repository have deliberately omitted elements from the view. For example a diagram does not need to show all the requirements owned by a particular stakeholder. The Traceability window will, however, present a complete and unabridged view of the relationships between elements. The element relationships will be displayed regardless of the location of the elements in the Browser window.



Visualizing Traces using the Relationships Window

Modelers often choose to hide one or more relationships on a diagram for the purpose of making the diagram simpler to understand or to hide detail. The Relationships window is a useful window to have open as it will display all the relationships that exist between the elements in the diagram indicating whether they are visible or hidden in the diagram.



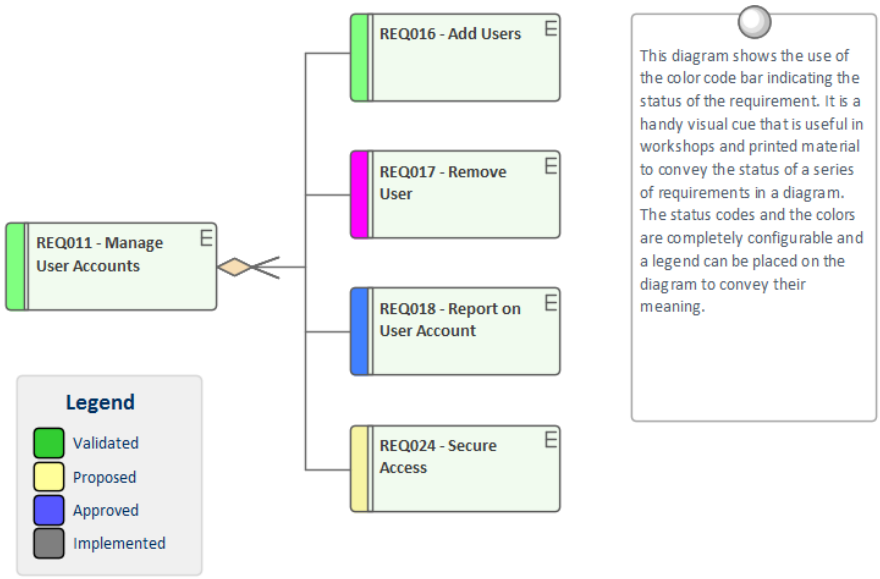
If relationships have been hidden in a diagram they can be made visible by selecting the 'Show Relationships' option on the 'Connectors' page of the 'Properties' dialog for the diagram.

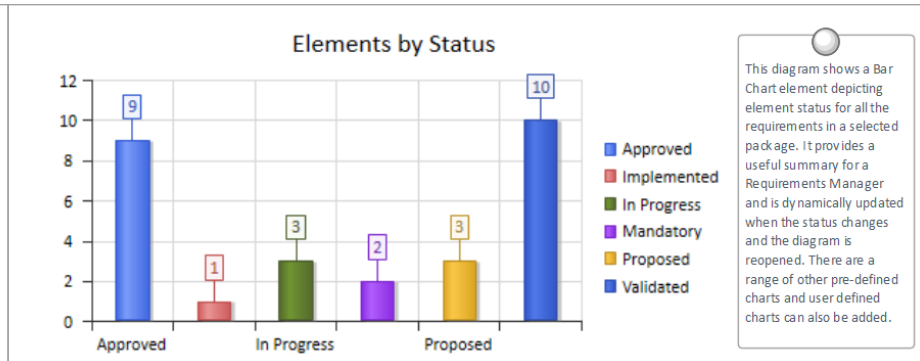
Tracking Requirements

The status of a requirement is a fundamental indicator of where it is positioned in the requirement's development process. For example requirements that have a status of 'Proposed' indicate that they are not yet ready and available for development work to begin. Enterprise Architect has a variety of tools to allow status to be tracked, analyzed and managed, starting with the fact that each requirement can be assigned a status and the list of status codes are completely configurable. The status is conveniently displayed in list views of the requirements including when using the Specification Manager. There are also a set of pre-defined and extensible dashboard charts and graphs that can be used to get a compelling visual representation of the status and other properties of requirements.

Tools for tracking requirements

Tool	Description
Status Codes	Status codes are a controlled list of statuses that can be applied to any element including Requirements. Enterprise Architect comes with a pre-defined list of codes but the list can be configured and codes in the list can be changed and deleted and new codes can

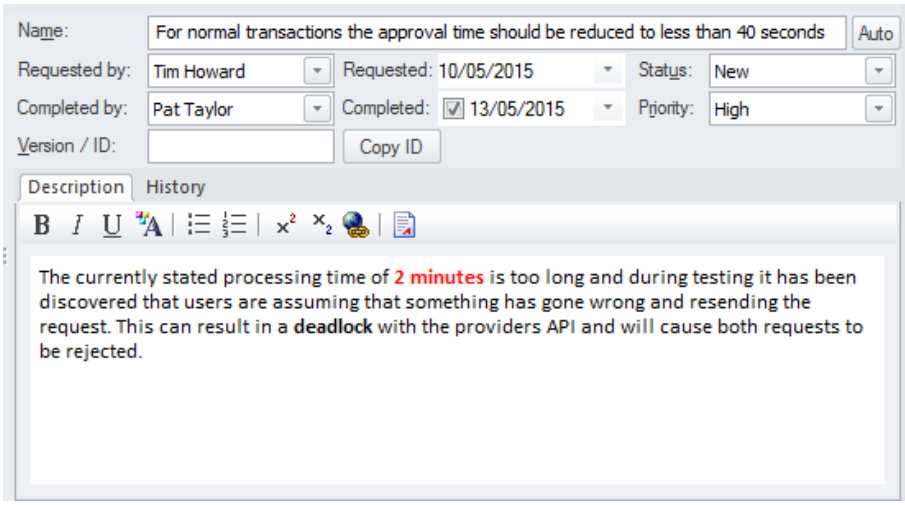
	be added. The status of Requirements can be displayed in a diagram as a color coded band on the side of the element. 
Dashboards charts and graphs	Dashboard diagrams are an extended diagram type and allow high quality charts and graphs to be created to display repository information in a visually compelling way. Any number of diagrams and charts can be created and the data can be sourced from any level in the repository Package hierarchy. Enterprise Architect comes with a toolbox page of pre-configured charts and graphs, but new charts can be created based on any information in the repository.


Elements by Status
12
10
8
6
4
2
0
9
1
3
2
3
10
Approved
In Progress
Proposed
Approved
Implemented
In Progress
Mandatory
Proposed
Validated
This diagram shows a Bar Chart element depicting element status for all the requirements in a selected package. It provides a useful summary for a Requirements Manager and is dynamically updated when the status changes and the diagram is reopened. There are a range of other pre-defined charts and user defined charts can also be added.

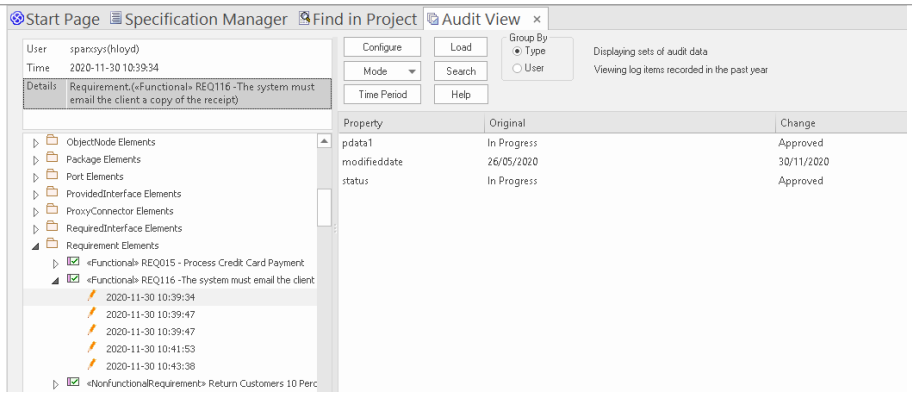
Managing Changing Requirements

It is inevitable that requirements will change during the specification and solution phases of a project, and most requirements management processes have some type of mechanisms for embracing these changes. Typically, a set of requirements will have been specified and groomed for the solution teams to implement; any subsequent changes are specified as Change Requests. Regardless of the rigor of the process being used, inadvertent changes will occur that need to be managed along with the Change Requests. Enterprise Architect is a sophisticated requirements management platform, with a range of tools to assist the requirements manager. Change Requests can be managed in the Maintenance window, which allows the requested change to be recorded and described, along with whoever requested it and when it was done and whoever completed the change. Inadvertent changes can be discovered and analyzed using a number of tool features, including Auditing, Baselines and Version Control; these tools have some overlapping features and can be used in isolation or together. The built-in Security system will also assist in preventing inadvertent changes to models, by allowing modelers to intentionally lock Packages and elements in the model.

Mechanisms for managing changing requirements

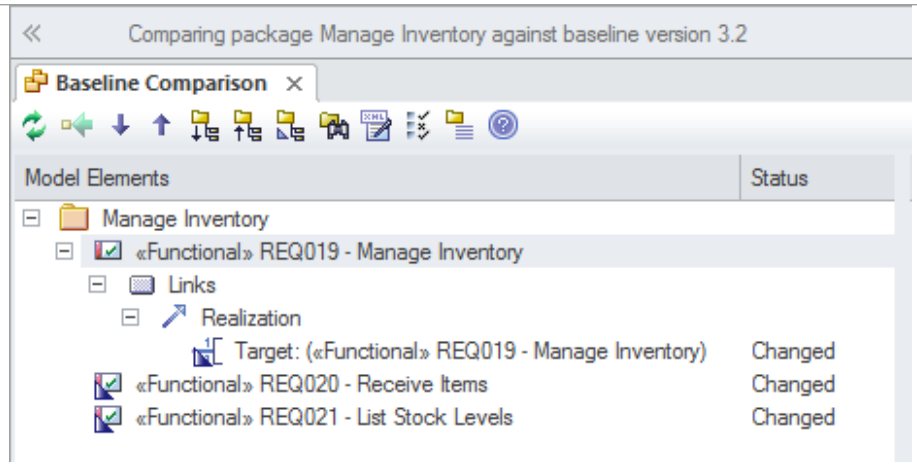
Mechanism	Description
Element change task and effort items	Changes to requirements can happen inadvertently but it is more common for there to be an intentional change in response to a wide variety of factors such as Stakeholders revising their needs, the business changing or a problem being poorly understood. Inadvertent changes can be picked up using a number of tools but deliberate changes can be assigned using the Change item, which can be recorded against each element. Once the impact of the change has been analyzed Tasks can be created to specify what needs to be done to implement the change and Effort can be assigned using the Requirements Effort item. 
Auditing	Auditing is a built-in tool that, when

	enabled, automatically records changes to the repository. It has a number of different modes and views, and can be configured to assist in the management of Requirements. It can track what was changed in the model, who made the change and when it was made, showing the before and after views. So if the text of a Requirement was updated or its status was changed, this would be recorded. Auditing functionality overlaps with the Baseline tool, but unlike the Baseline tool the changes are being recorded automatically and every discreet change is recorded. In contrast, the Baseline tool will only compare the current model to a Baseline regardless of how many intervening changes had been made. Auditing will not assist with the impact of the changes but just what changes have occurred. Once the changes have been established, tools such as the Relationship Matrix can be used to determine the impact.



Version Control	Version Control can be implemented in Enterprise Architect to manage changes and revisions to any Package including Requirements Packages. Once implemented changes to Requirements will be recorded and a requirements analyst will be able to view previous version and roll back to these versions if required. There is some overlap between this tool feature and Auditing and Baselines. The difference between this facility and Auditing is that Auditing simply records the changes but does not allow you to revert to a previous version. The difference between Version Control and Baselines is that a modeler must intentionally create a baseline whereas with Version Control the changes are being recorded automatically in the background. Also with Baselines the intervening changes are not recorded, just the difference between the current

	requirement and the one captured in the Baseline.
Baselines	Baselines provide a versatile mechanism for managing changes to Requirements. Any number of baselines can be created and when requirements are changed these changed requirements can be compared to one of the baselines. Baselines are typically created at important milestones in a project such as after a stakeholder meeting or before a development iteration is commenced. When differences are found and these changes were not intended or contravene project management practice the requirements from the baseline can be restored to the current model. Baselines will not help with assessing the impact of a change but once a change has been identified tools such as the Relationship Matrix and element traces can be used to determine the impact of a change.

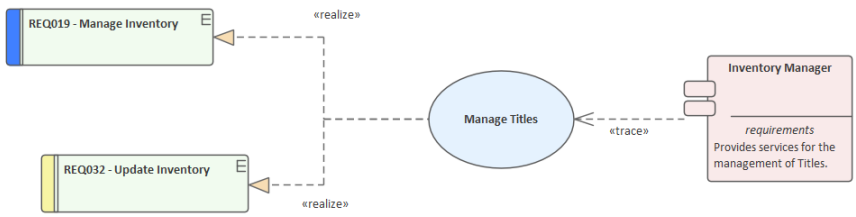

Comparing package Manage Inventory against baseline version 3.2
Baseline Comparison
Model Elements
Status
Manage Inventory
«Functional» REQ019 - Manage Inventory
Links
Realization
Target: («Functional» REQ019 - Manage Inventory)
Changed
«Functional» REQ020 - Receive Items
Changed
«Functional» REQ021 - List Stock Levels
Changed

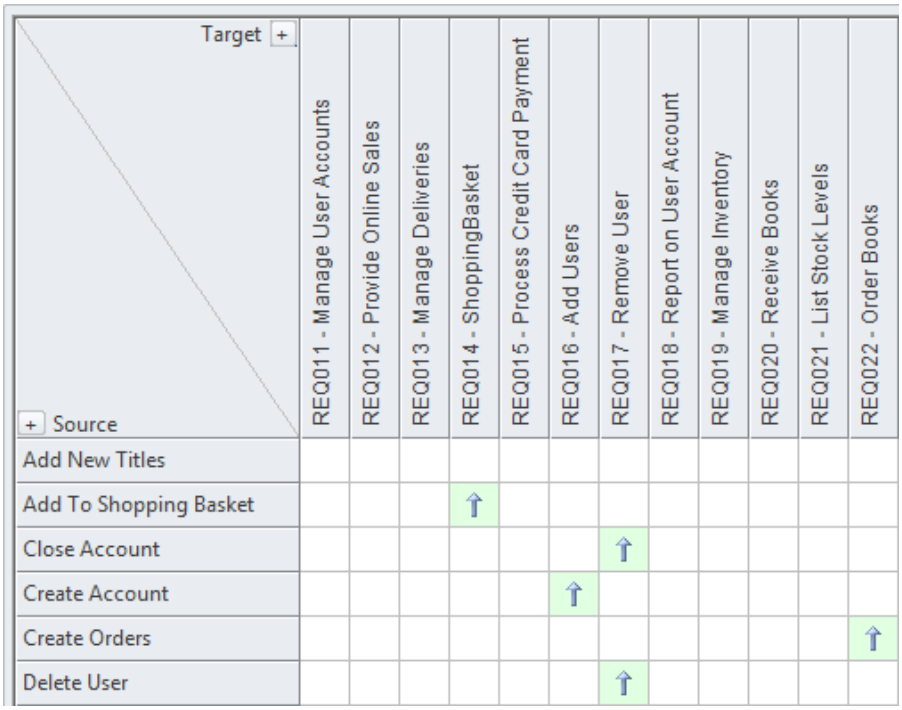
Impact Analysis of Changes

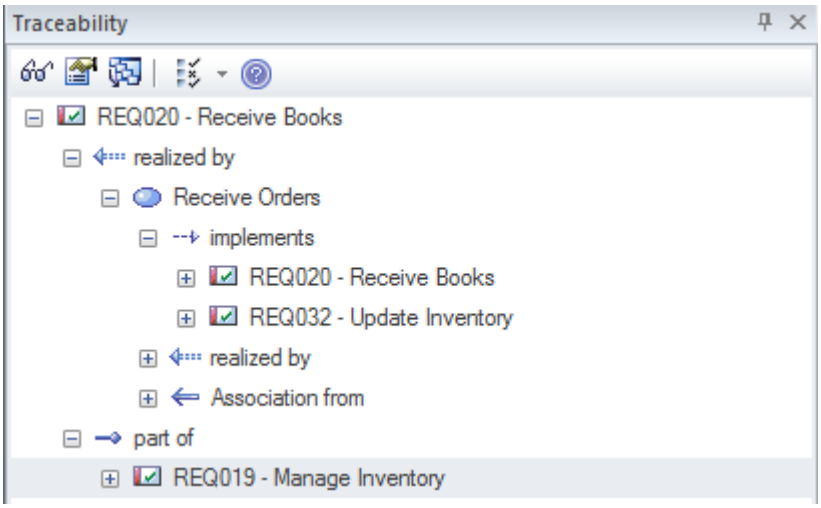
When the development of a system has started and requirements change there will be an impact of the change and the effect will need to be determined, understood and managed. Having traceability established both to up-process elements such as Stakeholders and Business Drivers and down-process elements such as Use Cases, Components, Test Cases and source code operations is critical to determining the impact of the change. Enterprise Architect has a number of facilities that can assist with this including the ability to visualize traces in diagrams, a Relationship Matrix, a Traceability window, element Change, Task and Effort items that can be used to record impact and what is required to implement it.

Tools to record and analyze the impact of change

Tool	Description
Analysis using requirement traces	The ability to visualize requirements and the way they are connected to other elements is a practical tool for analyzing the impact of change. Requirements often form a hierarchy and when one requirement is affected it typically has a

	ripple effect to the requirement's children and being able to visualize this in a diagram or in a hierarch is very useful. Requirements are also typically traced to up-process and down process elements and a diagram provides a way of viewing and analyzing these connections. The Insert Related Elements function can discover these connections and automatically draw and layout a diagram allowing the modeler to spend their time analyzing the impact. Tracing Requirements This diagram shows the expressive power of putting disparate elements onto a diagram. It shows the traceability between different layers of a system. The traceability can be from the Requirements to the Use Cases that Realize them, to the logical Components that will deliver the required functionality. 
Analysis using a relationship matrix	The Relationship Matrix can be used to visualize the requirements and their connections by placing the Requirement on one axis of the matrix and the connected elements on the other. This method is very useful in workshops when working with people who might not be

	familiar with modeling languages such as UML or who work better with spreadsheet types of view. Any number of matrices can be created and their specification can be stored so they can easily be recalled. 
Analysis using the traceability window	The Traceability window is a handy window that shows the hierarchy of elements in the Repository. It is particularly useful because it unconditionally shows how elements are related to each other. Other views of the repository could be configured just to display particular elements for the purpose of communicating an idea whereas the Traceability window will display all relationship that an element

	participates in which makes it particularly useful for analyzing the impact of change. 

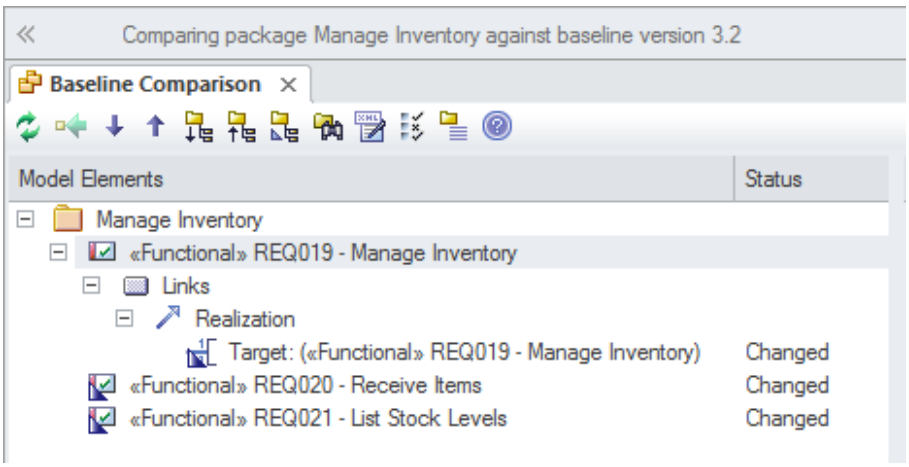
Requirement Volatility

There are ever increasing market place pressures to release products and systems as early as possible, putting stress on project teams to develop, test and deploy products in shorter and shorter periods of time. The requirements processes have changed significantly in recent years to ensure that stable, correct and well-articulated specifications are provided to architects, designers and developers when they need them. There has been a move to iterative and incremental processes and this necessitates providing a set of stable requirements for every iteration. The churning of requirements is often an indicator that a problem is not clearly understood, that stakeholders have not been compromised and there are unresolved political issues, the scope is not defined or the business itself is in fluctuation. Enterprise Architect has a number of mechanisms that can be used to assist with this problem. Enterprise Architect does not have a built-in property for requirement volatility (stability) but using the general purpose UML extension mechanism of Tagged Values a tag could be created to record this property.

Note: Internal requirements do have a stability property but external requirements do not.

Mechanisms for managing requirement volatitlity

Mechanism	Description
Volatility as a Tagged Value	Enterprise Architect provides a series of properties for Requirements, but additional properties can be created to record other properties such as a Requirement's volatility or the source of the Requirement. This is achieved using the UML Tagged Value mechanism, which allows any element including Requirements to have one or more tags applied, representing some property that can be assigned a value. Enterprise Architect has extended this mechanism and allows the modeler to create a list of values that can be chosen from a drop down list using the Predefined Structured Tagged Values. This allows a team to define their own list of volatility values, such as extreme, high, medium low, minimal. 
Using	The Baseline facility is an effective tool that enables a user to take a snapshot of a

Baselines	model or more typically a model fragment and then as the model is developed to compare the new version of the model to the baseline thus identifying anything that has changed since the baseline was taken. Baselines have general applicability but are particularly useful with requirements management where requirements are often said to be signed-off or frozen and any alterations to them must be registered as a change. The Baseline tool has a Compare utility that conveniently lists changes between the current model and the baseline. 
Searches for churning requirements	Enterprise Architect has a sophisticated search facility that allows a user to search across either a selected Package or the entire repository, to locate elements that meet fine-grained criteria. This can be used to locate requirements that have not changed by searching for a change in the

	modification date before a specified date, thus providing a list of stable requirements. Alternatively, if volatility has been set using a Tagged Value, all elements with a specified volatility could be located. The search facility returns a list of elements that can be located in the Browser window; the search can be used as the basis of a Model View to be used to view either volatile or non-volatile requirements.

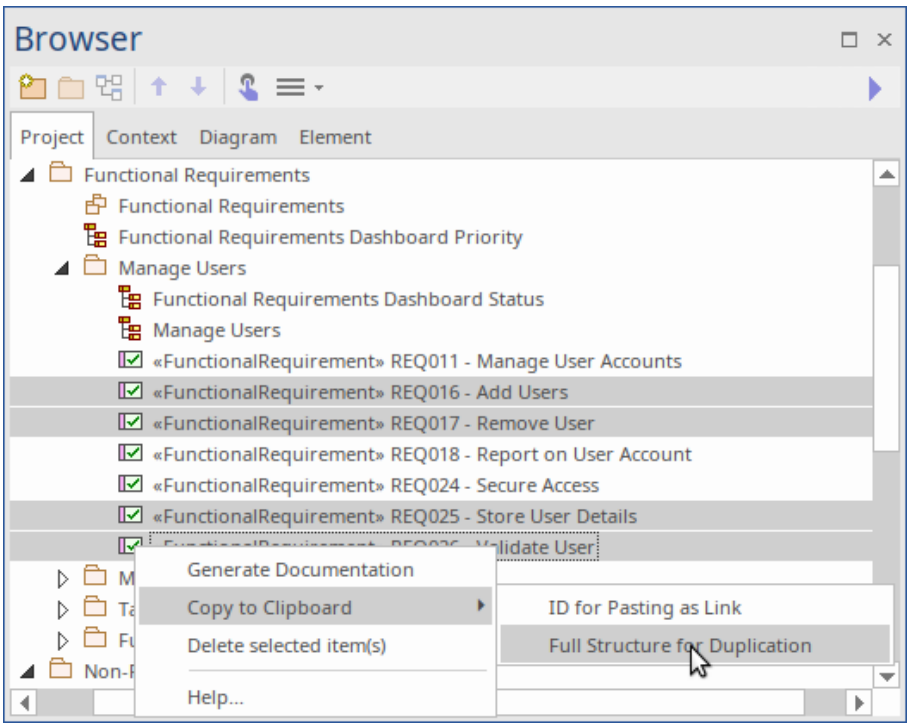
Requirement Reuse

The concept of reusing artifacts of a system development process has been written about in many papers and text books but has traditionally been confined to software components. In more recent years the notion of reusing specifications, including requirements, has started to get traction. The reuse is particularly important where organizations create a family of products with similar features, or where there is a community of users within an industry or domain. Other types of requirement such as security and regulatory requirements will typically apply to a number of projects. Business Rules and Stakeholders Concerns will also typically apply across many projects and are best catalogued outside individual project structures. Enterprise Architect provides a number of sophisticated mechanisms for managing the reuse of elements across projects, including structuring the repository for reuse, importing requirements from other sources, and a Reusable Asset Service.

Mechanism for requirements reuse

Mechanism	Description
Structuring the repository for	When you set up a repository, you have the choice of structuring it for a single project or for multiple projects, which in

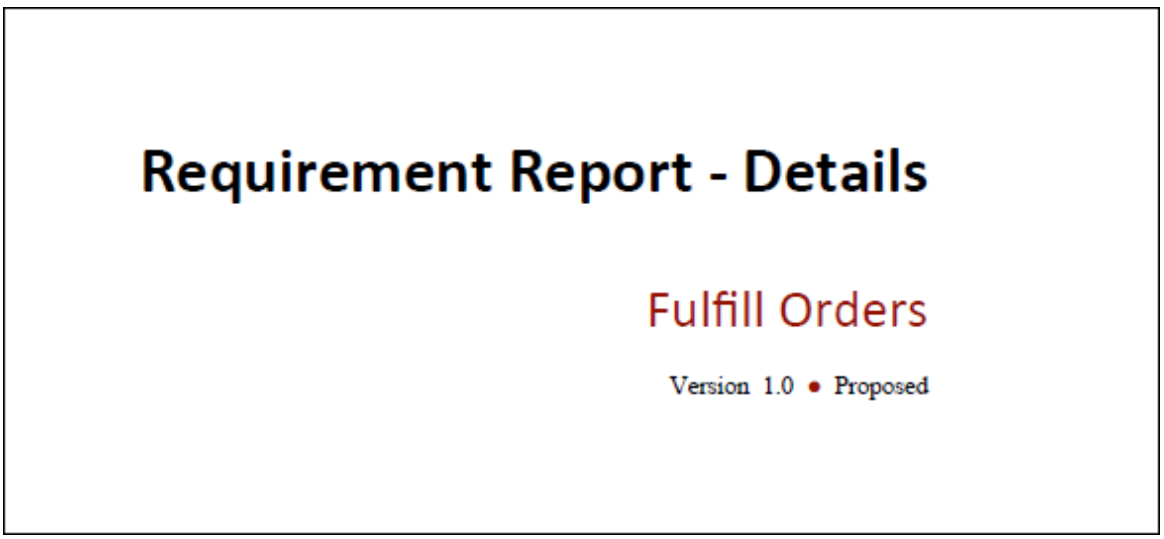
requirements reuse	turn could be organized by a number of programs of work. Enterprise Architect gives the modeler complete control on how the repository is structured, allowing Packages to be set up above the level of projects where some requirements such as Business, Regulatory and Architectural Requirements can be added.
Creating a base model	When you create a new repository in Enterprise Architect, you have the option of creating a blank model using the Model Wizard (Start Page 'Create from Pattern' tab) to help set up a repository structure, or you can use a base model as a template for the new model. The base model is a good place to store reusable assets such as Business, Regulatory and Architectural Requirements, and Policies and Business Rules. A base model can be created using the 'Save Project As...' option under the 'File Management' icon ().
Importing requirements from other models	It is quite common to have a number of Enterprise Architect Repositories in an organization and it is very easy to copy and reuse Requirements (or any other elements) from one model in another. This can be achieved by simply copying a

	selection of Requirements or an entire Package from one repository to another, or even from one project to another in the same repository. Enterprise Architect works in the same way as any other Windows program, simply copying the selection to the clipboard and then allowing it to be pasted in another location in the same model or in another open repository. 
Using the Reusable Asset Service	The Reusable Asset Service (RAS) is particularly useful for distributed teams and provides a simple and convenient mechanism for modelers to distribute or download reusable model structures and elements such as Requirements through a shared repository, accessible via a Pro

	Cloud Server connection. Enterprise or organizational level Requirements could be stored in the RAS and different teams could incorporate them into their models, governance of the assets would typically be managed by the owner of the asset (register) at the Reusable Asset Service level.

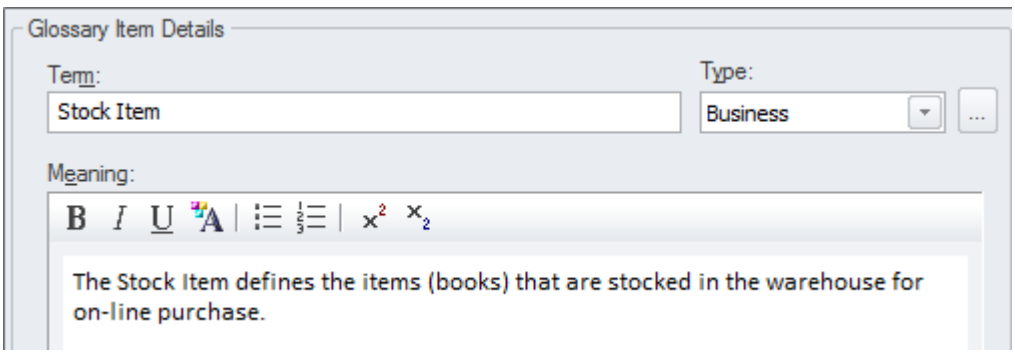
Requirement Documentation

A number of documents are commonly produced as part of the Requirements Engineering discipline, such as the Software (System) Requirements Specification and Use Case Report. These can be generated automatically from a requirements model using built-in templates. In addition a wide range of other documents can be produced using built-in or customized templates. The documentation facility in Enterprise Architect is highly configurable and many reports can be produced using the template system, but for more complex reports there is a facility called Virtual Documents that allows a publisher to model the structure of the document and to cherry pick content from anywhere in the repository, applying different templates to each section of the document. There is also a wide variety of options that can be applied at the template or document generation level, and the Scripting engine can be used to inject content into a document or to produce the entire report.



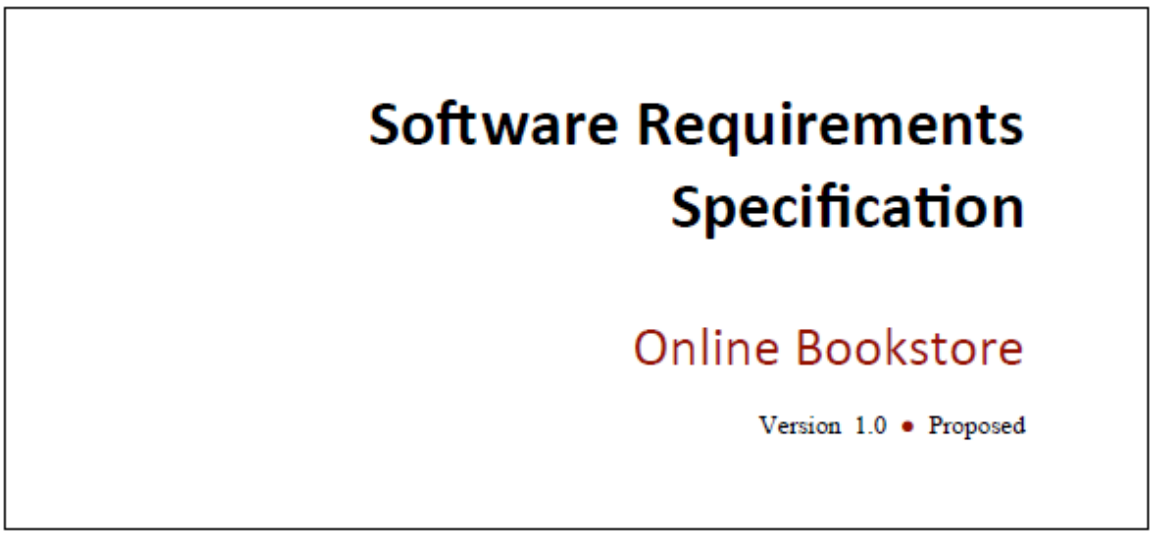
Project Glossary

A Project Glossary lists and defines the terms that are important for a project or program of work. The Project Glossary can be generated as an isolated document, or it can be included as a section in one or more other documents. It provides a single point of truth for the important project terms and their meanings; when new documentation is generated the terms will automatically be updated. The Glossary can be generated to a DOCX or PDF format, or to HTML that could be included in a project or organization level web site. The Glossary allows the modeler to categorize the terms into user-defined Types, and these can have styles applied when they are generated in documentation.



Software Requirement Specification

This document describes the Requirements of the system, its behavior under defined conditions, and the constraints that it must operate under; it will typically be read by a variety of stakeholders. There is a built-in Requirements template that can be used to generate the document, although the modeler is free to create a new template that could be either based on this or created from a blank template. When the document has content from a variety of locations in the Browser window, it would be most expedient to use the Virtual Documents facility, which allows the user to create a model of the document (similar to a Master document in a Word Processor) that has a number of sections called Model Documents. These can have content picked from anywhere in the Browser window.

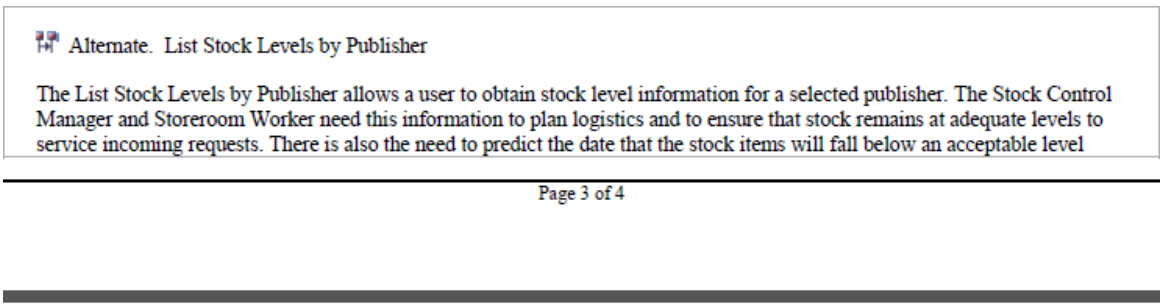


Use Case Report

The creation of Use Case documentation has traditionally been a manual process and with the documents in many projects running into hundreds of pages their production consumes valuable project resources. These hand-crafted documents become difficult to maintain and remain isolated from other parts of the project such as Requirements, Business Rules and solution Components. Enterprise Architect has a multi-featured tool called the Scenario Builder that allows the modeler to specify Use Cases and Scenarios inside the model and these can be automatically generated to high quality documentation using built-in templates. There are two built-in templates that can be used for generating a Use Case report: one documents the Use Case at a summary level and the other at a detailed level.

Example content from a Use Case Report

The detailed Use Case report will list all the details of the Use Case and the detailed steps, including Basic Paths, Alternate and Exception Scenarios. Other information, including Internal Requirements, Pre and Post Conditions and other Constraints will also be included in the report. If a Behavioral diagram such as an Activity diagram has been automatically created, this diagram will also be displayed in the report.



Alternate. List Stock Levels by Publisher

The List Stock Levels by Publisher allows a user to obtain stock level information for a selected publisher. The Stock Control Manager and Storeroom Worker need this information to plan logistics and to ensure that stock remains at adequate levels to service incoming requests. There is also the need to predict the date that the stock items will fall below an acceptable level

Page 3 of 4

Use Case Details 19 May, 2015

SCENARIOS

based on purchase cycles and promotional periods.

1. User selects "List Stock Levels by Publisher"
   Uses:

2. System returns a list of publishers to select from
   Uses:

3. User Selects a publisher
   Uses:

4. System returns a listing of titles and quantity in stock for the publisher
   Uses:

Data Dictionary

Many processes specify the creation of a Data Dictionary that acts as a reference for all the information that will be consumed, stored or created by the system. The Data Dictionary can be created in Enterprise Architect using the UML Class diagram and Classes can be defined to represent the important elements of the domain including Attributes and Data Types. Enterprise Architect can generate high quality documentation from the Class diagram and data items and their descriptions can be listed and the Attributes of these Classes can also be detailed with Data Types and Multiplicities. Enterprise Architect provides built-in templates that allow the Data Dictionary to be created automatically and these templates can be edited or new ones created.

Account

Class in package 'Domain Model'

The Account defines the user details that are stored as clients. It provides all the information required for billing and product delivery and other information can be related to this record such as preferences and alerts. it is self-managed in the sense that clients are able to keep their own information up-to-date through a web interface ensuring that the system has the latest information.

Account
Version 1.0 Phase 1.0 Proposed
Benjamin Hutton created on 17/03/2005. Last modified 15/05/2015

INCOMING STRUCTURAL RELATIONSHIPS

Page 4 of 12

Data Modeling Report 15 May, 2015

INCOMING STRUCTURAL RELATIONSHIPS

Collaboration from «control» View Account Details to Account
[Name is getAccountDetails(). Direction is 'Source -> Destination'.]

Collaboration from «control» Delete User to Account
[Name is retrieveAccountDetails(). Direction is 'Source -> Destination'.]

Collaboration from «control» Create New Account to Account
[Name is submitNewAccountDetails(). Direction is 'Source -> Destination'.]

Requirement Processes and Standards

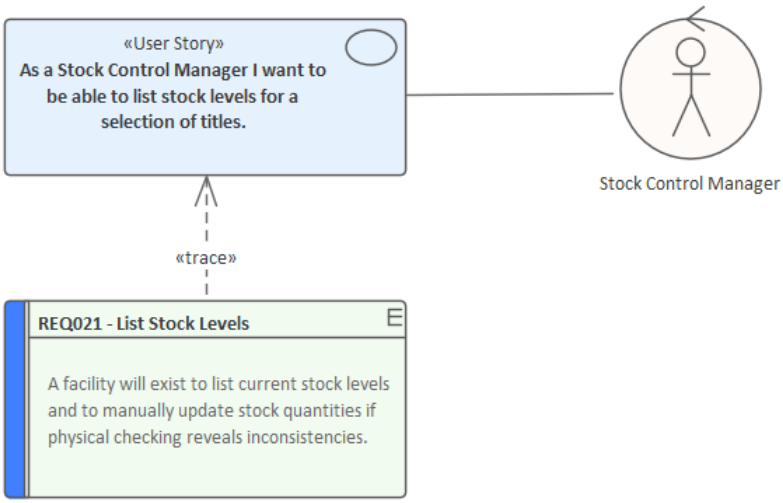
Enterprise Architect is an open platform that supports any Requirement Engineering process. The tool has a rich feature set and is highly configurable, and its flexible design means that whatever method is being used, the modeler will find features to help. So whether the team is using Formal Requirements, Use Cases, User Stories or Story-Boards in any combination, Enterprise Architect can be used to develop, manage and document the requirements. The implementation of the UML extension mechanisms means that any type of requirement can be created and managed using built-in types or by using Stereotyped elements and Tagged Values.

User Stories

User Stories are useful as an alternative way of describing user Requirements. They are typically used as part of an Agile development process, to provide a simple but clear description of what the user does or needs to do as part of the role they perform.

A User Story can be created using the stereotyped Artifact available from the Artifact Toolbox page, or as a stereotyped Use Case.

This diagram shows how a User Story can be modeled using a stereotyped Use Case. This allows the User Story to be described and to show the connection to a Persona.

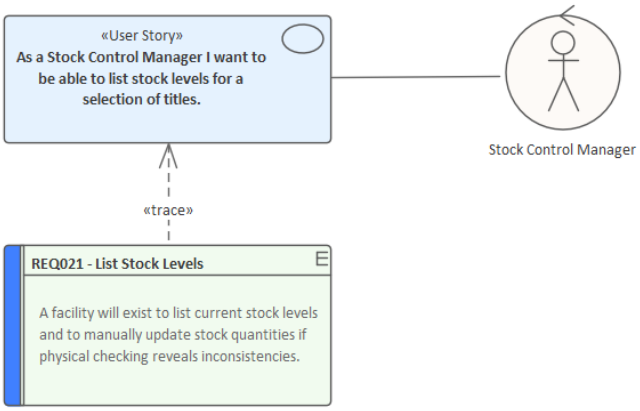


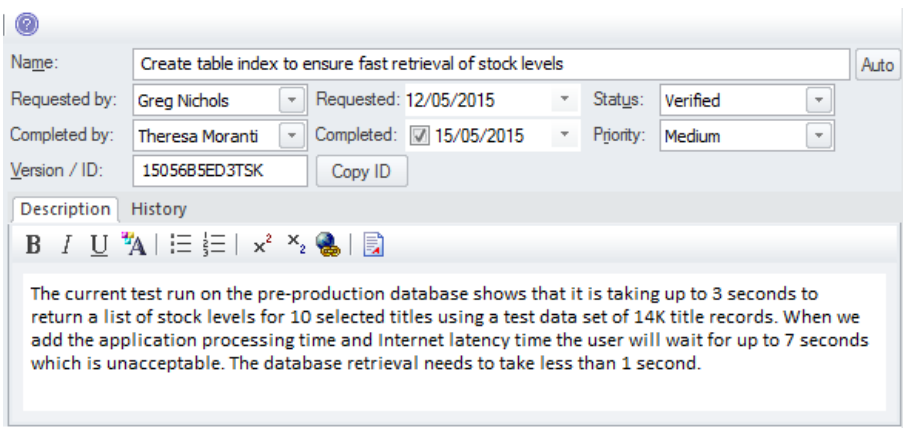
Agile Requirements Processes

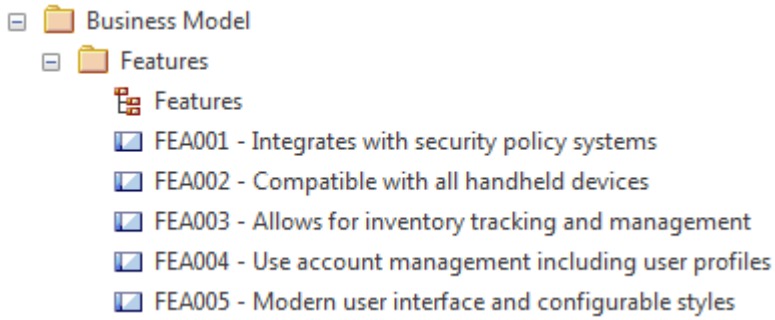
There are various Agile methods that have become prevalent for developing particular types of system, and the term 'Agile' has come to encompass a group of software development methods that are iterative and that focus on early development and delivery using customer and developer collaboration. Enterprise Architect has been built from the ground up as a flexible modeling platform that supports any software development methodology, and it has a wide range of tools and features that support Agile processes.

Support for Agile Methods

Method	Support
Scrum	'Scrum' is a project management approach for managing Agile projects; it is typically used in conjunction with other agile methods such as XP. The goal of Scrum is to deliver software releases in iterations, which provide the highest demonstrable business value. Enterprise Architect supports User Stories, textual requirements and product releases. Sprint backlogs can be managed using Tagged Values and Model Views. Estimates of

	effort can be recorded for Requirements in each of the backlogs, and refined as the items are promoted towards the sprint using the element Effort item. A built-in Gantt chart will automatically display the schedule for sprints, and a series of built-in and customizable dashboards can show the progress of a sprint. The Product Owner, Scrum Master and Team Member roles can all be supported. The tool provides a cohesive platform for collaboration and requirements management. **User Stories** User Stories are useful as an alternative way of describing user Requirements. They are typically used as part of an Agile development process, to provide a simple but clear description of what the user does or needs to do as part of the role they perform. A User Story can be created using the stereotyped Artifact available from the Artifact Toolbox page, or as a stereotyped Use Case. This diagram shows how a User Story can be modeled using a stereotyped Use Case. This allows the User Story to be described and to show the connection to a Persona. 
Extreme	Extreme Programming relies on the User

Programming (XP)	Story (User Card) to express Requirements and develops plans for iterations and releases. Enterprise Architect supports XP by allowing User Stories to be modeled using a stereotyped Use Cases. Iterations are supported by the Phase property built into every element, and a Gantt chart can be used to automatically display scheduled iterations and releases. There is support for developer tasks that can be created as element maintenance tasks for each User Story; these can include status, priority, requested and completion dates, history and more. The Customer, Coach, Programmer and Tracker roles can all be supported and the tool provides a cohesive platform for collaboration, including visual inspections of code and design and automatic documentation generation. 

Feature Driven Development (FDD)	Feature-driven development (FDD) is an iterative and incremental process that uses a feature to drive iterations and development. Domain models are created early in collaboration with subject matter experts, subject areas are defined and these are broken down into Feature Sets and these in turn into atomic features. A Feature is a piece of system functionality that is valued by the client and is used to drive design, development and implementation. During design a Feature or small Feature Set are assigned to Classes and to programmers and the Features are used to monitor progress within an iteration. Enterprise Architect has support for this method with the Feature element that is part of the Requirements Toolbox, Subject Areas and Feature Sets can be modeled as Packages, allowing a hierarchy to be created that can be displayed as a diagram, list, using the Relationship Matrix or in the Traceability window. 

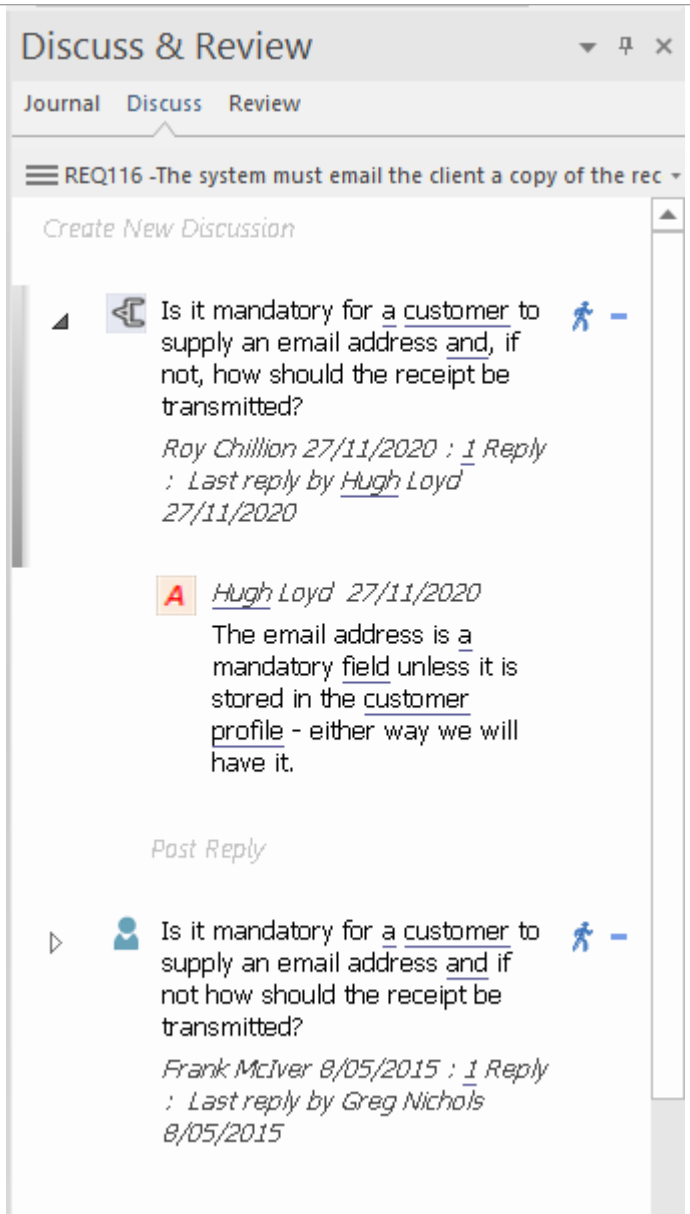
Business Analysis Body of Knowledge (BABOK)

The International Institute of Business Analysis publishes the Business Analysis Body of Knowledge (BABOK) Guide, which contains descriptive and prescriptive information on how to perform the activities and tasks of a business analyst in a number of areas of knowledge. The guide is produced and reviewed by a large number of people around the world and acts as a guide for business analysis, including requirements engineering. The guide defines a number of types of requirement, and these can all be implemented in Enterprise Architect by defining the types in the 'Requirements' section of the 'General Types' dialog.

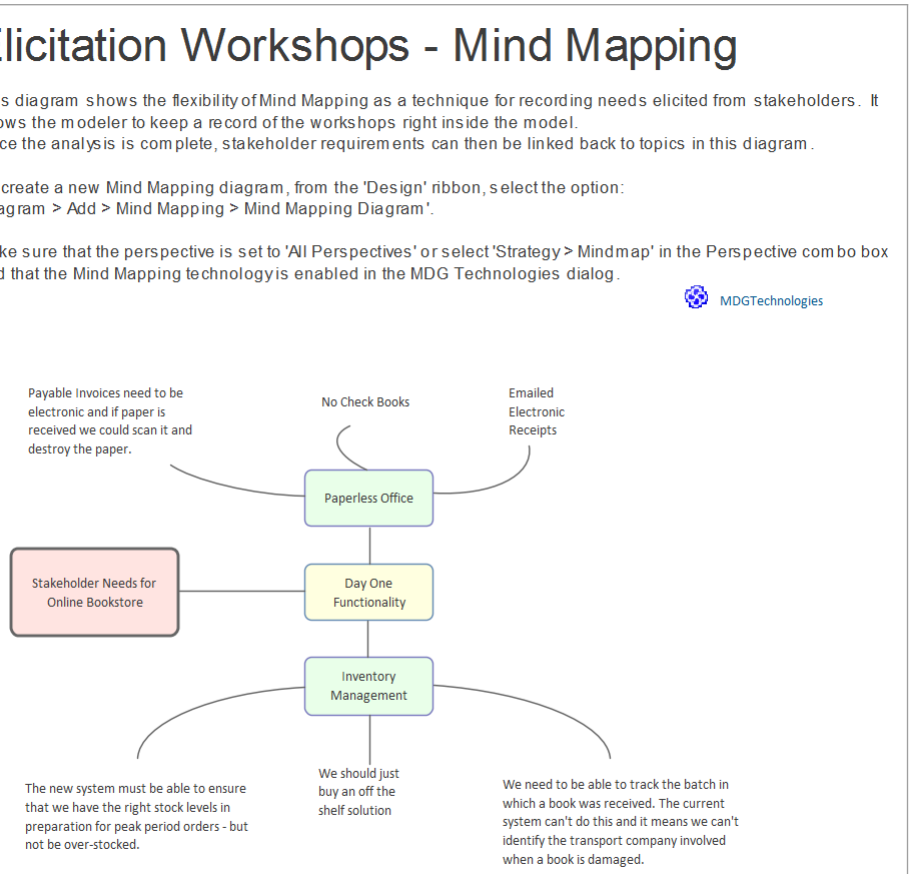
Support for BABOK Knowledge Areas

Knowledge Area	Description
Business Analysis Planning and Monitoring	The Business Analysis Planning and Monitoring knowledge area is concerned with planning the approach to the analysis effort, the engagement with stakeholder, the governance of the activities and the how the Requirements and other information discovered during

	the process is managed and maintained. Enterprise Architect allows a team of analysts to define the approach that will be taken in the tool by defining processes, stakeholders and the governance mechanisms that will be used including the information that is collected.
Elicitation and Collaboration	The Elicitation and Collaboration knowledge area is concerned with the discovery and conformation of information obtained from stakeholders and a variety of other sources. The collaboration with stakeholders during the entire requirements life cycle is critical to the success of the change or endeavor. Enterprise Architect has a number of tools that can assist with planning the elicitation, including a Project Calendar, Gantt chart and a number of team collaboration tools such as Model Mail, Discussions and Slide Shows.



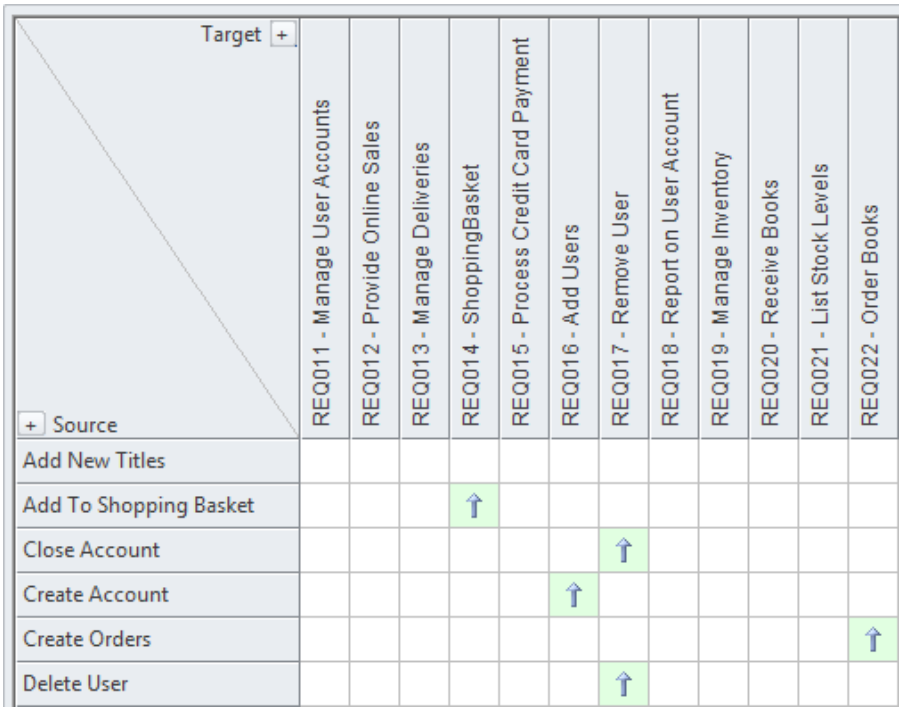
Information elicited during stakeholder workshops can be recorded using a Mind Mapping diagram and, once it has been analyzed, reports can be generated to a variety of formats for confirmation with stakeholders. Photos taken during user observations can be included in diagrams, creating a rich visual presentation.



Requirements Life Cycle Management

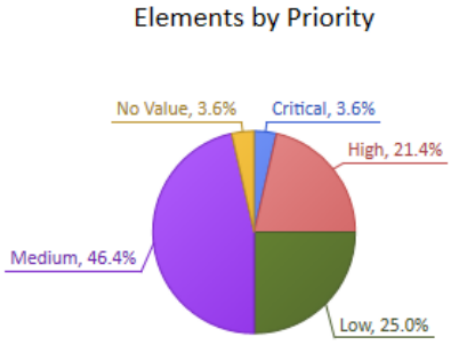
The Requirements Life Cycle Management knowledge area is concerned with the management of requirements and ensuring that the requirements can be related to the solution designs and, ultimately, to the solution components during the entire life cycle of the solution, from inception through to retirement. The management of requirements involves requirements governance, tracing the requirements, maintaining and prioritizing them, and managing change and assessment. Enterprise Architect has sophisticated

tools that enable requirements to be traced, maintained and prioritized. These include the Relationships Matrix, which can be configured to display sets of traces between source and target Packages; the Traceability window, which shows how elements are connected in the repository; and the Relationships window, which displays the relationships between elements in a diagram. The diagram is a convenient way to demonstrate and make visual underlying relationships, including the connection between stakeholders and the elicited information.

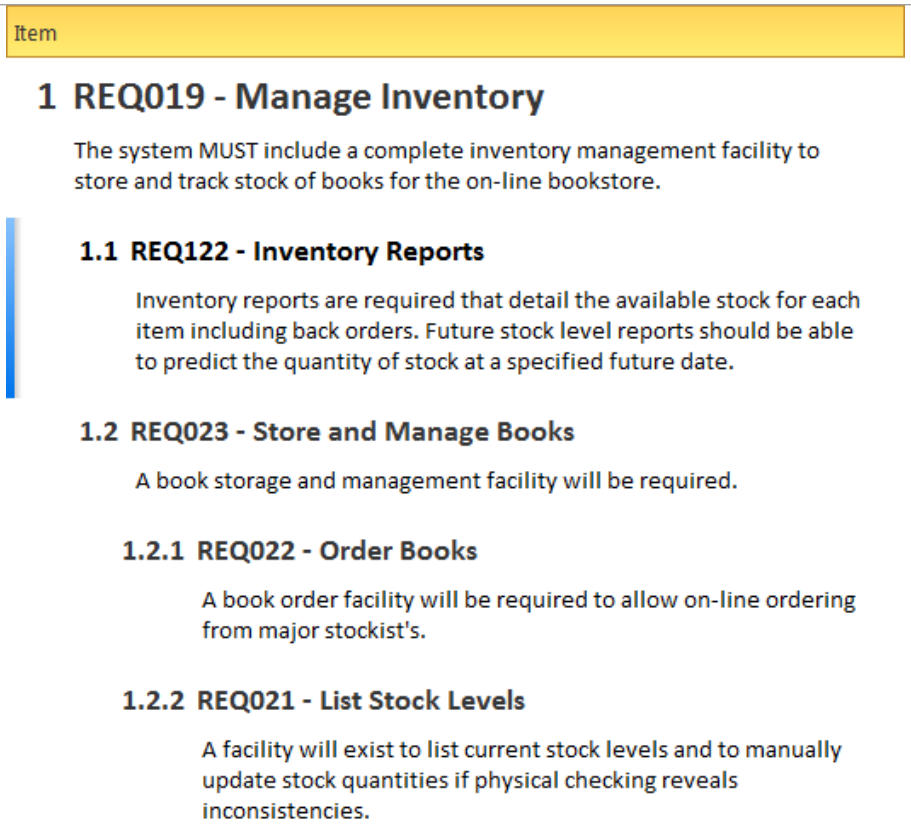


Source \ Target	REQ011 - Manage User Accounts	REQ012 - Provide Online Sales	REQ013 - Manage Deliveries	REQ014 - ShoppingBasket	REQ015 - Process Credit Card Payment	REQ016 - Add Users	REQ017 - Remove User	REQ018 - Report on User Account	REQ019 - Manage Inventory	REQ020 - Receive Books	REQ021 - List Stock Levels	REQ022 - Order Books
Add New Titles												
Add To Shopping Basket				↑								
Close Account							↑					
Create Account						↑						
Create Orders												↑
Delete User							↑					

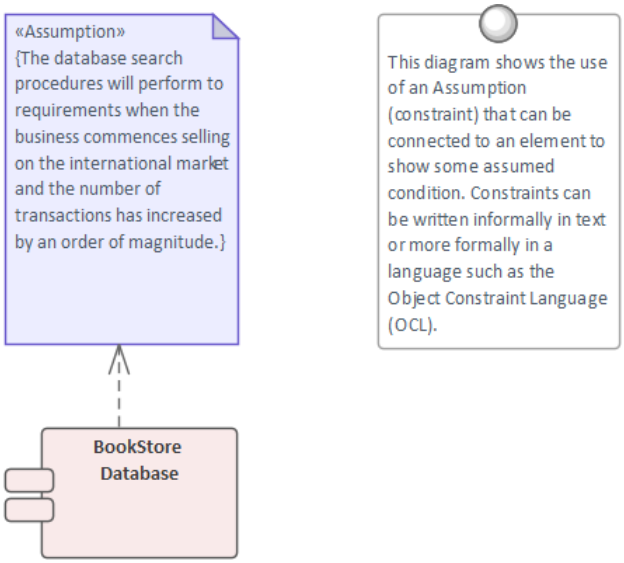
The requirements governance process could also be modeled using UML Activity diagrams or BPMN diagrams, and this process could be reused across a

	range of projects and changes. There is also a Dashboard diagram that allows Requirements properties to be categorized and displayed in a series of charts and graphs that assist in the collaboration and management of properties such as Status and Priority, and to communicate this information to stakeholders.  This diagram shows a Pie Chart element depicting element priorities for all the Requirements in a selected Package. It provides a useful summary for a Requirements Manager and is dynamically updated when the priority changes and the diagram is reopened. A range of other pre-defined Charts and user-defined Charts can also be added. A filter has been added to exclude all elements other than Requirements.
Strategy Analysis	The Strategy Analysis knowledge area is concerned with the discovery and articulation of a business need and the description of the future state and various transition (intermediate) states that will be used to move from the current state to the defined future state. It involves strategic thinking and the determination of a number of possible solutions that will create value for the organization and its stakeholders. Enterprise Architect has tools to describe the possible solutions

	and the value and outcomes they will produce and to model the current and future states and what is needed to transition from one state to another.
Requirements Analysis and Design Definition	The Requirements Analysis and Design Definition knowledge area is concerned with how to organize and structure the information that is discovered during elicitation and how to record these as a set of coherent Requirements. It is also concerned with the prioritization of the requirements and finding solution options that can be evaluated against the potential benefit they will bring to the business. The articulation and evaluation of solution options is done in conjunction with the implementation teams but ultimately it is the business analyst that is responsible for conveying the options and their business value and recommending a solution to the stakeholders. Enterprise Architect is a fully featured requirements management tool and allows requirements to be created, maintained and prioritized. The Specification Manager can be used to input, maintain and visualize requirements in a view that resembles working in a word processor or spreadsheet.

	
Solution Evaluation	The Solution Evaluation knowledge area is concerned with the evaluation of an implemented solution and the assessment of whether it meets the business needs. The implementation is typically a partial solution or proof of concept and the analyst must work to determine if there are obstacles or removable constraints that are limiting the value that can be delivered by the solution. Enterprise Architect is a full life cycle tool that not only is a requirements management platform but also allows the modeler to describe the solution in detail including constraints and design decisions and

rationales.

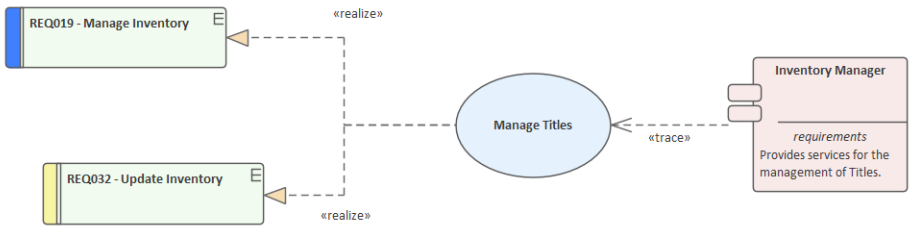


The solution components can be traced back to the functional, stakeholder and ultimately business requirements allowing the analyst to visualize the solution in the context of the problem and to record issues and decisions that will help inform how to proceed with a change.

Tracing Requirements

This diagram shows the expressive power of putting disparate elements onto a diagram.

It shows the traceability between different layers of a system. The traceability can be from the Requirements to the Use Cases that Realize them, to the logical Components that will deliver the required functionality.

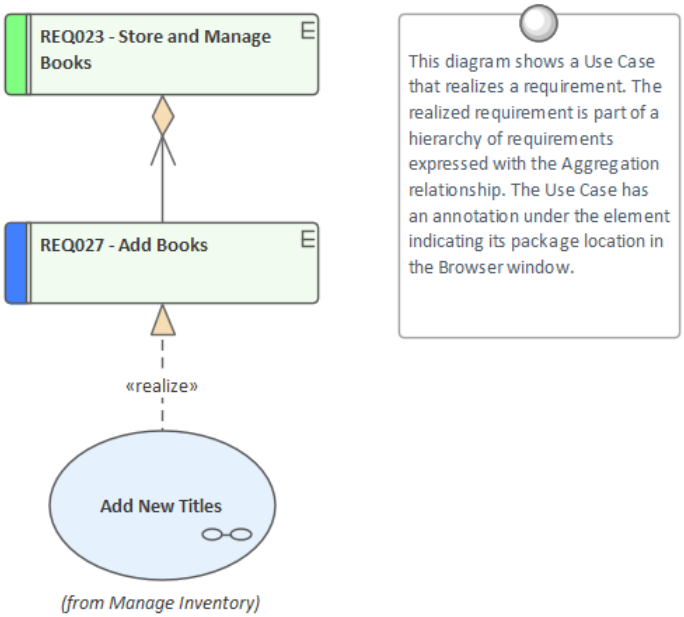


Notes

- Support for BABOK in Enterprise Architect includes a BABOK-specific Glossary for the product

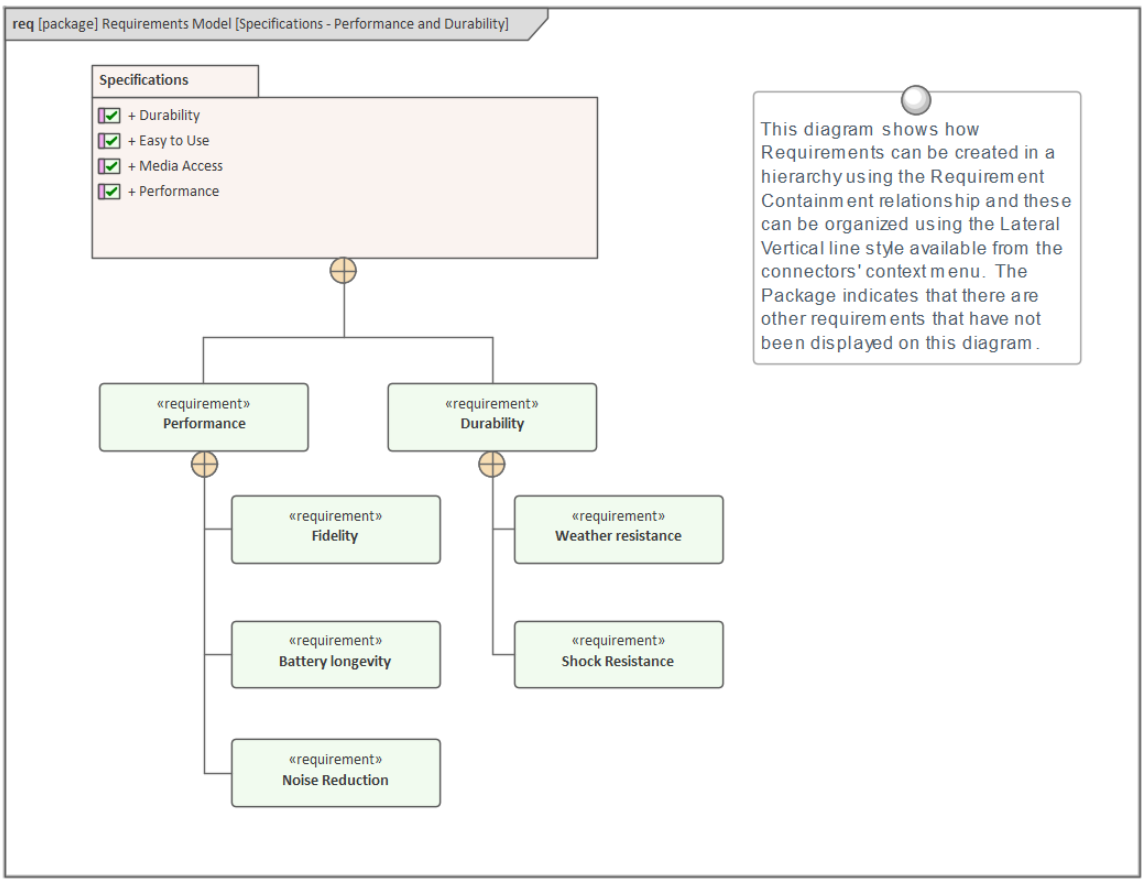
UML Requirements

The Unified Modeling Language does not specify an element for modeling Requirements other than the Use Case, which it states can be used to describe system usages. In very early versions of UML a Requirement was defined as a stereotyped comment, but this was later made obsolete. Enterprise Architect has, since very early versions, filled this gap in the UML specification by extending the language to include a Requirement element that allows this important concept to be modeled textually and graphically in diagrams and other views. Enterprise Architect has rich support for modeling Use Cases in compliance with the specification but allows the requirements analyst to specify other Requirements using the flexible Requirements element. When used together the Requirement element and the Use Case equip the analyst with a useful palette to describe the Requirements for any system at any level. The Requirement element can be used to describe Business Drivers, Business Goals and Stakeholder Requirements, and these can be linked to Use Cases with the Realize connector. The Use Cases can be augmented with detailed Functional, Architectural and Implementation Requirements suitable for the implementation teams.


REQ023 - Store and Manage Books
REQ027 - Add Books
«realize»
Add New Titles
(from Manage Inventory)
This diagram shows a Use Case that realizes a requirement. The realized requirement is part of a hierarchy of requirements expressed with the Aggregation relationship. The Use Case has an annotation under the element indicating its package location in the Browser window.

SysML Requirements

The Systems Modeling Language (SysML) specifies a rich set of modeling constructs to represent text-based requirements as elements that can be related to other elements in a model. Model Based Engineering has become important in recent years, as the complexity of systems has increased and the engineering profession has recognized the need for more sophisticated mechanisms for describing and developing these complex systems. Enterprise Architect has full support for SysML and the modeling of Requirements, including support for a range of ways of presenting Requirements such as: Diagrams, Lists, Matrices, Hierarchies and documentation. There are even dashboard diagrams that present charts and graphs that summarize requirements' properties such as Status, Priority and Difficulty in a visually compelling way.


req [package] Requirements Model [Specifications - Performance and Durability]
Specifications
+ Durability
+ Easy to Use
+ Media Access
+ Performance
This diagram shows how Requirements can be created in a hierarchy using the Requirement Containment relationship and these can be organized using the Lateral Vertical line style available from the connectors' context menu. The Package indicates that there are other requirements that have not been displayed on this diagram.
«requirement» Performance
«requirement» Durability
«requirement» Fidelity
«requirement» Battery longevity
«requirement» Noise Reduction
«requirement» Weather resistance
«requirement» Shock Resistance

Additional Requirement Tools

In addition to the key tools listed in *Meet the Requirements Tools* there are a number of other tools that can be used for Requirement Development and Management and team collaboration in the Requirement Engineering discipline. These include the ability to assign sequential numbers to Requirements, importing Requirements from a spreadsheet file, creating Requirement documentation, and team collaboration features such as Discussions.

More Requirement Tools

Tool	Overview
Auto-Names and Counters	Use to assign a sequential number to a Requirement, including a prefix and suffix.
Requirements Checklist	Provides a graphical list of checks that can be applied to individual Requirements.
Import and Export Spreadsheets	A tool to import and export Requirements from spreadsheet files in the CSV format.
Documentati	

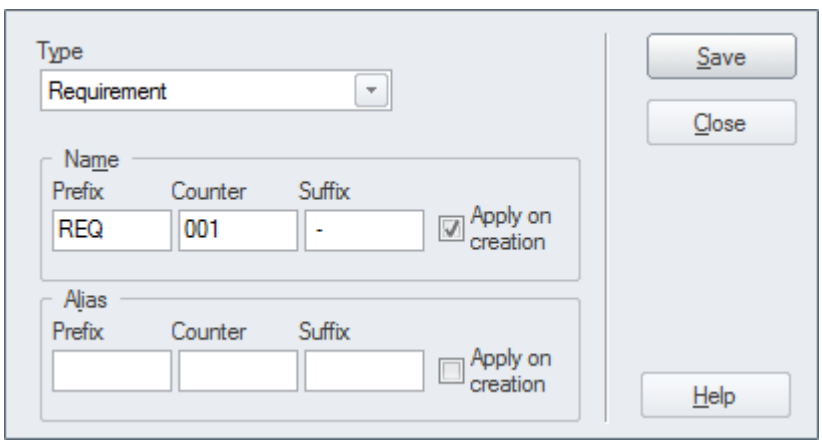
on	A useful engine to automatically create high quality documentation directly from the model, using built-in or user-defined templates.
Glossary	A feature to create and maintain a lexicon of terms and their meaning that can grouped by type and styled when included in documentation.
Auditing	Used to keep a trail of what has changed in a repository, who it was changed by and when.
Discussions	A facility to allow modelers to create posts and replies to discuss model elements.
Maintenance Items	A series of Items that can be applied to Requirements to define such things as Changes, Issues, Defects and more.
Model Library	Allows reviews to be created for user defined categories and topics with links to model elements such as Requirements and Scenarios that can be referenced in the review.

Auto-Names and Counters

Getting to know Auto Names and Counters

Introducing Auto Names and Counters

To aid, regulate and enforce a naming standard, Enterprise Architect includes some capabilities to configure the default names assigned to new elements of a specific type. This is a useful feature when dealing with complex and large sets of requirements, but is also relevant when dealing with smaller data sets. Auto Names and Counters can be used to assign a sequential number to any element type including Requirements. It includes a prefix definition, a counter and a suffix definition allowing numbers such as: 'REQ007 - Manage Inventory' to be created.



Where to find Auto Names and Counters

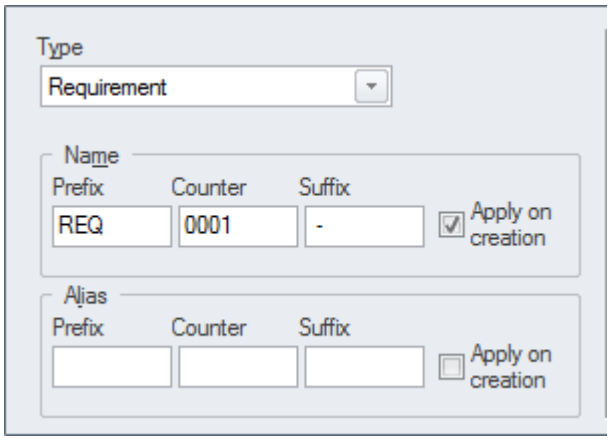
Ribbon: Settings > Reference Data > Settings > Auto Names and Counters

Usage of Auto Names and Counters

Analysts and others can use the sequential number for communicating unambiguously about the requirements without having to use the often long requirement name. Select the 'Apply on Creation' option to start using the auto numbering feature; this can also be used to temporarily suspend auto naming, for example if other types of requirements are being entered that don't need to have sequential numbers assigned.

Options for Auto Names and Counters

There are options to define the prefix, counter and suffix for a requirement.



Learn more about Auto Names and

Apply Auto Naming to Existing Elements

Counters

Import and Export Spreadsheets

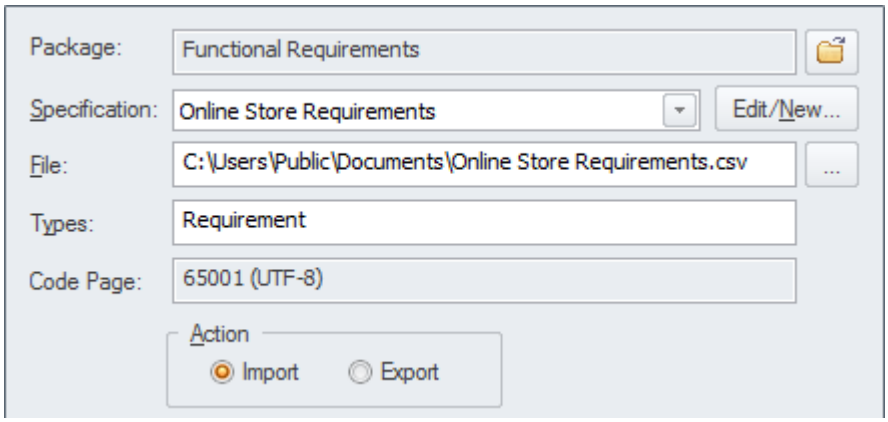
Import and Export Spreadsheets

Introducing Import and Export Spreadsheets

This facility is a useful mechanism to import Requirements that have been defined in a Spreadsheet or a Word Processor table into Enterprise Architect. Once in Enterprise Architect the Requirements can be managed and traced to elements such as business drivers and Scenarios and Components. Alternatively Requirements in Enterprise Architect can be exported to a Spreadsheet for the purposes of providing them to a third party or for some type of numerical or statistical analysis. The mapping between fields in the Spreadsheet and the analogous properties in Enterprise Architect is completely configurable using a specification.

For more detailed information exchange, the MDG Link for Microsoft Office (available from Sparx Systems) provides additional functionality and integration points useful when dealing with complex

Requirements.



Where to find Import and Export Spreadsheets

Ribbon: Publish > Model Exchange > CSV

Use of Import and Export Spreadsheets

This feature can be used to import or export Requirements from a CSV file. Before a tool such as Enterprise Architect was installed, Analysts might have used a Spreadsheet or a table in their favorite word processor to record Requirements; these can conveniently be imported using the CSV import facility. Alternatively, Requirements sometimes have to be provided to a third party who will typically specify that they want them in a Spreadsheet file; this can be achieved using the export facility.

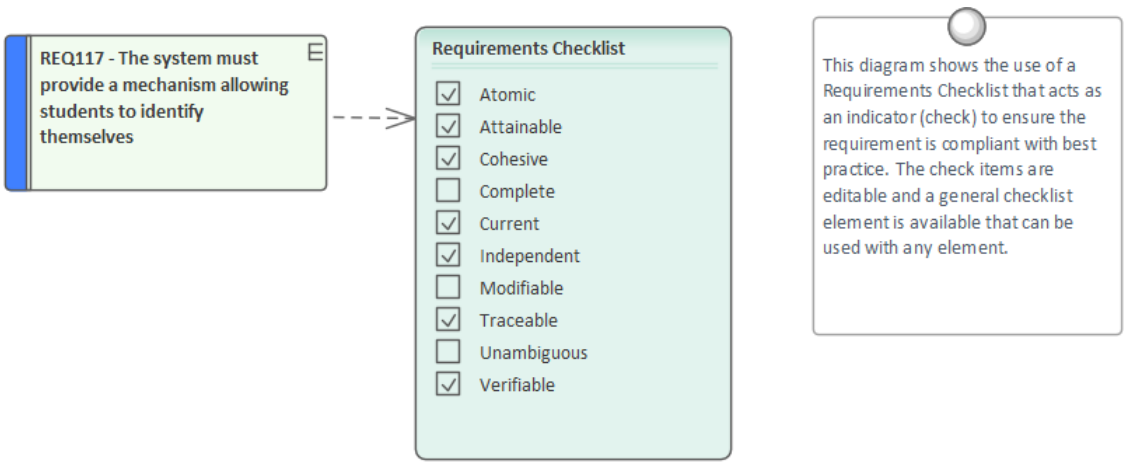
Options to Import and

The import and export facility is completely configurable and has a

Export Spreadsheets	user-defined specification to facilitate the mapping of Spreadsheet fields to Requirements properties in Enterprise Architect. This facility also includes the ability to import and export fields in Tagged Values of the Requirement.
Learn more about Import and Export Spreadsheets	CSV Import and Export

Requirements Checklist

The Requirement Checklist is a convenient element that acts as a tally to indicate whether a Requirement complies with a set of predefined measures such as whether the Requirement is Atomic, Cohesive, Traceable and Verifiable. It can be assigned to any Requirement and the measures can be updated directly in the diagram. When working with Requirements it is sometimes very useful to refer to a common set of 'best practices' and qualities that help define the nature of a well formed specification. The Requirement Checklist element is designed to meet this need.



Getting to know the Requirements Checklist

Where to find the Requirements Checklist	Toolbox : to display the 'Find Toolbox Item' dialog and specify 'Requirements Checklist'

Usage of the Requirements Checklist

Analysts and Requirements Managers can use the checklist to annotate whether one or more elements such as a Block or Activity or even a set of Requirements meet a set of predefined checks.

Options for the Requirements Checklist

The list of measures is completely configurable and items can be added or removed from the list for each individual checklist by using the Checklist Tagged Value notes.

```
<Checklist>
        <Item Text="Atomic" Checked="True"/>
        <Item Text="Attainable" Checked="True"/>
        <Item Text="Cohesive" Checked="False"/>
        <Item Text="Complete" Checked="False"/>
        <Item Text="Current" Checked="True"/>
        <Item Text="Independent" Checked="False"/>
        <Item Text="Modifiable" Checked="True"/>
        <Item Text="Traceable" Checked="True"/>
        <Item Text="Unambiguous" Checked="True"/>
        <Item Text="Verifiable" Checked="True"/>
</Checklist>
```

Learn more about the Requirements Checklist

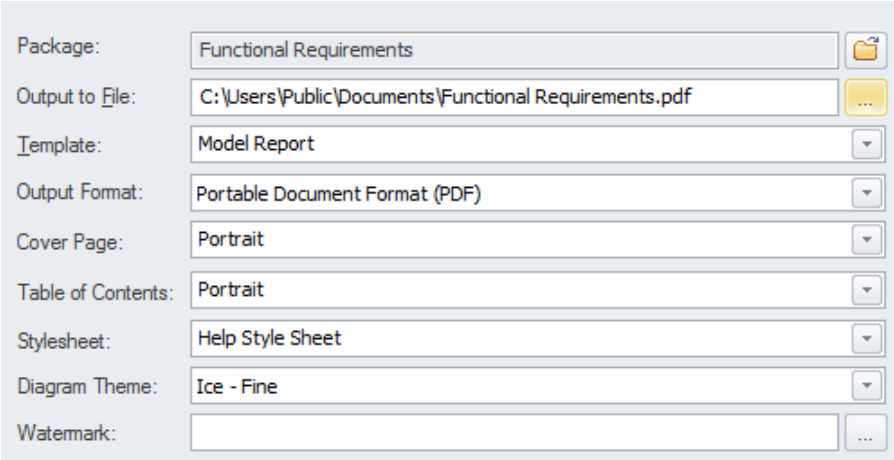
Using the Checklist and Audited Checklist Artifacts

Documentation

Getting to know Documentation

Introducing Documentation

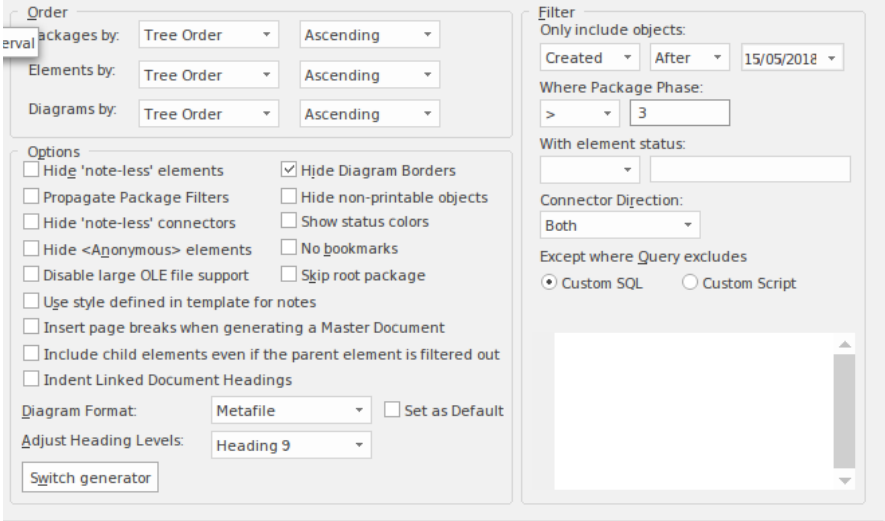
The Documentation features can be used to automatically generate a wide range of documentation directly from the models. These can be document-based such as PDF and Docx format or HTML-based. Flexible templates can be used to completely tailor the documents that are generated including company logos, tables of content, tables of element information and diagrams. Ad-hoc reports can also be created from a number of tools such as the Glossary and the Search Window.



Where to

Ribbon: Publish > Model Reports >

find Documentation	Report Builder
Use of Documentation	Modelers, Analysts, Architects, Project Managers and others can use the facility to produce a wide range of document-based publications and reports, such as a System Requirements Specification, Use Case Report, Data Dictionary, Solution Architecture Description and more. It can also be used for ad-hoc reporting to create reports such as a list of the most volatile requirements. HTML documentation can also be published to allow stakeholders who don't have access to Enterprise Architect to view the models from an Intranet site that can just be placed on a file system without the need for a Web Server.
Options for Documentation	There are several options that can be set to tailor the information that is included in a generated document, including the ordering of elements and diagrams and hiding certain elements. Filters and word substitutions and other options can also be applied.



Learn more about Documentation

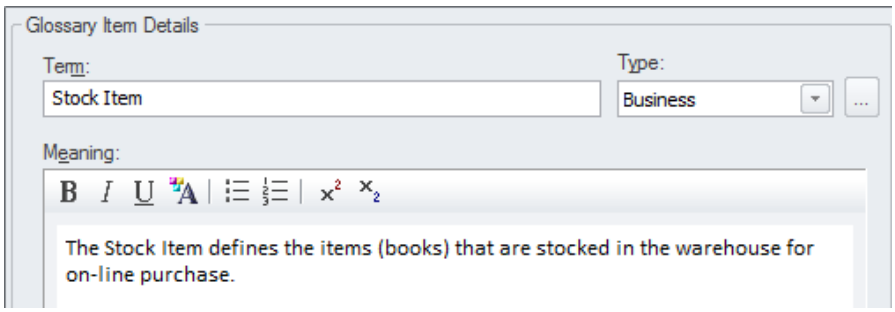
[Model Publishing]

Glossary

Getting to know the Glossary

Introducing the Glossary

The Glossary is a project level lexicon of the important terms and their meanings categorized by type. Any number of terms, their types and meanings can be defined and these can be referenced from the notes of model elements. The terms can be included in documentation or generated as a stand-alone report. When working with domain specific requirement specifications, architectures and other models it is essential that new terms and over-ridden meanings for common words or phrases are kept in a suitable dictionary format to ensure proper understanding of documentation and specifications.



Where to find the Glossary

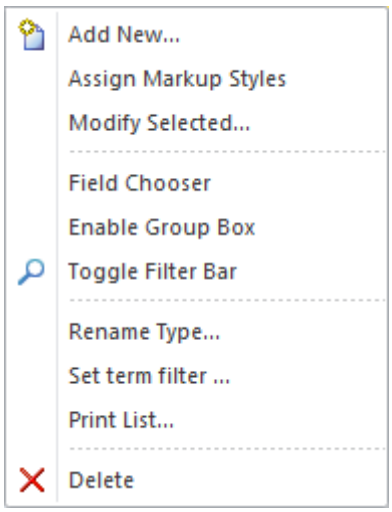
Ribbon: Publish > Dictionary > Glossary

Usage of the Glossary

The project Glossary can be used to record the important terms of a project or domain grouped by the type of term, allowing business, technical and domain specific types to be defined. A Glossary Report can be generated as a stand alone report or the glossary can be included as a section of another document.

Options for the Glossary

The Glossary has a number of options to determine the terms that are displayed in the list and to define the style that is used for types of terms in documentation.



Learn more about the Glossary

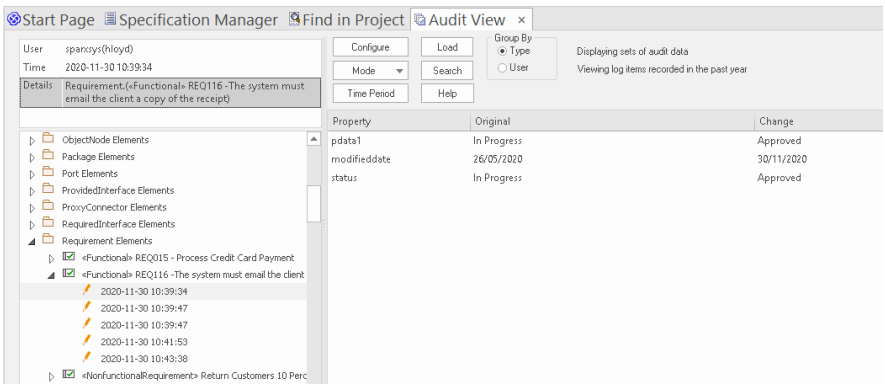
Model Glossary

Auditing

Getting to know Auditing

Introducing Auditing

The Auditing feature can keep track of the changes to Requirements including what was changed, when it was changed and by whom. Auditing is by default disabled and must be enabled before the changes to requirements will be recorded. Once enabled it is a passive tool that silently records the changes to elements. It does not replace Version Control or Baselines and in contrast to these tools it can not be used to return to a previous state of the model. Change management, governance and quality control are all aided by the use of Auditing.



Where to

Ribbon: Settings > Model > Auditing

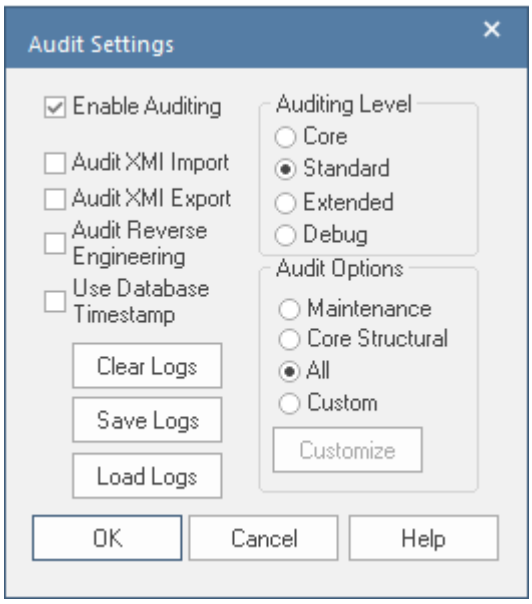
find Auditing

Use of Auditing

Auditing can be used to track what was changed in a model, who changed it and when. There are a number of modes and a repository administrator can use the settings to specify what is recorded in the audit. While a baseline can be used to show the difference between a model and a snapshot at a point in time, the Auditing tool records each individual change; it can not, however, be used to revert to a previous state.

Options for Auditing

There is a wide range of settings to configure auditing, starting with enabling or disabling the settings that determine which elements have an audit trail and the level of detail recorded. Audit logs can be exported from the repository to increase performance.



Learn more about Auditing

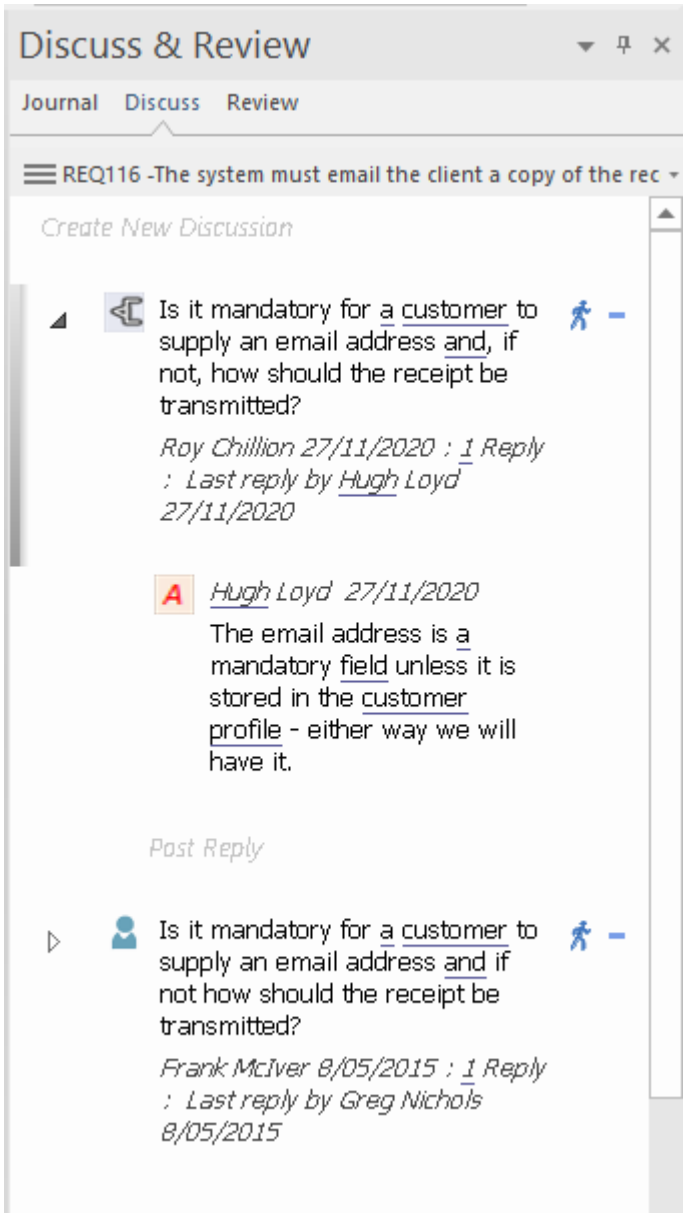
Auditing

Discussions

Getting to know Discussions

Introducing Discussions

The Discussions facility allows modelers to have conversations about elements, posting discussions and replying to existing posts. The Discussions for all elements in the model are conveniently listed in the Discussions Review window, allowing a modeler to see all the elements with posts.



Where to find Discussions

To post or view an element's discussion

Ribbon: Start > Collaborate > Discuss > Discuss

To view recently discussed elements

Ribbon: Start > Collaborate > Discuss > Recently Discussed

To view all discussions in the repository

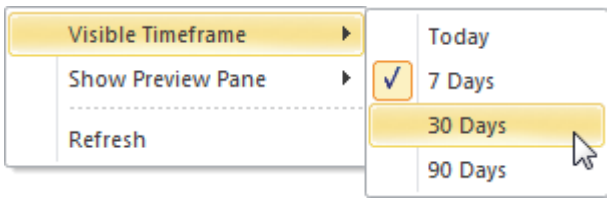
Ribbon: Start > Collaborate > Discuss > Discussion History

Usage of Discussions

Discussions allow modelers to have conversations about elements without 'polluting' the element's notes with questions or modeling level comments such as 'Properties need to be added before the first release'. This feature brings the collaborative modeling platform alive, where modelers can add informal discussions about elements, emulating the discussions held in a physical workshop.

Options for Discussions

The Discussion Review window has a number of options to tailor the Discussions that are listed, including defining the timeframe to allow older and perhaps less relevant Discussions to be hidden.



Learn more about Discussions

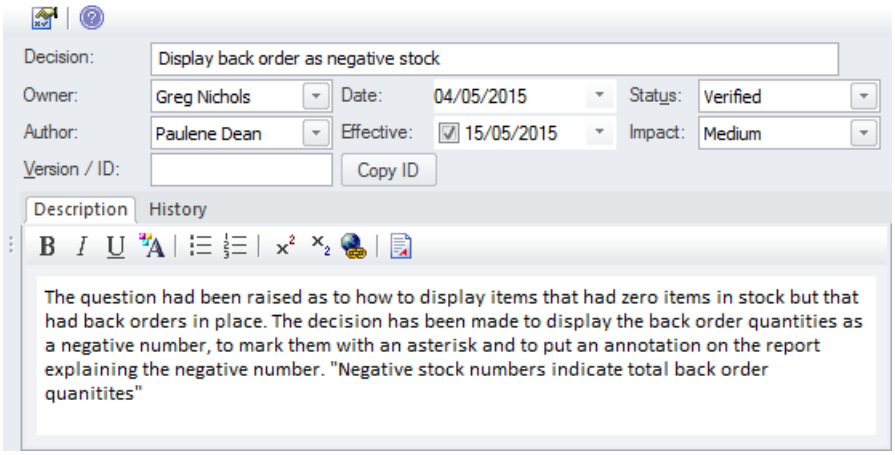
Model Discussions

Maintenance Items

Getting to know Maintenance Items

Introducing Maintenance Items

Element maintenance items can be used with any element, including Requirements, Components and Classes, to capture problems, changes, issues, tasks, events and decisions that affect the individual element. Several types of Maintenance Items can be used to track official changes, additions and deletions to sets of Requirements, Components, User Stories and other specifications, providing a sound basis for overall project governance and traceability.



Where to find Maintenance

Ribbon: Construct > Change Management > Features / Changes /

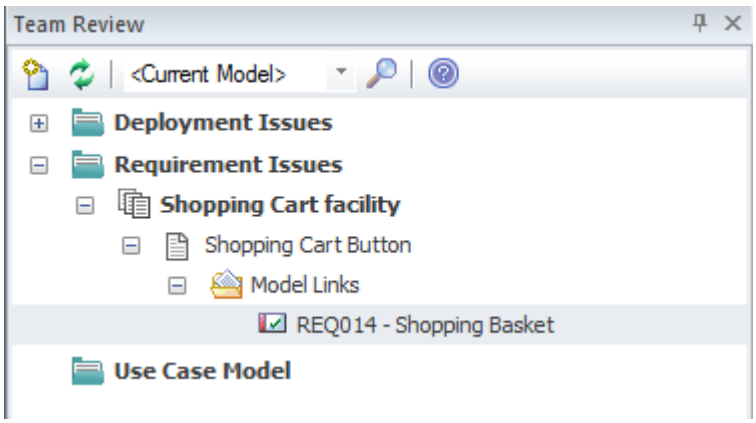
Items	Documents / Issues / Defects / Tasks
Usage of Maintenance Items	Element Maintenance Items can be used to record a wide range of information about elements including: changes, issues and tasks and more. These can be used to track the way an element is altered and maintained and can provide information about the Item including the rationale for the alteration, status and priority. While the Baseline and Auditing features record what has changed automatically the Maintenance Items provide the flexibility for a modeler to specify details manually.
Options for Maintenance Items	There are several types of maintenance item that can be used to add information to elements, including Features, Changes, Documents, Issues, Defects and Tasks. Items can present information such as who requested the item, who completed the item, and the status and priority of the item.
Learn more about Maintenance Items	Maintenance Items

Library

Getting to know The Library

Introducing The Library

The Library window provides an opportunity for developers, modelers, customers and stakeholders to comment and provide feedback on the work in progress or at the completion of a milestone or project.



Usage of The Library

The Library feature can be used to conduct model reviews from any number of perspectives, including walk-throughs, formal model reviews, or ad-hoc reviews.

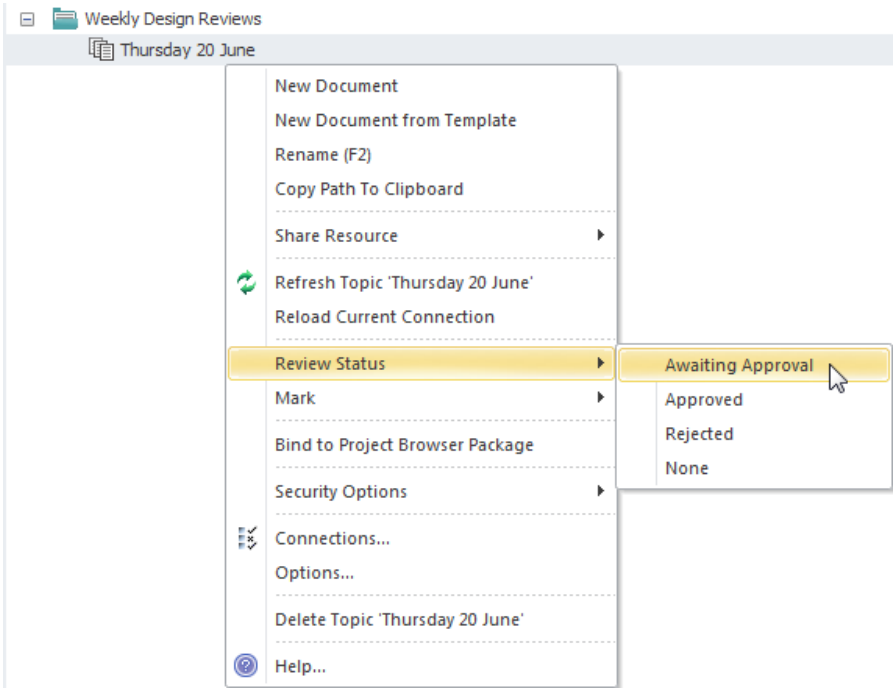
Where to find The Library

To post or view an element's discussion

Ribbon: Start > Collaborate > Model Library

Options for The Library

There is a wide range of settings available to configure the Library, available from the Category and Topic context menus, and including setting the status of the category or topic and other options. Diagrams, elements and element features can be conveniently dragged from the Browser window to create model links that can be used by team members to hyperlink directly from the Library window to these items in the Browser window.



Learn more about Model Library

The Model Library

MDG Link for DOORS

Welcome to the Model Driven Generator (MDG) Link™ for DOORS, which provides support for linking an Enterprise Architect model to an IBM® Rational® DOORS® Requirements package.

Features

The MDG Link for DOORS is useful when you need to perform the management of requirements external to your Model Driven Development. Using this Add-In you can interchange requirements defined within IBM DOORS with the traceable Requirements used within Enterprise Architect's Requirements Management features.

The MDG Link for DOORS supports:

Importing:

- Objects from IBM® Rational® DOORS®
- Object attributes from IBM® Rational® DOORS®
- Links between objects
- External links (attachments)
- Discussions

Exporting:

- Requirements and Use Cases
- Element properties and Tagged Values
- Connectors between elements

- Attachments

Synchronization:

- Of IBM® Rational® DOORS® requirements and Enterprise Architect elements through import and export (or vice-versa)

Obtaining the MDG Link for DOORS

For all Enterprise Architect Editions other than Ultimate, you can purchase the MDG Link for DOORS separately and download the installer from the Sparx Systems website. The product pages on the website provide:

- A product overview
- A video demonstration of the product
- Pricing and purchasing information
- System requirements for the product
- A trial edition of the product to explore for 30 days

When you purchase the product, you will receive download and installation instructions by email.

The Enterprise Architect Ultimate Edition includes a licence for the MDG Link for DOORS, although you download the Link installer from the Sparx Systems website and execute it separately.

Enable MDG Link for DOORS

When you have installed the MDG Link for DOORS, and before you can access its facilities, you must enable the product for use.

1. In Enterprise Architect, select the 'Specialize > Add-Ins > Manage-Addin' ribbon option.
2. Against the 'DoorsEASync' entry, select the 'Load on Startup' checkbox.
3. Click on the OK button.

Getting Started

After installing and enabling the MDG Link for DOORS, you can access the facilities of the technology in Enterprise Architect.

Access

Ribbon	Click on target Package, then: Specialize > Add-Ins > DOORS > [option]
Context Menu	Browser window | Right-click on target Package | Specialize | DOORS

Tasks

There are three principal tasks that you can perform through the MDG Link for DOORS.

Task	Description
Create a Module Link	On the 'DOORS' menu you can select the 'Add/Edit Module' option to create a link

	between Sparx Systems Enterprise Architect and an existing IBM® Rational® DOORS® module. You can also redirect a link to a different module. The link enables you to exchange requirements data between DOORS and Enterprise Architect, importing data from DOORS to Enterprise Architect, and exporting data from Enterprise Architect to DOORS.
Export Data To a DOORS Module	On the 'DOORS' menu you can select the 'Export to Doors' option to transfer requirements from a selected Enterprise Architect Package to a selected DOORS module. As part of this process you can create an export profile to define what the export should operate on.
Import Data From a DOORS Module	On the 'DOORS' menu you can select the 'Import from Doors' option to transfer requirements into a selected Enterprise Architect Package from a linked DOORS module. As part of this process you can create an import profile to define what the import should operate on.

Create a Link to a DOORS Module

In the MDG Link for DOORS you can create a link between Sparx Systems Enterprise Architect and an existing IBM® Rational® DOORS® module, which enables you to exchange requirements data between DOORS and Enterprise Architect. You can also redirect the link to a different module. Through this link you can import data from DOORS to Enterprise Architect, and export data from Enterprise Architect to DOORS.

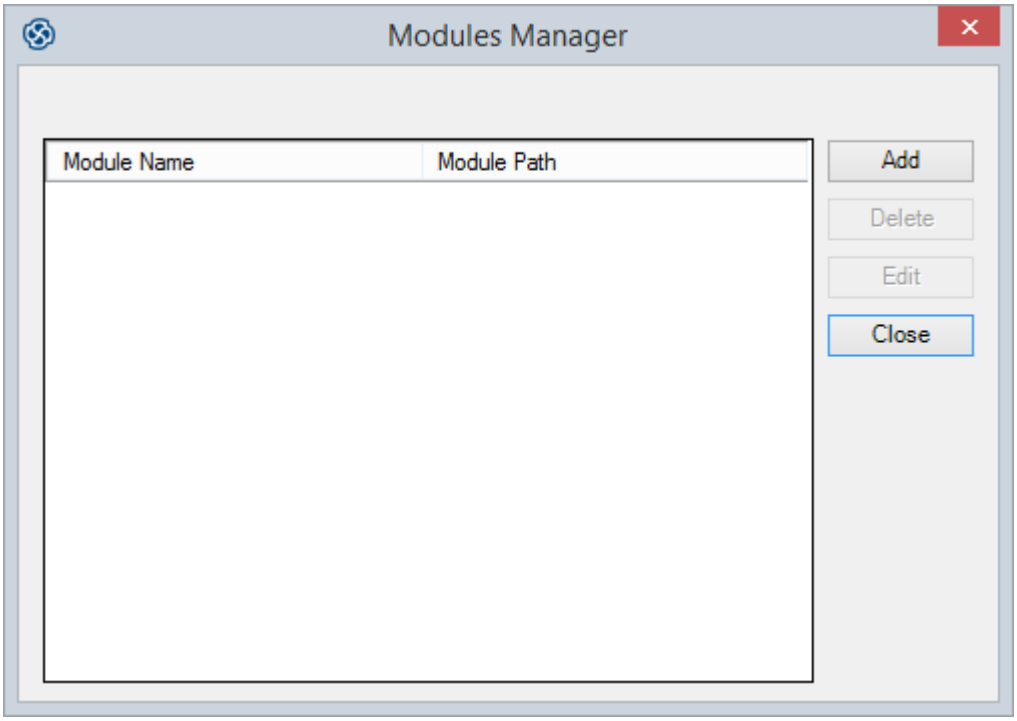
Access

Ribbon	Click on target Package, then: Specialize > Add-Ins > DOORS > Connect External Project
Context Menu	Browser window | Right-click on target Package | Specialize | DOORS | Add/Edit Module

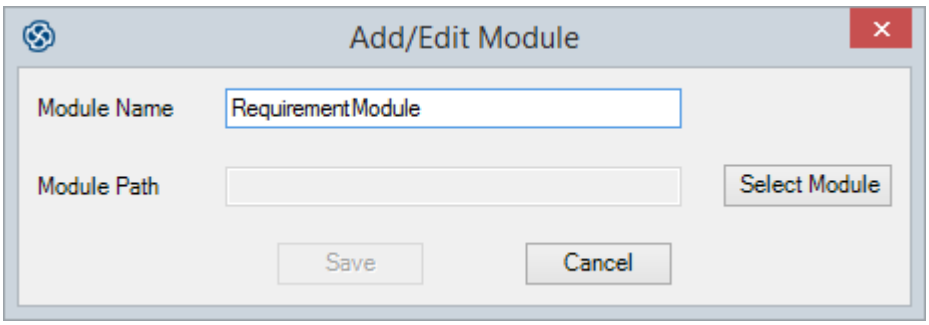
Modules Manager Dialog

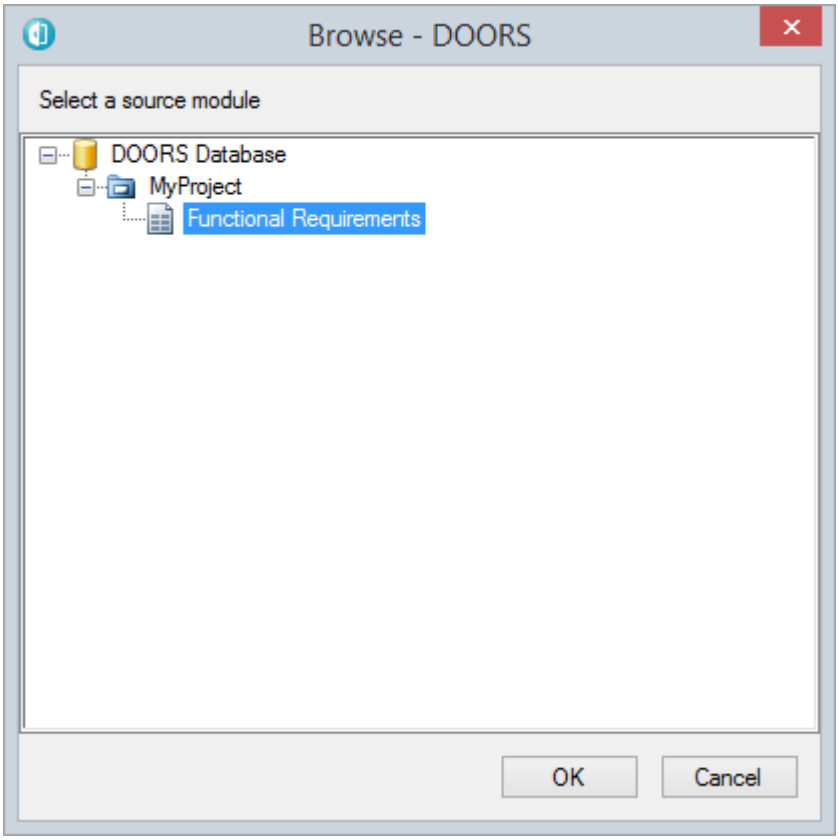
Use the 'Modules Manager' dialog to locate an existing

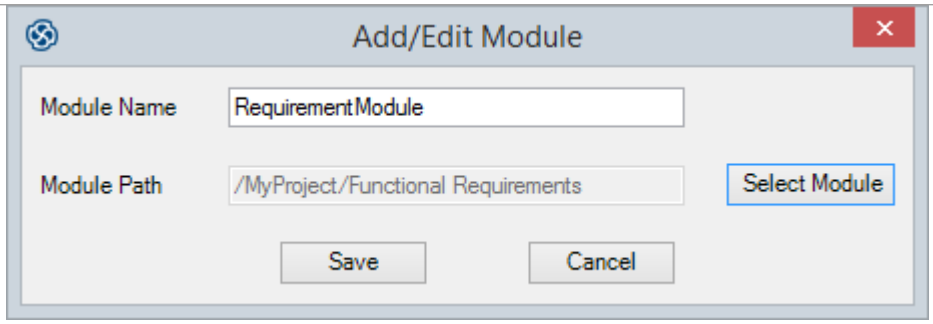
DOORS module and create a link to it.



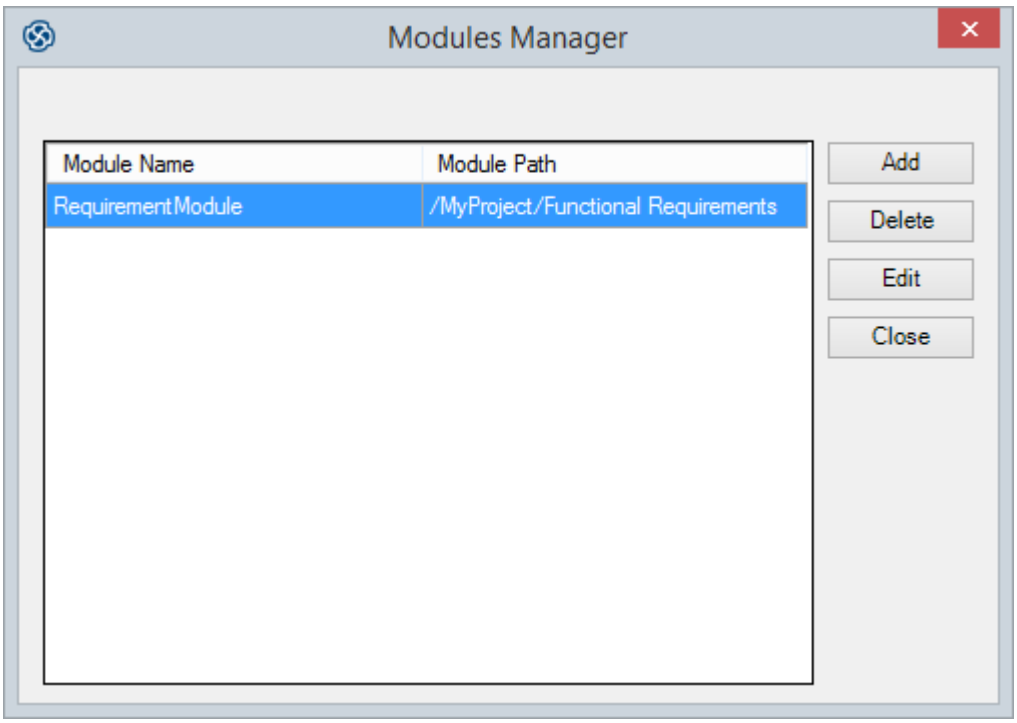
Create a Link to a DOORS Module

Step	Action
1	On the 'Modules Manager' dialog, click on the Add button. The 'Add/Edit Module' dialog displays.  The 'Module Name' field displays the name of the

	selected Module, to identify the link to the module. You cannot edit this name.
2	Click on the Select Module button to open the 'Browse - DOORS' dialog (in DOORS itself). 
3	Expand the module hierarchy as necessary to locate the required module, and click on it.
4	Click on the OK button to return to the 'Add/Edit Module' dialog, which now displays the module path and name in the 'Module Path' field.



6 Click on the Save button. The 'Module Manager' dialog redisplays, showing the newly-created module link. You can now use this link to exchange requirements data between Enterprise Architect and DOORS.



Click on the Close button to close the dialog.

Edit or Delete a Link to a DOORS Module

If the existing link to the DOORS module is not appropriate, you can delete or redirect it.

Step	Action
1	On the 'Modules Manager' dialog, click on the link to change. • To delete it, click on the Delete button and on the Close button; this ends the procedure • To redirect it, click on the Edit button; the 'Add/Edit Module' dialog redisplays
2	On the 'Add/Edit Module' dialog, click on the Select Module button to display the 'Browse - DOORS' dialog (in DOORS).
3	Expand the hierarchy as necessary on the 'Browse - DOORS' dialog, and click on the replacement module for the link.
4	Click on the OK button to return to the 'Add/Edit Module' dialog, which now shows the module path of the replacement module. Note that you cannot edit the link **name**.
5	Click on the Save button. The 'Module Manager' dialog redisplays, showing the edited module link. Click on the Close button to close the dialog.

Notes

- You can also create and edit links to DOORS modules through the 'Export to Doors' and 'Import from Doors' dialogs, using the Module Manager button

Export Requirements to DOORS

Using the Sparx Systems MDG Link for DOORS, you can transfer all the Requirement elements under the selected Enterprise Architect Package to a linked IBM® Rational® DOORS® module, as DOORS objects.

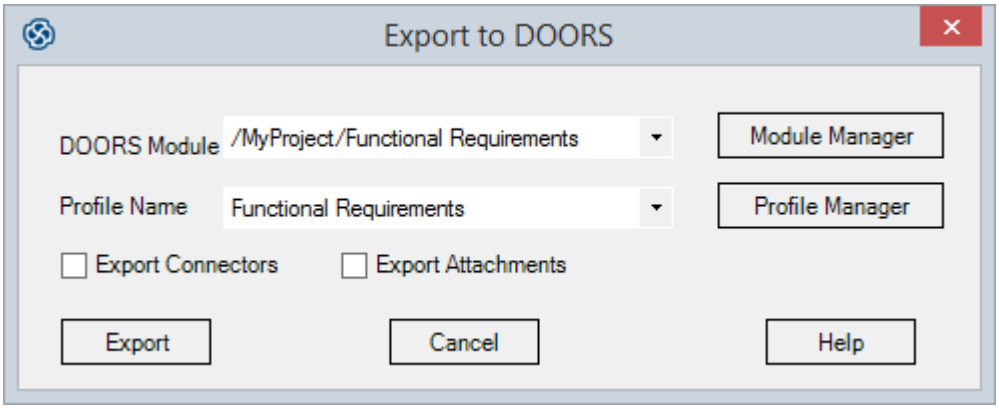
As part of this process you can create an export profile to define what the export should operate on. An export profile is a template in which you specify the Enterprise Architect element properties to be exported to the mapped fields in DOORS object properties.

Access

Ribbon	Click on target Package, then: Specialize > Add-Ins > DOORS > Export to Doors
Context Menu	Browser window | Right-click on target Package | Specialize | DOORS | Export to Doors

Export to DOORS Dialog

On the 'Export to DOORS' dialog you set the module you are exporting into, and specify which export profile to apply and whether to also export connectors and attachments.



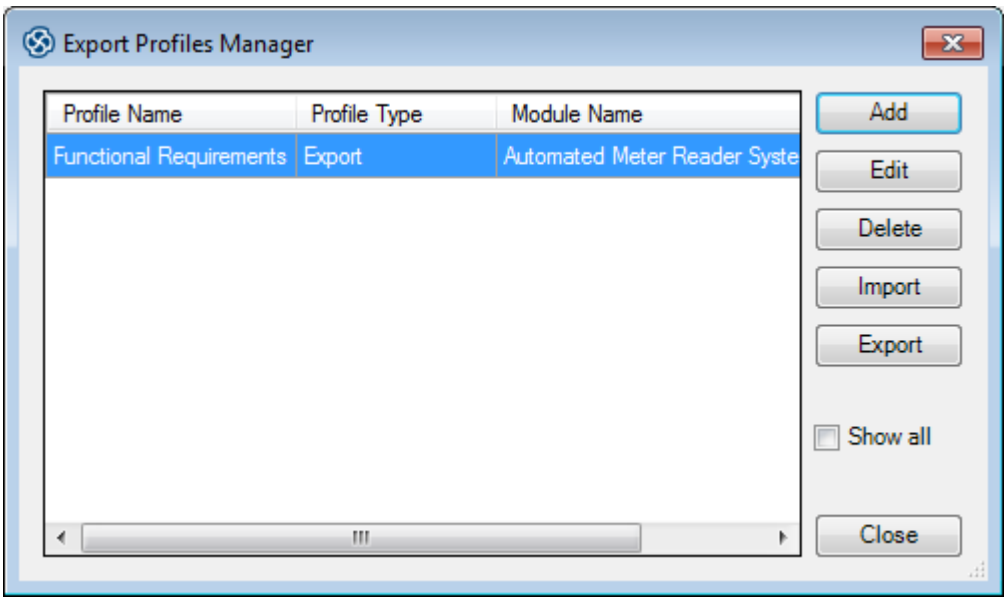
Export Options

Field/Button	Action
DOORS Module	Click on the drop-down arrow and select the linked DOORS module into which to export the Requirements.
Module Manager	If the required module is not listed, click on this button to display the 'Module Manager' dialog, and locate and link to the module.
Profile Name	Click on the drop-down arrow and select the appropriate export profile to use in

	this operation.
Profile Manager	If the export profiles listed are not appropriate, click on this button to display the 'Export Profiles Manager' dialog and create, edit or import the profile. (See the *Manage Export Profiles* table.)
Export Connectors	Select this checkbox to export any connectors between the Requirement elements.
Export Attachments	Select this checkbox to export any attachments that the Requirement elements might have.
Export	Click on this button to begin the export of Requirements from the selected Enterprise Architect Package to the specified DOORS module.
Cancel	Click on this button to close the 'Export to Doors' dialog without exporting any Requirements.
Help	Click on this button to display this Help topic.

Export Profiles Manager Dialog

When you click on the Profile Manager button on the 'Export to DOORS' dialog, the 'Export Profiles Manager' dialog displays. You use this dialog to create or import new profiles, edit or delete existing profiles, and export profiles to your preferred file system.



Manage Export Profiles

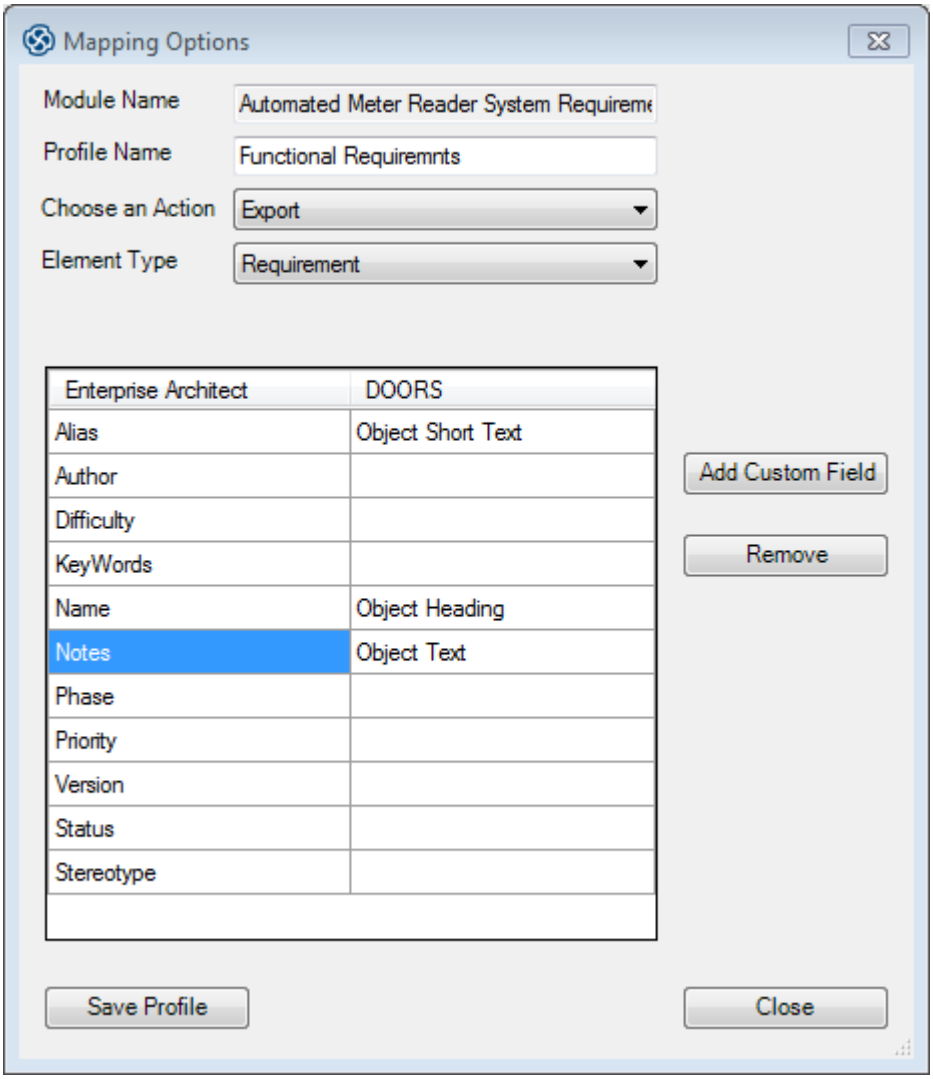
Field/Button	Action
Add	Click on this button to add a new export profile. The 'Mapping Options' dialog displays; see the *Create or Edit an Export*

	Profile table.
Edit	Click on an existing profile name and click on this button to edit the parameters of that profile. The 'Mapping Options' dialog displays, showing the current parameters of the profile; see the *Create or Edit an Export Profile* table.
Delete	Click on an existing profile name and click on this button to immediately remove that profile from the list.
Import	Click on this button to display the 'Import a Profile - Doors Extension' browser. Browse for the location of the required profile file (.eProfile), and click on the file name and on the Open button. A status message displays. Click on the OK button; if the operation was successful, the profile name displays on the 'Export Profiles Manager' dialog. If unsuccessful, the profile is not added to the list.
Export	Click on a profile name and click on this button to export that profile to a file system location as a .eProfile file. The 'Save As' browser displays. Browse to the

	required location and click on the Save button. A status message displays; click on the OK button to clear the message.
Show all	Turn on this option to display profiles which belong to all the linked modules.
Close	When you have finished using the 'Export Profiles Manager' dialog, click on this button to return to the 'Export to Doors' dialog.

Export Mapping Options Dialog

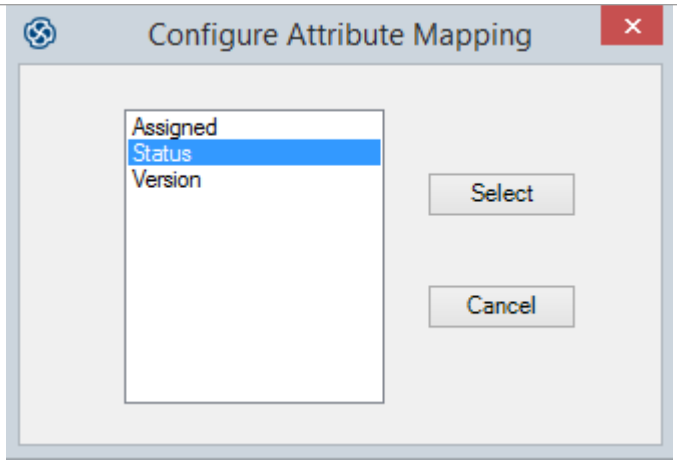
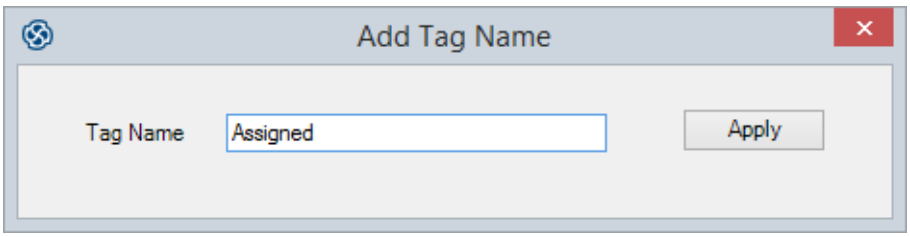
When you click on the Add button or Edit button on the 'Export Profiles Manager' dialog, the 'Mapping Options' dialog displays. This dialog maps Enterprise Architect element properties to IBM® Rational® DOORS® Object properties, and helps you to define which properties - including Tagged Values - to export. You can use the selected profile many times to update the DOORS module with changes in the Enterprise Architect Requirements.



Create or Edit an Export Profile

Field/Button	Action
Profile Name	Type in an appropriate name for the profile (if editing an existing profile, the name of the selected profile displays here).

Choose an Action	Defaults to 'Export'. You can click on the drop-down arrow and change this action to 'Import' if you are switching the direction of the data exchange.
Element Type	Click on the drop-down arrow and select the type of Enterprise Architect element to export.
Requirement Type	This option will be available only if ‘Requirement’ is selected as Element Type. Click on the drop-down arrow and select the type of Requirement Type to filter for.
Enterprise Architect	Lists the Enterprise Architect element properties that could be exported.
DOORS	Lists the DOORS properties that can be exported to. Double-click on an empty property field to map that field to an Enterprise Architect property. If there are no available properties, an error message displays; otherwise, the 'Configure Attribute Mapping' dialog displays.

	
Add Custom Field	Click on this button to select an Enterprise Architect Tagged Value name to export. The 'Add Tag Name' dialog displays. 
Remove	Click on a DOORS property value and click on this button to clear the value mapped to the corresponding Enterprise Architect property.
Save Profile	Click on this button to save the Profile definition you have created, and close the 'Mapping Options' dialog and return to the 'Export Profiles Manager' dialog.
Close	Click on this button to close the 'Mapping Options' dialog.

Import Requirements from DOORS

Using the Sparx Systems MDG Link for DOORS, you can transfer all the objects in a linked IBM® Rational® DOORS® module into the selected Enterprise Architect Package, as the required types of Enterprise Architect element.

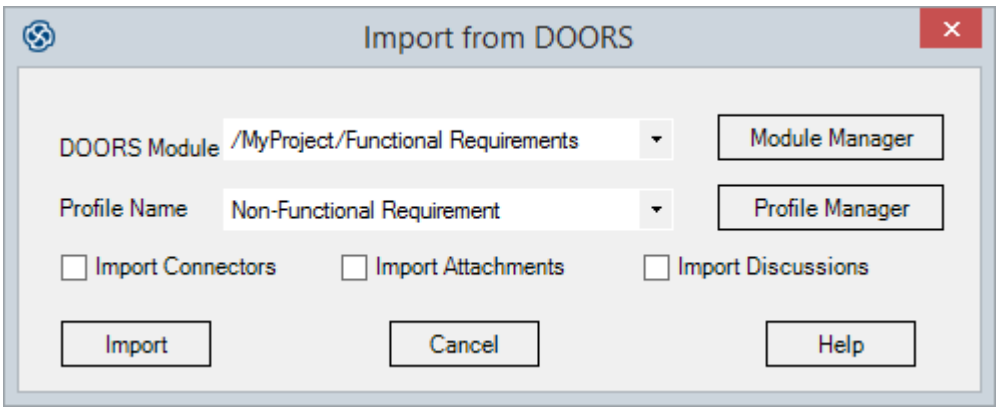
As part of this process you can create an import profile to define what the import should operate on. An import profile is a template in which you specify the DOORS object properties to be imported into mapped Enterprise Architect element properties.

Access

Ribbon	Click on target Package, then: Specialize > Add-Ins > DOORS > Import from Doors
Context Menu	Browser window | Right-click on target Package | Specialize | DOORS | Import from Doors

Import from DOORS Dialog

The 'Import from DOORS' dialog helps you to specify which module you are importing from, which import profile you are using, and whether to import connectors, attachments and Discussions.



Import Options

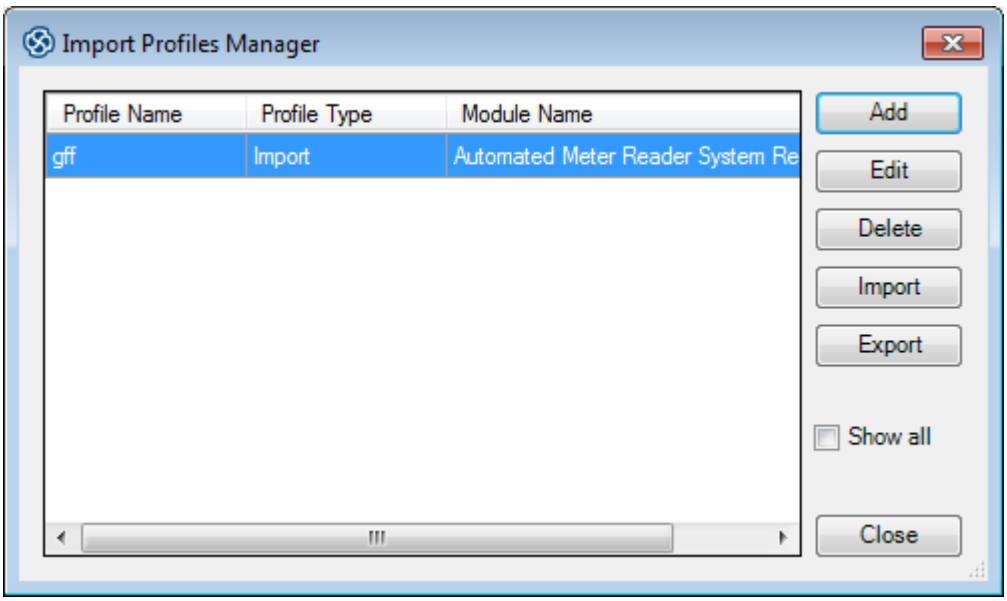
Field/Button	Action
Doors Module	Click on the drop-down arrow and select from the list of DOORS modules that are linked to this project.
Module Manager	If the module you want is not listed in the 'Doors Module' field, click on this button to display the 'Modules Manager' dialog and locate and link to the module you require.
	Click on the drop-down arrow and select

Profile Name	the Import profile to apply to this operation.
Profile Manager	If an appropriate profile to use is not listed in the 'Profile Name' field, click on this button to display the 'Import Profiles Manager' dialog (see the *Manage Import Profiles* table).
Import Connectors	Select this checkbox to import any connectors (relationships) between the already imported DOORS objects from various DOORS modules into different EA packages.
Import Attachments	Select this checkbox to import any attachments that the imported DOORS objects might have.
Import Discussions	Select this checkbox to import any Discussions associated with the imported DOORS objects.
Import	Click on this button to begin the import process.
Cancel	Click on this button to close the 'Import from Doors' dialog without performing the import.

Help	Click on this button to display this Help topic.

Import Profiles Manager Dialog

When you click on the Profile Manager button on the 'Import from Doors' dialog, the 'Import Profiles Manager' dialog displays. You use this dialog to create or import new profiles, edit or delete existing profiles, and export profiles to your preferred file system.



Manage Import Profiles

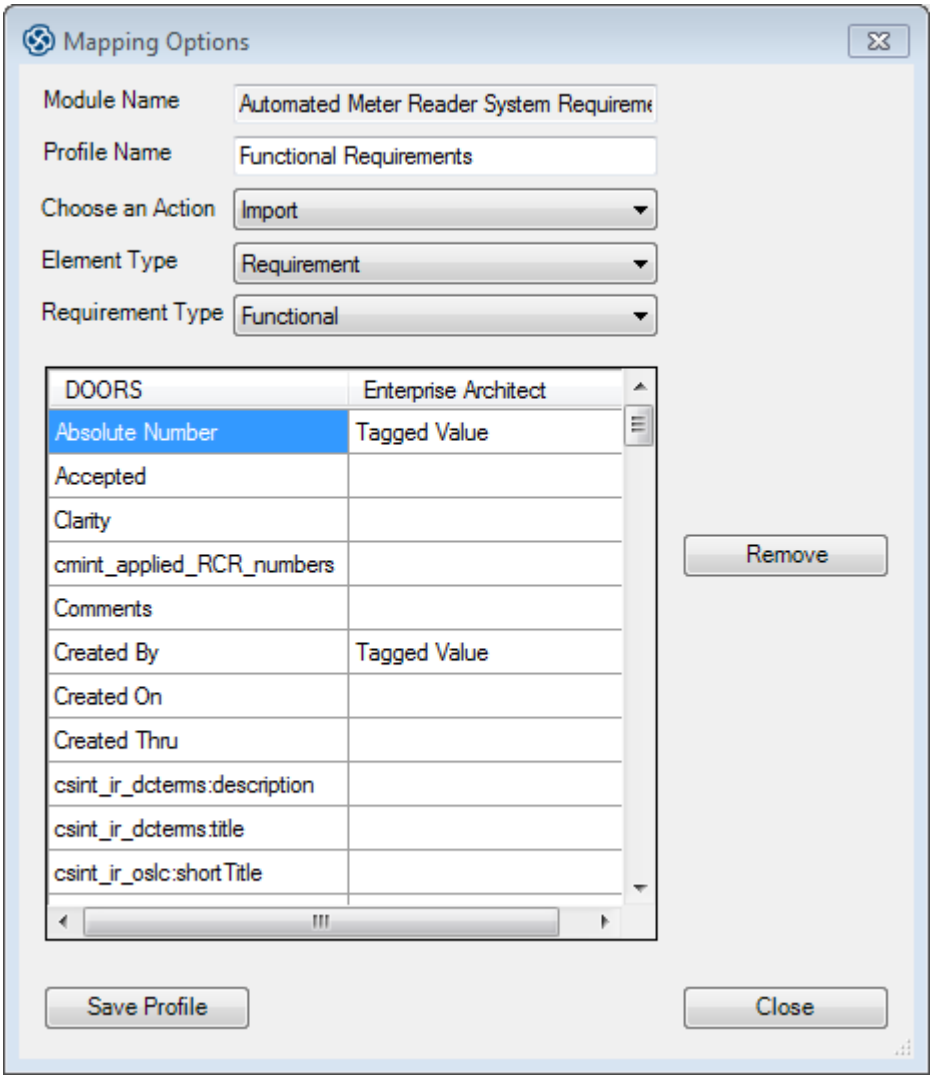
Field/Button	Action

Add	Click on this button to add a new Import profile. The 'Mapping Options' dialog displays; see the *Create or Edit an Import Profile* table.
Edit	Click on an existing profile name and click on this button to edit the parameters of the selected profile. The 'Mapping Options' dialog displays, showing the current parameters of the profile; see the *Create or Edit an Import Profile* table.
Delete	Click on an existing profile name and click on this button to immediately remove that profile from the list. There is no confirmatory prompt.
Import	Click on this button to display the 'Import a Profile - Doors Extension' browser. Browse for the location of the required profile file (.eProfile), and click on the file name and on the Open button. A status message displays. Click on the OK button; if the operation was successful, the profile name displays on the 'Import Profiles Manager' dialog. If unsuccessful, the profile is not added to the list.

Export	Click on a profile name and click on this button to export that profile to a file system location as a .eProfile file. The 'Save As' browser displays. Browse to the required location and click on the Save button. A status message displays; click on the OK button to clear the message.
Show All	Turn on this option to display profiles which belong to all the linked modules.
Close	When you have finished using the 'Import Profiles Manager' dialog, click on this button to return to the 'Import from Doors' dialog.

Import Mapping Options Dialog

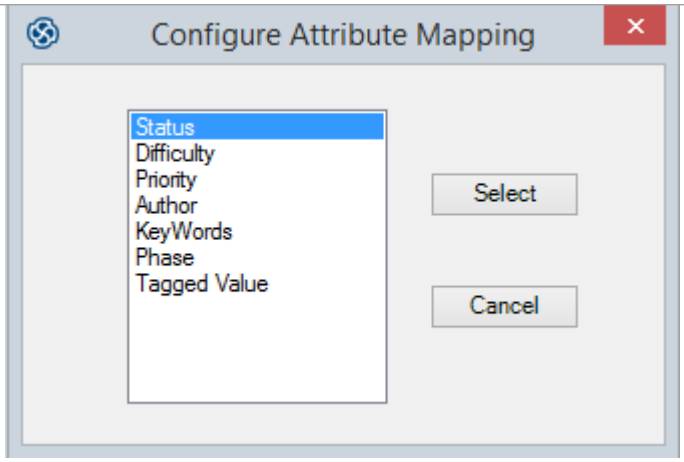
When you click on the Add button or Edit button on the 'Import Profiles Manager' dialog, the 'Mapping Options' dialog displays. This dialog maps IBM® Rational® DOORS® Object properties to Enterprise Architect element properties, and helps you to define which properties to import. You can use the profile many times to update the Enterprise Architect Package with changes in the DOORS module objects.



Create or Edit an Import Profile

Field/Button	Action
Profile Name	Type in an appropriate name for the import profile.
Choose an Action	Click on the drop-down arrow and select 'Import'. You can also reset the profile to

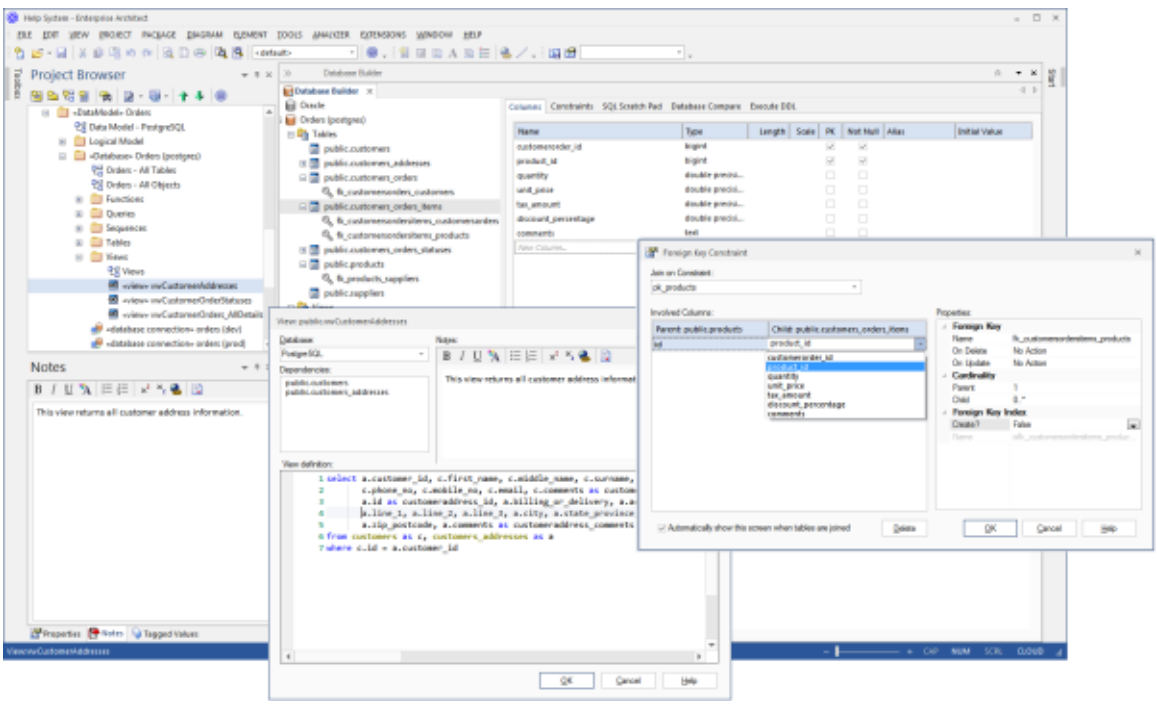
	'Export' to use it for exporting Requirements from the Enterprise Architect Package to the DOORS module.
Element Type	Click on the drop-down arrow and select the element type to create or overwrite with the DOORS data - either 'Requirement' or 'Use Case'.
Requirement Type	This option will be available only if ‘Requirement’ is selected as the Element Type. Click on the drop-down arrow and select any type you want to set on the Requirement elements resulting from the import.
DOORS	Lists the DOORS object properties available in the selected DOORS module.
Enterprise Architect	Lists the Enterprise Architect properties mapped to the DOORS properties. You can double-click on an empty property field to display the 'Configure Attribute Mapping' dialog, which provides a list of Enterprise Architect properties that can be mapped to the DOORS property.

	 Click on a property and on the Select button to map it to the DOORS property.
Remove	If necessary, click on a mapped Enterprise Architect property field and click on this button to clear the field and unmap the property.
Save Profile	Click on this button to save the profile definition you have created and return to the 'Import Profiles Manager' dialog.
Close	Click on this button to close the 'Mapping Options' dialog.

Information Engineering

Design, Create and Manage Conceptual, Logical and Physical Data Models

The power of model-based system development is the ability to visualize, analyze and design all aspects of a system. Being able to view and manage information and data alongside other models of a system provides great clarity and reduces the chance of error. Enterprise Architect has extensive support for the data modeling discipline, ranging from the representation of information in a conceptual model right down to the generation of database objects. Whether you are generating database objects from the UML model or reverse engineering legacy DBMS into a model for analysis, the tool features will save time and valuable project resources.



This illustration shows the Database Builder Interface including DDL Generation and the Foreign Key dialog.

Enterprise Architect supports the modeling of information at the conceptual, logical and physical layers. Using a number of standard features, these models can be interconnected, providing traceability. The logical and physical models can also be generated automatically using a fully customizable Transformation engine. Legacy systems can be imported, analyzed and compared using the handy reverse engineering facility.

In this topic you will learn how to use the feature rich toolset including the Database Builder to design, create, manage, visualize data including reverse and forward engineering of data models to live database.

The Database Builder tool can be used to create and maintain physical data models and can connect to a running DBMS, so you can therefore import, generate, compare and alter a live database.

Getting Started

Information Modelers, Data Modelers and Architects are responsible for creating models of an organization's information that span multiple levels of abstraction, from conceptual through to logical and physical. The conceptual models are technology independent and can be used for discussions with business people and domain experts, allowing the basic concepts in the domain to be represented, discussed and agreed upon. The logical model elaborates the conceptual model, adding more detail and precision but is still typically technology neutral, allowing Information Analysts to discuss and agree on logical structures. The physical model applies technology specific data to the models and allows engineers to discuss and agree on technology decisions in preparation for generation to a target environment, such as a database management system.

Selecting the Perspective

Enterprise Architect partitions the tool's extensive features into Perspectives, which ensures that you can focus on a specific task and work with the tools you need without the distraction of other features. To work with the Data Modeling features you first need to select one of these Perspectives:

 <perspective name> > Database Engineering > Database Engineering

<perspective name> > Database Engineering > Entity Relationships

Setting the Perspective ensures that the Case Management Model and Notation diagrams, their tool boxes and other features of the Perspective will be available by default.

Example Diagram

An example diagram provides a visual introduction to the topic and allows you to see some of the important elements and connectors that are created in specifying or describing the way a data model is defined including: Tables, Views, Procedures, Sequences, Functions.

Data Model Types

Information can be modeled at a number of level of abstraction starting with a conceptual model that is typically created by or for business people, a Logical model which is used by business and systems analysts and a physical model which is the concern of the technologists such as database engineers. In this topic you will learn how to manage all three levels of information models.

Creating and Managing Data Models

In this topic you will learn how to work in detail using Enterprise Architect to manage your Physical Database schema. This includes the use of the Database Builder tool which allows you to interact with any number of live database through an ODBC connection.

Import Database Schema

This topic will show you how to connect to a live database including Production, Test and Development systems and reverse engineer the database into a model creating Tables, Views, Procedures, Declarative Referential Integrity and more. A diagram of the database is automatically created and the elements such as tables can be related to other elements in the model including Conceptual and Logical Models, Programming classes tests and more.

Generate Database Definition Language (DDL)

In this topic you will learn how to harness the power of the data models by generating Database Definition Language code directly from the model. Enterprise Architect can

generate code into a wide range of Database Management systems.

Supported Database Management Systems

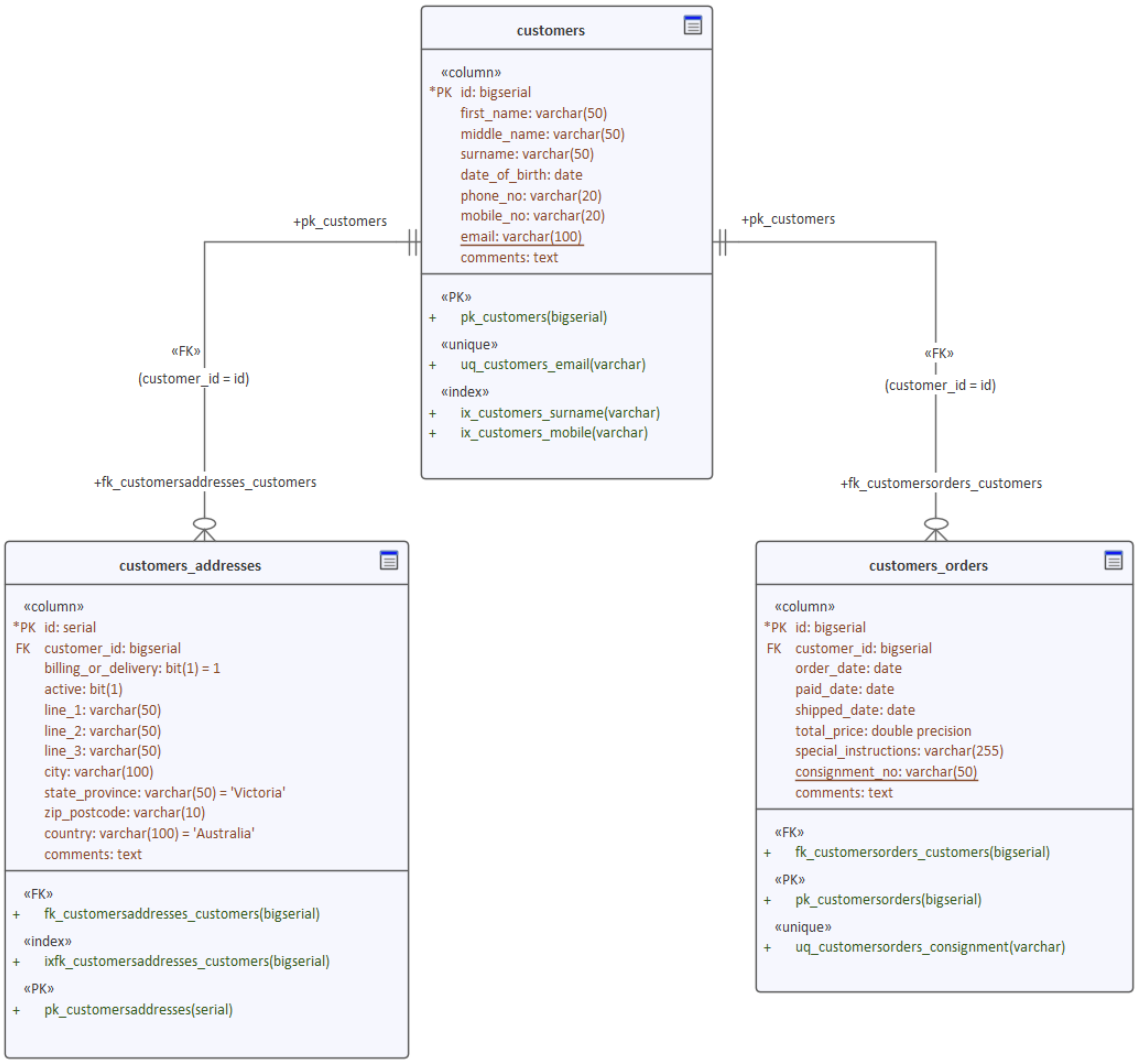
Enterprise Architect has rich support for most of the main stream Database Management Systems (DBMS). This feature allows models from disparate systems to be compared either for code generation or for analysis by using the import feature. This topic lists the supported DBMS and

More Information

This section provides useful links to other topics and resources that you might find useful when working with the Data Modeling tool features.

Example Diagram

Using the Database engineering features of Enterprise Architect you can create rich models of the objects that make up a data model at any level of abstraction from Conceptual through Logical to Physical. These models are created by adding tables and other database objects from the toolbox or by reverse engineering and existing database into a model from a range of RDBMSs. A database diagram can contain Tables, VIews, Procedures, Sequences and Functions. Table Columns are annotated as Primary and Foreign Keys are modeled using specialized association relationships. In this example the user has created a simple physical data model of Customers and their Addresses and Orders.



Physical Data model showing Tables with Columns and Primary and Foreign Keys.

Working with Data Model Types

Enterprise Architect provides a number of features to assist in the process of creating models of information, including the ability to develop conceptual, logical and physical models and to be able to trace the underlying concepts between the models. The physical models can be developed for a wide range of database systems, and forward and reverse engineering allows these models to be synchronized with live databases.

Data Models

Type	Description
Conceptual Data Models	Conceptual data models, also called Domain models, establish the basic concepts and semantics of a given domain and help to communicate these to a wide audience of stakeholders. Conceptual models also serve as a common vocabulary during the analysis stages of a project; they can be created in Enterprise Architect using Entity-Relationship or UML Class models.

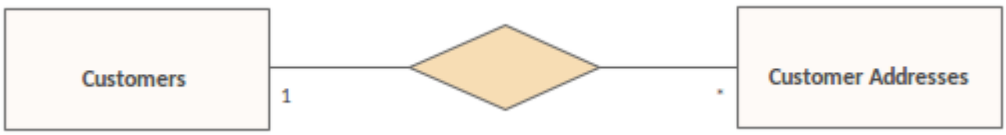
Logical Data Models	Logical data models add further detail to conceptual model elements and refine the structure of the domain; they can be defined using Entity-Relationship or UML Class models. One benefit of a Logical data model is that it provides a foundation on which to base the Physical model and subsequent database implementation. Entity-relationship modeling is an abstract and conceptual database modeling method, used to produce a schema or semantic data model of, for example, a relational database and its requirements, visualized in Entity-Relationship Diagrams (ERDs). ERDs assist you in building conceptual data models through to generating Data Definition Language (DDL) for the target DBMS. A Logical model can be transformed to a Physical data model using a DDL Transformation.
Physical Data Models	Physical data models in Enterprise Architect help you visualize your database structure and automatically derive the corresponding database schema; you use Enterprise Architect's

	UML Profile for Data Modeling specifically for this purpose. The profile provides useful extensions of the UML standard that map database concepts of Tables and relationships onto the UML concepts of Classes and Associations; you can also model database columns, keys, constraints, indexes, triggers, referential integrity and other relational database features. Because Enterprise Architect helps you visualize each type of data model in the same repository, you can easily manage dependencies between each level of abstraction to maximize traceability and verify completeness of system implementation.

Conceptual Data Model

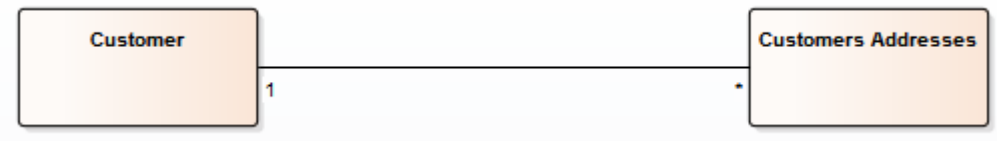
A Conceptual data model is the most abstract form of data model. It is helpful for communicating ideas to a wide range of stakeholders because of its simplicity. Therefore platform-specific information, such as data types, indexes and keys, is omitted from a Conceptual data model. Other implementation details, such as procedures and interface definitions, are also excluded.

This is an example of a Conceptual data model, rendered using two of the notations supported by Enterprise Architect.



Entity Relationship diagram showing a One-to-Many relationship

Using Entity-Relationship (ER) notation, we represent the data concepts 'Customers' and 'Customers Addresses' as Entities with a 1-to-many relationship between them. We can represent exactly the same semantic information using UML Classes and Associations.



Unified Modeling Lnaguage diagram showing the same One-to_many Relationship

Whether you use UML or ER notation to represent data concepts in your project depends on the experience and preferences of the stakeholders involved. The detailed structure of the data concepts illustrated in a Conceptual data model is defined by the Logical data model.

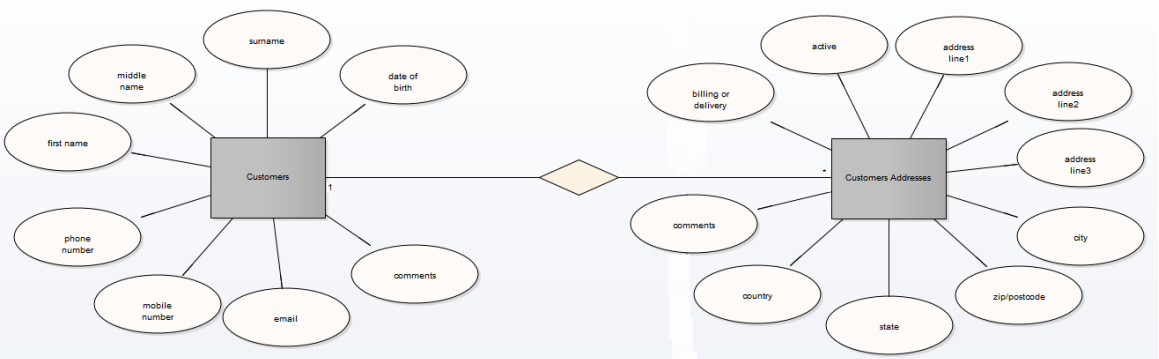
Entity Relationship Diagrams (ERDs)

According to the online Wikipedia:

An entity-relationship model (ERM) is an abstract and conceptual representation of data. Entity-relationship modeling is a database modeling method, used to produce a type of conceptual schema or semantic data model of a system, often a relational database, and its requirements in a top-down fashion. Diagrams created by this process are called Entity-Relationship Diagrams, ER Diagrams, or ERDs.

Entity Relationship Diagrams in Enterprise Architect

Entity Relationship diagrams in Enterprise Architect are based on Chen's ERD building blocks: entities (tables) are represented as rectangles, attributes (columns) are represented as ellipses (joined to their entity) and relationships between the entities are represented as diamond-shape connectors.



ERD technology in Enterprise Architect assists you in every stage from building conceptual data models to generating Data Definition Language (DDL) for the target DBMS.

ERD and ERD Transformations

Enterprise Architect enables you to develop Entity Relationship diagrams quickly and simply, through use of an MDG Technology integrated with the Enterprise Architect installer.

The Entity Relationship diagram facilities are provided in the form of:

- An Entity Relationship diagram type, accessed through the 'New Diagram' dialog
- An Entity Relationship Diagram page in the Diagram Toolbox
- Entity Relationship element and relationship entries in the 'Toolbox Shortcut' menu and Quick Linker

Enterprise Architect also provides transformation templates to transform Entity Relationship diagrams into Data Modeling diagrams, and vice versa.

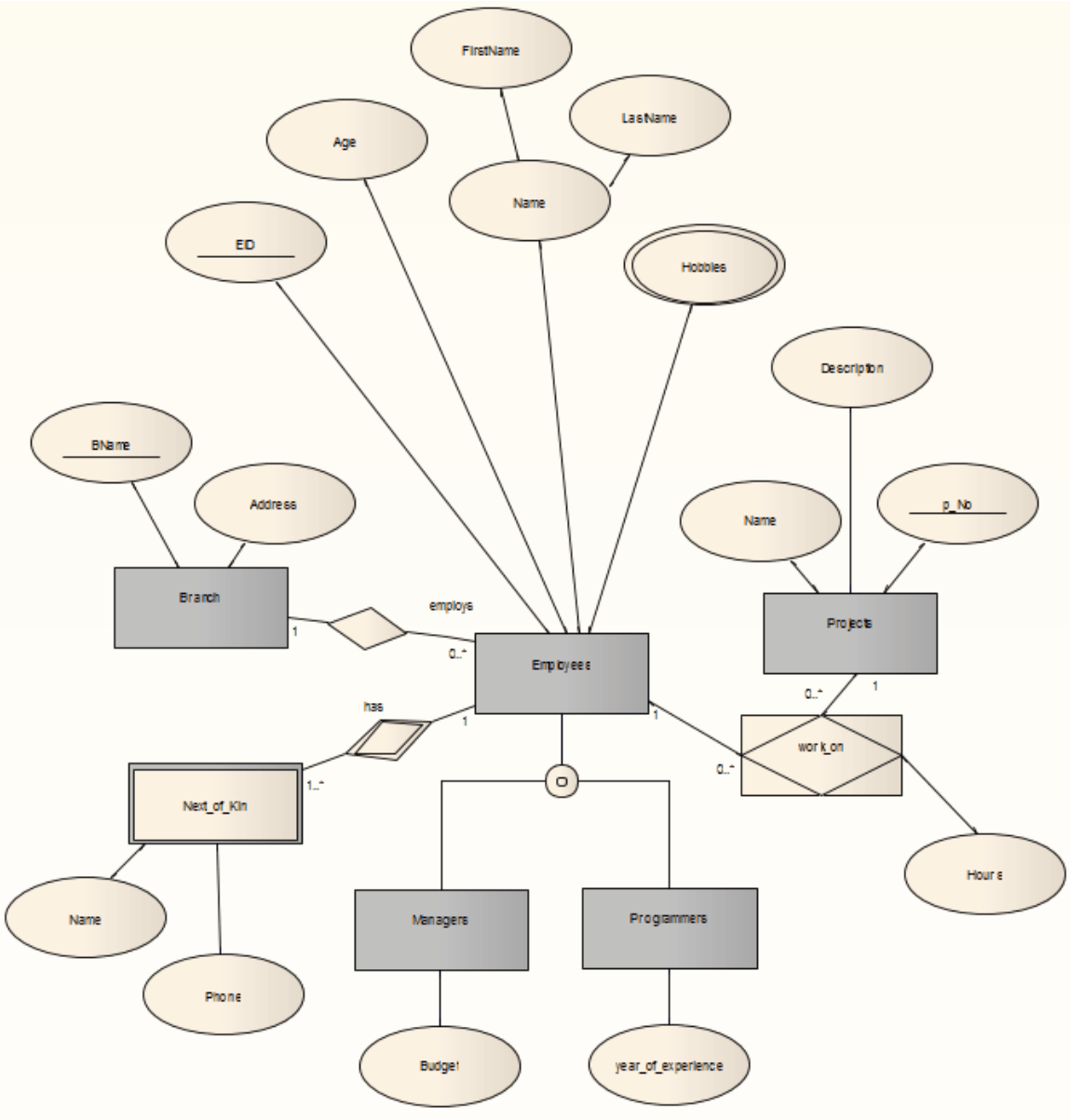
Entity Relationship Diagram Toolbox Page

You can access the 'Entity Relationship Diagram' page of

the Diagram Toolbox by specifying 'Entity Relationship Diagrams' in the Toolbox 'Find Toolbox Item' dialog

- Entity is an object or concept that is uniquely identifiable; the property of 'Multiplicity' in the SourceRole and TargetRole definitions for the Relationship connector can be used to define the cardinality of an Entity that participates in this relationship
- Attribute is a property of an entity or a relationship type
- N-ary Association represents unary (many-to-many recursive) or ternary relationships and can also be used to represent relationships that have attributes among the entities; the N-ary Association element should always be at the target end of a connector
- Connector is a connector between an Entity and an Attribute, and between two Attributes
- Relationship is a diamond-shape connector, representing the meaningful association among entities
- Disjoint and Overlapping represent the relationships between the super-class Entity and the sub-class Entity

A typical Entity Relationship diagram



Tagged Values

Some of the Entity Relationship diagram components can be modified by Tagged Values, as indicated:

Component	Tagged Value / Notes
Entity	isWeakEntity Notes: If true, this entity is a weak entity.
Attribute	attributeType Notes: There are four valid options: 'normal', 'primary key', 'multi-valued' and 'derived'
Attribute	commonDataType Notes: Defines the common data type for each attribute.
Attribute	dbmsDataType Notes: Defines the customized DBMS data type for each attribute. This option is only available when the *commonDataType* tag is set to 'na'. You must define the customized type first through the 'Settings > Reference Data > Settings > Database Datatypes' ribbon option.
N-ary Association	isRecursive Notes: If true, the N-ary Association represents the many-to-many recursive relationship.

	For one-to-many and one-to-one recursive relationships, we suggest using the normal Relationship connector. Sometimes you might want to limit the stretch of the diamond-shape Relationship connectors; simply pick a Relationship connector, right-click to display the context menu, and select the 'Bend Line at Cursor' option.
Relationship	isWeak Notes: If true, the Relationship is a weak relationship.
Disjoin Overlapping	Participation Notes: There are two valid options, 'partial' and 'total'.

Notes

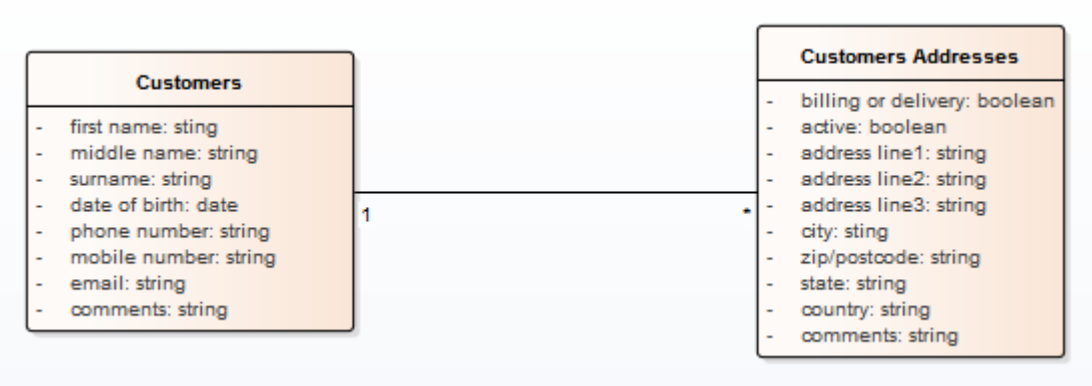
- Entity Relationship diagrams are supported in the Corporate, Unified and Ultimate Editions of Enterprise Architect

Logical Data Model

Logical data models help to define the detailed structure of the data elements in a system and the relationships between data elements. They refine the data elements introduced by a Conceptual data model and form the basis of the Physical data model. In Enterprise Architect, a Logical data model is typically represented using the UML Class notation.

Example

The following diagram is a simple example of a Logical data model. The Logical Model adds detail to the Conceptual Model but without going to the level of specifying the Database Management System that will be used.



Conceptual Data Model with tables modeling Customers and their Addresses.

Note that the data elements 'Customers' and 'Customers Addresses' contain UML attributes; the names and generic

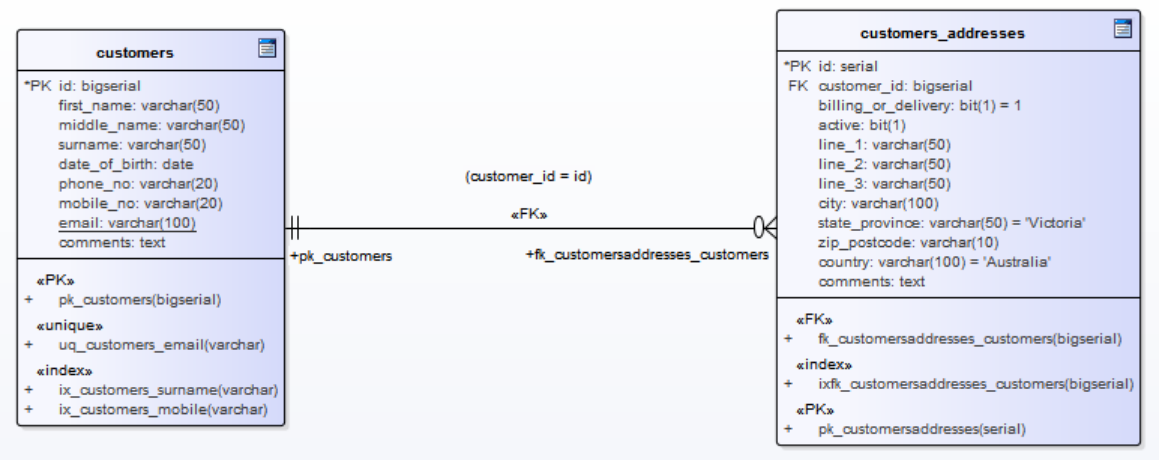
data types to remain platform-independent.
Platform-specific data types and other metadata that relate to a specific DBMS implementation are defined by the Physical data model.

Physical Data Models

A Physical Data Model visually represents the structure of data as implemented by a relational database schema. In addition to providing a visual abstraction of the database structure, an important benefit of defining a Physical Data Model is that you can automatically derive the database schema from the model. This is possible due to the richness of metadata captured by a Physical Data Model and its close mapping to aspects of the database schema, such as database Tables, columns, Primary Keys and Foreign Keys.

Example Data Model

This example shows a Physical Data Model that could be used to automatically generate a database schema. Each Table is represented by a UML Class; Table columns, Primary Keys and Foreign Keys are modeled using UML attributes and operations. This model demonstrates the use of the Information Engineering connector style.



Notation

The example model is defined using Enterprise Architect's UML Profile for Data Modeling; the relationship between the Tables uses the default Information Engineering notation.

Information Engineering is one of three notations that Enterprise Architect supports to help Data Modelers identify cardinality in relationships. You can change the notation by selecting the 'Design > Diagram > Manage > Properties' ribbon option, clicking on the 'Connectors' page and selecting the required option in the 'Connector Notation' drop-down list. You would most probably change the notation to IDEFX1, but the UML2.1 notation is also available.

Default DBMS

Prior to creating a Physical Data Model it is advisable for you to set the default DBMS for the project. Setting a default DBMS ensures that all new database elements that are created on diagrams are automatically assigned the default DBMS.

If the default DBMS is not set, new Tables are created without a DBMS assigned, this restricts Enterprise

Architect's ability to model the physical objects correctly. For example Enterprise Architect is unable to determine the correct list of datatypes for columns.

You can set the default DBMS type using:

- 'Start > Appearance > Preferences > Preferences > Source Code Engineering > Code Editors', or
- 'Settings > Reference Data > Settings > Database Datatypes or
- 'Develop > Data Modeling > Datatypes or
- The second data entry field in the Code Generation Toolbar

Note: When modeling via the Database Builder the default DBMS is defined at the model level (as a Tagged Value 'DBMS' against the <<Database>> Package) instead of at the project level, thereby allowing for greater flexibility when projects involve multiple DBMSs.

DDL Transformation

The DDL transformation converts the logical model to a data model structured to conform to one of the supported DBMSs. The target database type is determined by which DBMS is set as the default database in the model (see the *Database Datatypes* Help topic, 'Set As Default' option). The data model can then be used to automatically generate DDL statements to run in one of the system-supported database products.

The DDL transformation uses and demonstrates support in the intermediary language for a number of database-specific concepts.

Concepts

Concept	Effect
Table	Mapped one-to-one onto Class elements. 'Many-to-many' relationships are supported by the transformation, creating Join tables.
Column	Mapped one-to-one onto attributes.
Primary Key	Lists all the columns involved so that they exist in the Class, and creates a

	Primary Key Method for them.
Foreign Key	A special sort of connector, in which the Source and Target sections list all of the columns involved so that: • The columns exist • A matching Primary Key exists in the destination Class, and • The transformation creates the appropriate Foreign Key

MDG Technology to customize default mappings

DDL transformations that target a new, user defined DBMS require an MDG Technology to map the PIM data types to the new target DBMS.

To do this, create an MDG Technology .xml file named 'UserDBMS Types.xml', replacing UserDBMS with the name of the added DBMS. Place the file in the EA\MDGTechnologies folder. The contents of the MDG Technology file should have this structure:

```
<MDG.Technology version="1.0">
    <Documentation id="UserdataTypes" name="Userdata
```

```
Types" version="1.0" notes="DB Type mapping for
UserDBMS"/>
    <CodeModules>
        <CodeModule language="Userdata" notes="">
            <CodeOptions>
                <CodeOption
name="DBTypeMapping-bigint">BIGINT</CodeOption>
                <CodeOption
name="DBTypeMapping-blob">BLOB</CodeOption>

                <CodeOption
name="DBTypeMapping-boolean">TINYINT</CodeOptio
n>
                <CodeOption
name="DBTypeMapping-text">CLOB</CodeOption>
                ...
            </CodeOptions>
        </CodeModule>
    </CodeModules>
</MDG.Technology>
```

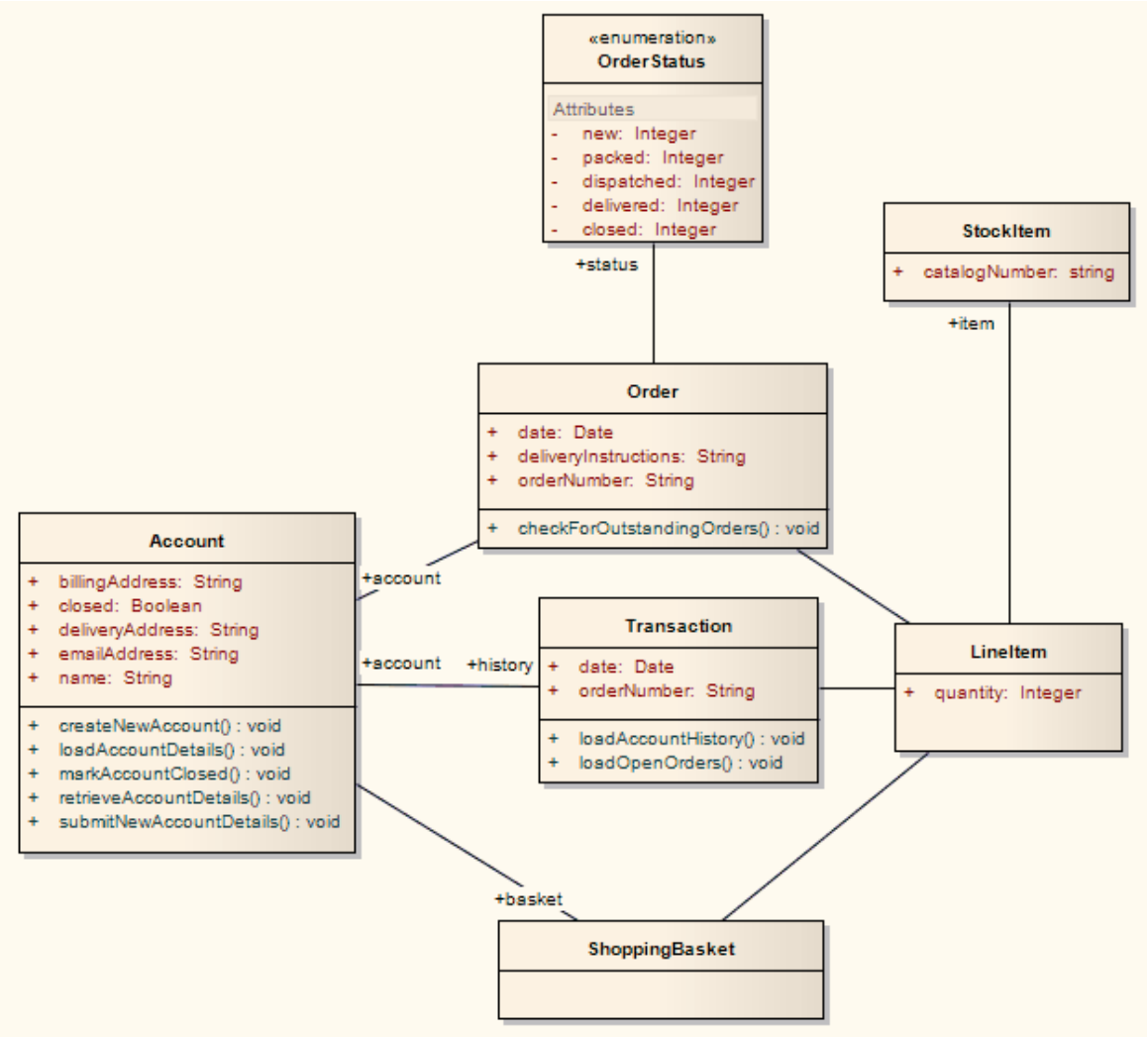
As an example, 'text' is a Common Type (as listed in the 'Database Datatypes' dialog) that maps to a new DBMS's 'CLOB' data type.

Notes

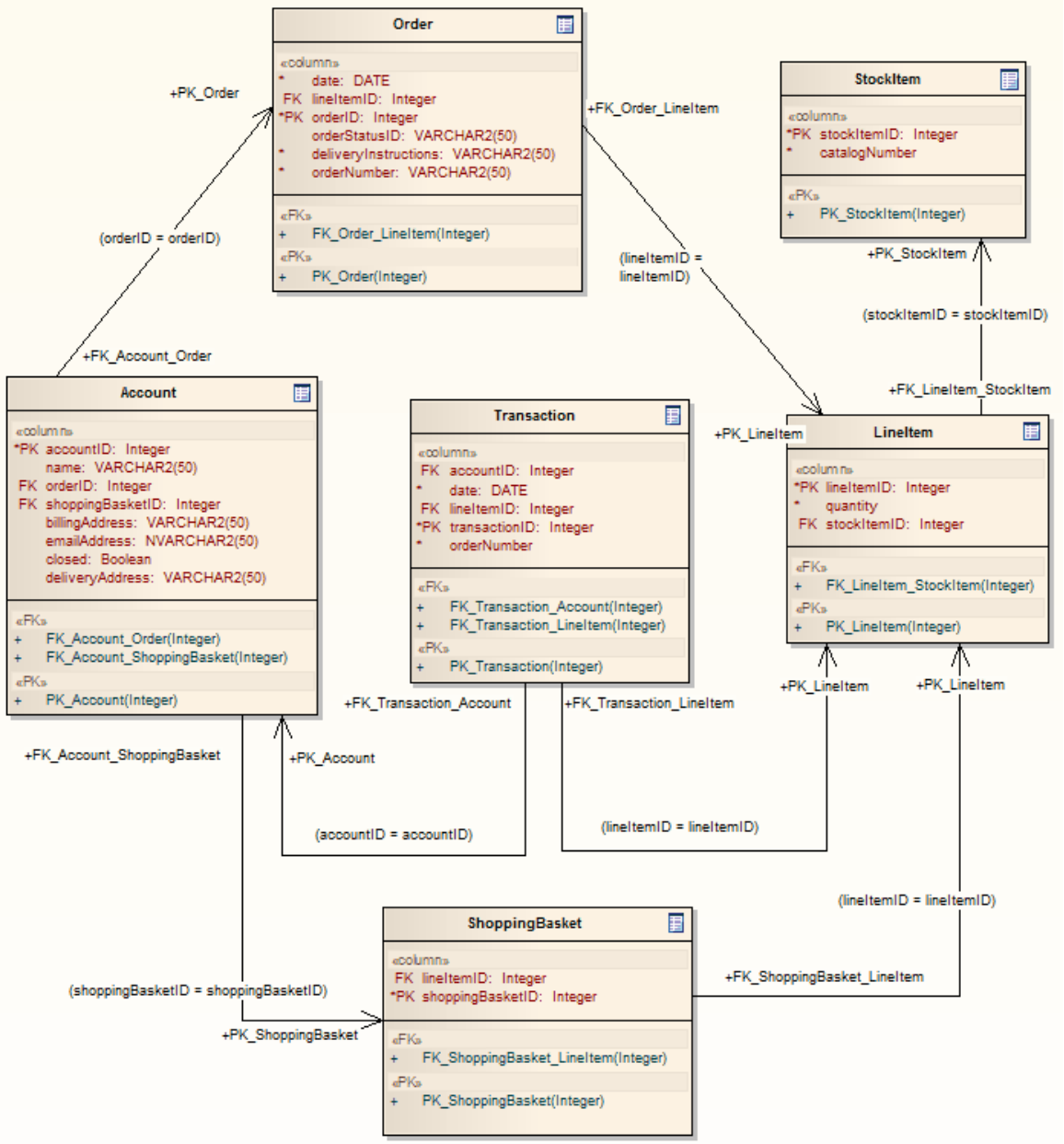
- You can define DBMS-specific aspects not depicted in a Logical model, such as Stored Procedures, Triggers, Views and Check Constraints, after the transformation; see the *Physical Data Model* Help topic

Example

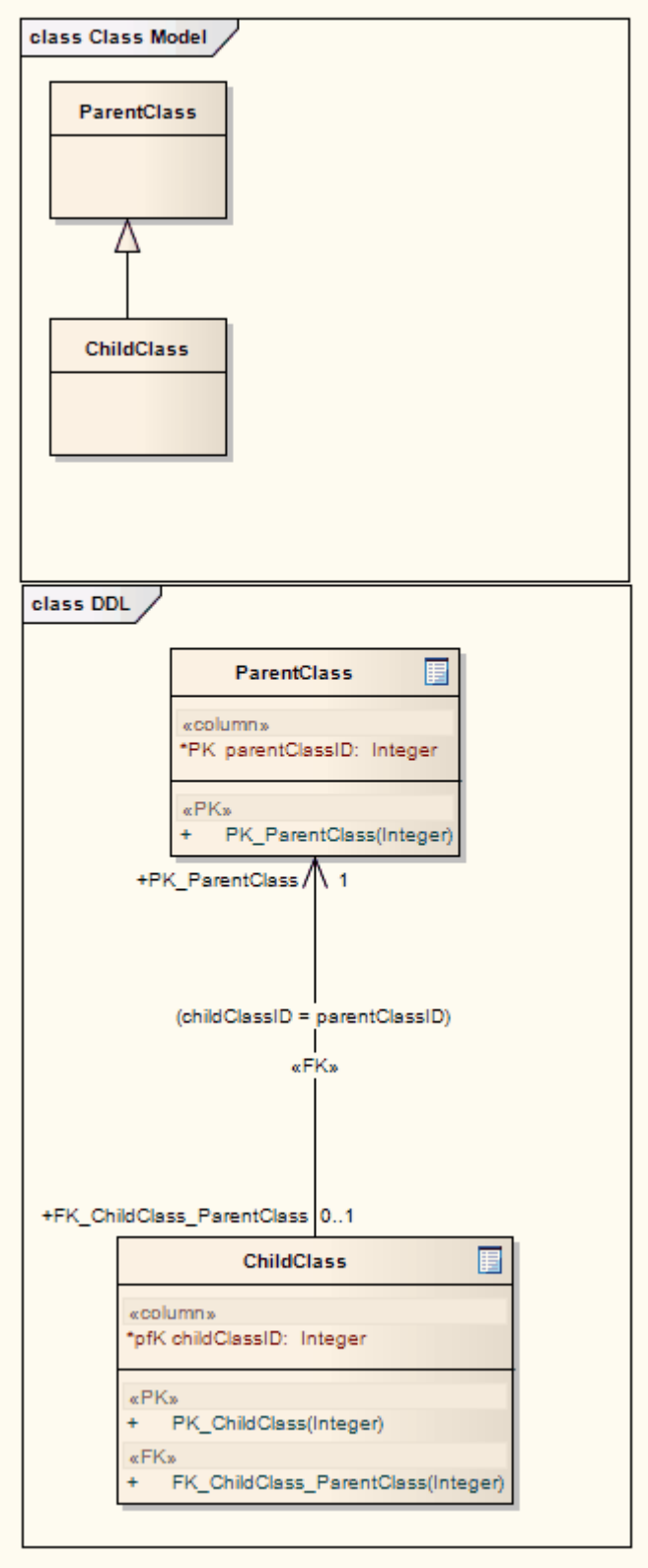
The PIM elements



After transformation, become the PSM elements

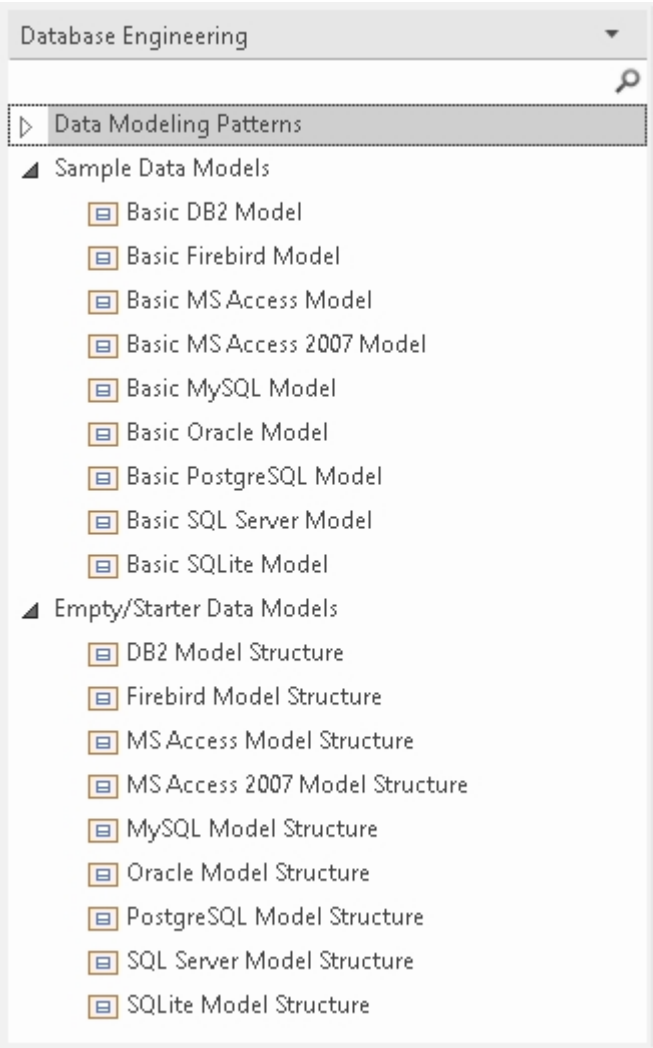


Generalizations are handled by providing the child element with a Foreign Key to the parent element, as shown. Copy-down inheritance is not supported.


class Class Model
ParentClass
ChildClass
class DDL
ParentClass
«column»
*PK parentClassID: Integer
«PK»
+ PK_ParentClass(Integer)
+PK_ParentClass
1
(childClassID = parentClassID)
«FK»
+FK_ChildClass_ParentClass
0..1
ChildClass
«column»
*pfK childClassID: Integer
«PK»
+ PK_ChildClass(Integer)
«FK»
+ FK_ChildClass_ParentClass(Integer)

Creating and Managing Data Models

Enterprise Architect is a fully featured database modeling platform that enables the user to work with their Physical Data models at all stages, from design right through to the implementation of the live database, for a wide range of database management systems such as Microsoft SQL Server, Oracle, PostgreSQL and MySQL.



This figure shows the starter model wizard patterns for database design for a range of RDBMS.

Create a Data Model from a Model Pattern

The easiest way to create a Data Modeling workspace is to use the predefined Database Model Patterns, available through the Model Wizard (Start Page 'Create from Pattern' tab). Enterprise Architect provides a Pattern for each DBMS supported by the system.

Access

Display the Model Wizard (Start Page 'Create from Pattern' tab) using any of the methods outlined here.

In the Model Wizard, select the 'Database Engineering' Perspective.

Ribbon	Design > Package > Model Wizard
Context Menu	Right-click on Package | Add a Model Using Wizard
Keyboard Shortcuts	Ctrl+Shift+M
Other	Browser window caption bar menu | New Model from Pattern

Create a Data Model

Field/Button	Action
Add to Package	Displays the name of the selected root Package.
Technology	Click on 'Database'.
Name	If necessary, expand the Database Engineering group of Patterns. Click on the checkbox against each Database Management System you are supporting in the model.
All	Click on this button to select the checkboxes for all Database Engineering model types and the Entity Relationship diagram, to include them all in the model.
None	Click on this button to clear all selected checkboxes so that you can re-select certain checkboxes individually.
OK	Click on this button to add to the Browser window the Packages and diagram for

	each Database Management System you are modeling.

What each Data Modeling Pattern provides

- A summary diagram of the model
- A Report Specification Artifact element (on the summary diagram) that can be used to quickly document the data model
- A Package for each of the Logical and Physical models
- Within the Physical Model Package, a predefined hierarchy of sub-Packages, one for each object type supported by the DBMS being modeled (such as Tables, Views, Procedures and Functions); these automatically organize the database objects as they are added
- The DBMS type for the workspace
- A default owner
- A Data Modeling diagram in each Package with the connector notation set to IDEF1X

Notes

- Once a data modeling workspace has been created, you can begin to develop your model in one of two ways:

- Through the Database Builder, which is a purpose-built view that supports database modelers
- Through the Browser window and diagrams, which is the traditional method that might suit users who are experienced UML modelers

Create a Data Model Diagram

To model the structure of a relational database you use Data Modeling diagrams, which are extended Class diagrams. When you open a Data Modeling diagram the matching Diagram Toolbox is automatically opened, which contains the diagram elements:

- Table
- View
- Procedure
- Sequence
- Function
- Association and
- Database Connection

Access

Display the 'New Diagram' dialog using any of the methods outlined here.

Ribbon	Design > Diagram > Add
Context Menu	Right-click on Package | Add Diagram Right-click on element | Add | Add Diagram

Keyboard Shortcuts	Ctrl+Insert
Other	Browser window caption bar menu | New Diagram

Create a Data Modeling diagram

Field/Button	Action
Package	Defaults to the name of the Package selected in the Browser window or, if the parent is an element, the name of the Package containing that element. If you are adding a diagram directly to a Package and notice that it is not the correct Package, click on the  button and browse for the correct Package.
Parent	If you are adding a diagram to an element, this field displays the element name.
Diagram	This field defaults to the name of the parent Package or element. If required, overtype the default name

	with your preferred name.
Select From	Click on this header and select the Perspective Group and Perspective or Workspace most appropriate to the area you are working in (in this instance, 'Information Engineering > Database Models'). From the options listed in the panel, click on 'Extended'.
Diagram Types	Click on 'Data Modeling'.
OK	Click on this button to create the diagram. The Diagram View displays the blank diagram, and the 'Data Modeling' pages display in the Diagram Toolbox. Drag elements and connectors from the Toolbox onto your diagram, to create your data model.

Notes

- The default diagram connector notation for all new diagrams is Information Engineering, although many data

modelers prefer the notation IDEF1X; to make this change select 'Design > Diagram > Manage > Properties > Connectors' and click on the required option in the 'Connector Notation' drop-down list

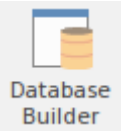
Example Data Model Diagram

This example of a Data Model diagram shows a data model of a bookstore warehousing system. The tables are modeled using a stereotyped class with a compartment for Columns which displays the name and type of the Columns. Primary and Foreign Keys are indicated by stereotypes on the columns. You can examine this model in greater detail in the Example model, installed with Enterprise Architect and available from the following ribbon location.

Start > Help > Help > Open the Example Model

Data modeling diagram with suppressed operation compartment showing tables connected to indicate foreign key relationships.

The Database Builder

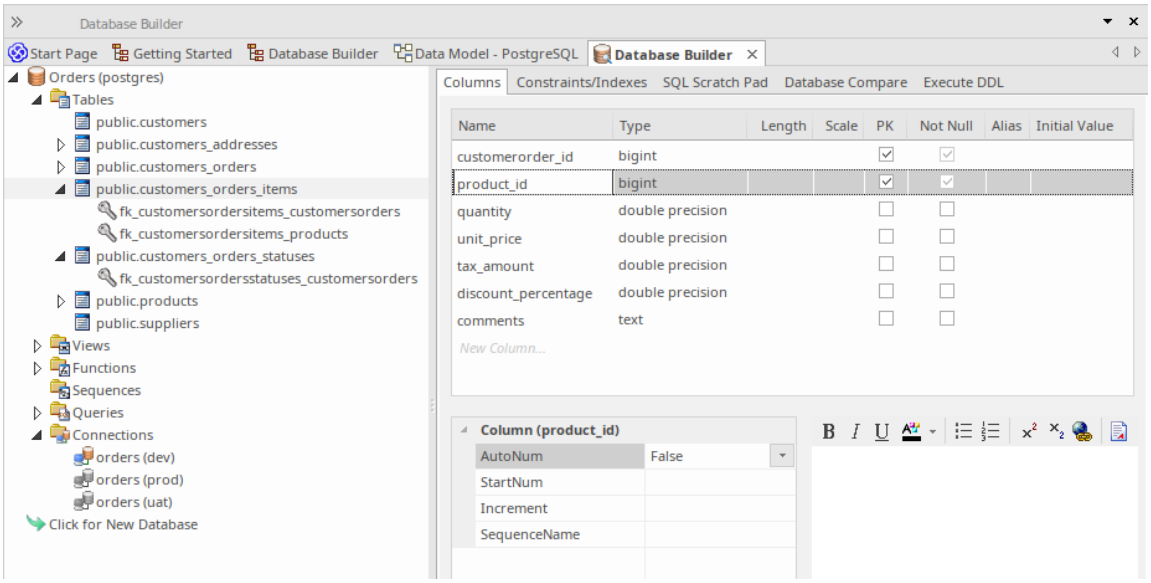


The Database Builder is a tailored interface for the data modeler; all database-related modeling tasks can be performed in a single location. The interface and its related screens include only the information relevant to data modeling, thereby streamlining and simplifying the modeling process.

Access

Ribbon	Develop > Data Modeling > Database Builder

Database Builder



This figure shows the Database Builder loaded with the 'Orders (postgres)' data model as it appears in the Example model.

Overview

The interface of the Database Builder consists of:

- A Tree of data models, listing all defined data models in the current repository
- A 'Columns' tab through which you directly manage the Table columns
- A 'Constraints/Indexes' tab for the direct management of Table constraints such as Primary Keys, Foreign Keys and Indexes
- An SQL Scratch Pad that you can use to run ad-hoc SQL queries against a live database
- A 'Database Compare' tab that displays the results of

comparisons between the data model and a live database
- An 'Execute DDL' tab on which you can execute generated DDL against a live database, instantly

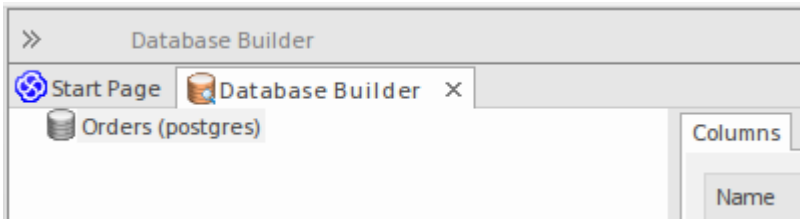
You can use the Database Builder to:
- Create, edit and delete database objects (Tables, Views, Procedures, Sequences and Functions)
- Create, edit and delete Table constraints (Primary Keys, Indexes, Unique Constraints, Check Constraints and Triggers)
- Create, edit and delete Table Foreign Keys
- Reverse engineer database schema information
- Generate DDL from a modeled database
- Compare a live database schema with a modeled database
- Execute generated DDL against a live database
- Execute adhoc SQL statements against a live database

Notes

- The Database Builder is available in the Corporate, Unified and Ultimate Editions of Enterprise Architect

Opening the Database Builder

When you first open the Database Builder, it searches the entire project for all Packages that have the stereotype <<Data Model>> and loads the corresponding data models as root nodes into the tree. A grayed-out icon indicates that the details of the data model are not loaded.



This figure shows the Database Builder with a single unloaded data model called 'Orders (postgres)'.

Using the Database Builder

You can start working in the Database Builder in one of these two ways:

Task	Action
Create a new data model	Once you have opened the Database Builder view, right-click in the empty space of the tree and select 'New Data Model' to invoke the Model Wizard (Start Page 'Create from Pattern' tab).

Load an existing Data Model	Once the Database Builder view is opened, load any of the defined data models by either: • Right-clicking on the name and selecting 'Load', or • Double-clicking on the name

Data Model Properties

In earlier versions of Enterprise Architect (prior to the introduction of the Database Builder) it was necessary for the data modeler to manually set properties on database objects before some tasks were allowed. For example, Enterprise Architect would not allow the definition of a Table column without the Table first being assigned a DBMS. This was because the DBMS controls the list of available datatypes.

To improve efficiency and the user experience, the Database Builder defines defaults for a number of properties at the data model level and then applies these default values automatically whenever new objects are created.

Properties

Option	Description
DBMS	*Defined against:* The data model's <<Database>> Package *Defined as:* Tagged Value *Details:* Defines the DBMS of the current data model *Extra Information:* • Controls which logical folders are shown for the current data model in the Database Builder's tree • Controls what DBMS rules are applied during database comparisons • Is automatically assigned to every new database object created in the current data model
DefaultOwner	*Defined against:* The data model's <<Database>> Package *Defined as:* Tagged Value *Details:* Defines the default Owner for the current data model *Extra Information:* • Is automatically assigned to every new database object created in the current data model, if the DBMS supports owners/schemas

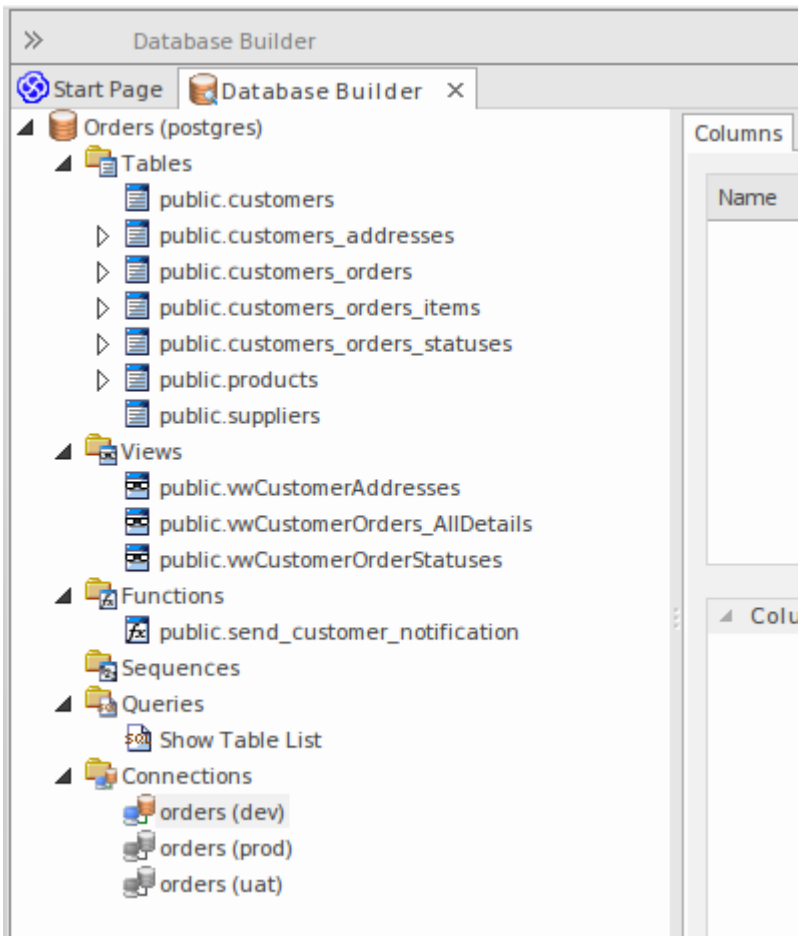
DefaultConnection	*Defined against:* The data model's <<Database>> Package *Defined as:* Tagged Value *Details:* (Optional) the name of the default connection *Extra Information:* • Whenever a data model is loaded, the 'DefaultConnection' property is checked; if present, the Connection by that name is automatically made active • The database engineering model Patterns do not define a value for this property, it is created or updated whenever a user sets a Connection as the default

Notes

- If a data model is selected in the Browser window when the Database Builder is opened, that model's details will be automatically loaded

Working in the Database Builder

When a data model is loaded, the Database Builder creates a set of logical folders, one for each object type supported by the current DBMS. Each logical folder is populated with all objects of that type found in the data model's hierarchy of Packages (as shown in the Browser window).



In this image the data model 'Orders (postgres)' shows logical folders for Tables, Views, Functions, Sequences, Queries and Connections. It is worth noting there is no folder for 'Procedures' since PostgreSQL does not support database procedures.

Available Actions in the Database Builder Tree

The majority of the Database Builder functions are accessible via context menus. Each object in the Tree has its own set of unique menu items based on its type and status. This table describes the available context menu items and identifies which objects they apply to.

Menu Option	Applies to / Description
New data model	*Applies To:* Blank Space *Description:* Opens the Start Page 'Create from Pattern' tab (Model Wizard).
Refresh All	*Applies to:* Blank Space *Description:* Reloads the complete list of data models.
Load	*Applies to:* Root Node *Description:* Loads the full details of the data model.
Unload	*Applies to:* Root Node *Description:* Unloads the full details of the data model.
Import DB	*Applies to:* Loaded Root Node

Schema	*Description:* Opens the 'Import DB schema' dialog using the current active connection as the Live database source.
Generate DDL	*Applies to:* Loaded Root Node, Folder, Table, View, Procedure, Function, Sequence, Package *Description:* Opens the 'Generate DDL' dialog with the current object(s) selected.
Show Differences	*Applies to:* Loaded Root Node, Folder, Table, View, Procedure, Function, Sequence *Description:* Compares the selected objects to the current active connection.
Show Differences with Options	*Applies to:* Loaded Root Node, Folder, Table, View, Procedure, Function, Sequence, Package *Description:* Compares the selected objects to the current active connection and optionally ignore some of the differences based on the specified compare options.
Manage DBMS Options	*Applies to:* Loaded Root Node *Description:* Opens the 'Manage DBMS Options' dialog, which can be used to change the allocated DBMS and Owner

	of multiple objects.
View Record Count	*Applies to:* Table, View *Description:* Builds and runs a SELECT query (formatted to suit the element's DBMS) to show the number of records in the selected Table or View. If there is no active connection, you are prompted to select one.
View Top 100 Rows	*Applies to:* Table, View *Description:* Builds and runs a SELECT query (formatted to suit the element's DBMS) to show the top 100 rows of the selected Table or View. If there is no active connection, you are prompted to select one.
View Top 1000 Rows	*Applies to:* Table, View *Description:* Builds and runs a SELECT query (formatted to suit the element's DBMS) to show the top 1000 rows of the selected Table or View. If there is no active connection, you are prompted to select one.
View All Rows	*Applies to:* Table, View *Description:* Builds and runs a SELECT

	query (formatted to suit the element's DBMS) to show all rows of the selected Table or View. If there is no active connection, you are prompted to select one.
Properties	*Applies to:* Loaded Root Node, Folder, Table, View, Procedure, Function, Sequence, Package, Connection *Description:* Opens the standard 'Properties' dialog for the selected object.
Find in Project Browser	*Applies to:* Loaded Root Node, Folder, Table, View, Procedure, Function, Sequence, Package, SQL Query, Connection *Description:* Finds the selected object in the Browser window.
Refresh	*Applies to:* Loaded Root Node *Description:* Reloads the details of the current loaded data model. This is necessary when objects are added, changed or deleted by other users or when the changes are performed outside of the Database Builder.
Add new	*Applies to:* Folder, Table, View, Procedure, Function, Sequence, Package,

<type>	SQL Query, Connection *Description:* Creates a new object of the specified type.
Clone <name>	*Applies to:* Folder, Table, View, Procedure, Function, Sequence, Package, SQL Query, Connection *Description:* Makes a new copy of the selected object. When you select this option, a prompt displays on which you set the name and owner of the new object. For Table objects, you can choose which existing constraints should be copied (and set a name for each one) along with which Foreign Keys should be copied. For SQL-based objects, you can make any necessary changes to the SQL for the new element.
Delete <name>	*Applies To:* Table, View, Procedure, Function, Sequence, Package, SQL Query, Connection *Description:* Permanently deletes the selected object from the repository.
Add new Foreign Key on <name>	*Applies to:* Table *Description:* Creates a new relationship between the selected Table and another one, then shows the 'Foreign Key

	Constraint' screen for the new relationship.
SQL Object Properties	*Applies to:* View, Procedure, Function, Sequence *Description:* Opens the 'SQL Object Editor' screen.
Edit	*Applies to:* SQL Query *Description:* Loads the SQL (as defined in the selected element) into the SQL Scratch Pad.
Run	*Applies to:* SQL Query *Description:* Loads the SQL in the SQL Scratch Pad and runs it. If there is no active connection, you are prompted to select one.
Set as active DB Connection	*Applies to:* Connection *Description:* Flags the selected Database Connection as the active one for the current session.
Set as Default DB Connection	*Applies to:* Connection *Description:* Flags the selected Database Connection as the active one each time the data model is loaded.

DB Connection Properties	*Applies to:* Connection *Description:* Opens the 'Database Connection Properties' screen, to manage the connection settings.

Create/Edit/Delete Database Objects

The pages listed in this section describe in detail how to use the Database Builder's interface to create and manipulate database Tables; however, the process of creating and manipulating SQL-based database objects is documented in other areas. See these topics for details:

- Database Views
- Database Procedures
- Database Functions
- Database Sequences
- Database Connections

Database Connections in the Database Builder

When performing certain tasks such as 'Compare' or

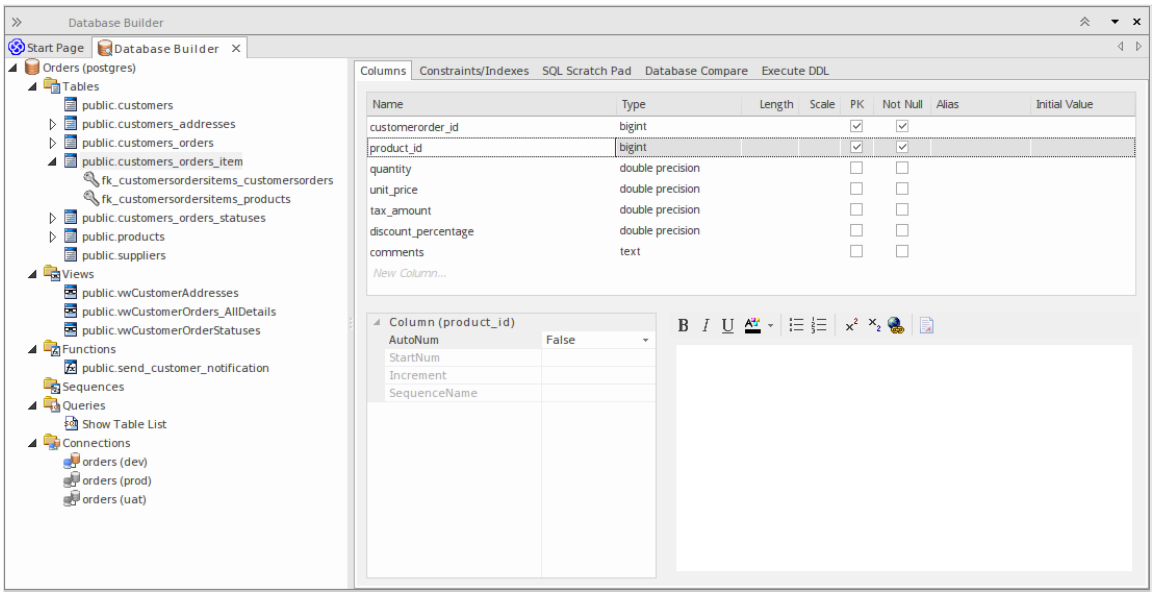
'Execute DDL', the Database Builder requires an active database connection. Only one database connection can be made active (indicated by a colored 'Database Connection' icon, while the others are gray) at a given time. If a database connection is not currently active and you try to perform a task that requires one, the Database Builder performs one of these actions based on how many connections are defined:

- 0 Connections – prompts you to create a connection and, if successful, continues
- 1 Connection – sets it as active and continues
- 2 (or more) Connections – prompts you to select one and, if successful, continues

Columns

Tables are the fundamental database object, and Columns (and their properties) are the most frequently modified Table feature updated and changed by data modelers, therefore the 'Columns' page is conveniently located as the first page of the Database Builder's interface.

Once a Table is selected in the Database Builder's tree, the 'Columns' page is populated with the currently defined list of columns for that Table. The data modeler can then make changes to main column properties directly in the list or grid. As the data modeler selects individual columns in the list, the column's extended properties (and Comments) are shown immediately under the list, allowing modification to these extended properties.



This figure show the Database Builder interface showing the tree of objects and the Columns tab showing the Columns for the selected table.

Notes

- The 'Columns' page will only be populated when a Table item is selected in the Database Builder's tree

Create Database Table Columns

A database Table column is represented in the UML Data Modeling Profile as an attribute with the <<column>> stereotype. For a selected Table, you can review the existing columns and create new columns, on the 'Columns' page of the Database Builder or on the 'Columns and Constraints' screen.

You can define column details directly on the list of columns on the 'Columns' tab. The changes are automatically saved as you complete each field. Some fields have certain restrictions on the data you can enter, as described here. The tab also contains a 'Properties' panel and a 'Notes' field, which are populated with the existing information on the selected column. Each new column that you create is automatically assigned a set of default values and added to the bottom of the list.

Access

Ribbon	Develop > Data Modeling > Database Builder > Click on Table > Columns > Right-Click > Add new Column
Context Menu	In diagram, right-click on required Table | Features | Columns | Right-Click | Add

	new Column
Keyboard Shortcuts	Select a table | F9 | Tab Key (to set input focus on the 'Columns' tab) | Ctrl+N

Create columns in a Table

Option	Action
Name	Overtype the default name with the appropriate column name text.
Type	Click on the drop-down arrow and select the appropriate datatype for the column. The available datatypes depend on the DBMS assigned to the parent Table.
Length	(Optional) Some datatypes have a length component - for example, VARCHAR has a length that defines the number of characters that can be stored. If the datatype does not have a length component, this field is disabled. If the field is available and if you need to define a number of characters, type the

	value here.
Scale	(Optional) Some datatypes have a scale component - for example, DECIMAL has a scale that defines the number of decimal places that can be held. If the datatype does not have a scale component, this field is disabled. If the field is available and if you need to define a scale, type the value here.
PK	Select the checkbox if the column is part of the Primary Key for this Table.
Not Null	Select the checkbox if empty values are forbidden for this column. The checkbox is disabled if the 'PK' checkbox is selected.
Alias	If required for display and documentation purposes, type in an alternative name for the field.
Initial Value	If required, type in a value that can be used as a default value for this column.
Notes	Type in any additional information necessary to document the column. You can format the text using the Notes

	toolbar at the top of the field.

Column Properties

The appropriate properties for the Table's Database Management System automatically display in the 'Property' panel (expand the 'Column (<name>)' branch if they are not visible).

Property	DBMS
Autonum (Startnum Increment)	Oracle MySQL SQL Server DB2 PostgreSQL Notes: If you require an automatic numbering sequence, set this property to True and, if necessary, define the start number and increment.
Generated	DB2 Notes: Set this additional property for auto numbering in DB2, to 'By Default' or 'Always'.

NotForRep	SQLServer Notes: Set this property to True if you want to block replication.
Zerofill	MySQL Notes: Set this property to True or False to indicate if fields are zerofilled or not.
Unsigned	MySQL Notes: Set this property to True or False to indicate whether or not fields accept unsigned numbers.
LengthType	Oracle Notes: Set this property to define the character semantics as 'None', 'Byte' or 'Char'.

Delete Database Table Columns

For a selected database Table, you can review the existing columns and delete any individual column, on the 'Columns' tab of the Columns and Constraints screen.

Access

Use one of the methods outlined here to display a list of columns for a table, then select a column and delete it.

When you select the 'Delete column '<name>" option, if all validation rules are satisfied the column is immediately deleted.

Ribbon	Develop > Data Modeling > Database Builder > Click on Table > Columns > Right-click on column name > Delete column <name>
Context Menu	In diagram, right-click on required Table | Features | Columns | Right-click on column name | Delete column <name>
Keyboard Shortcuts	F9 | Use 'Up Arrow' or 'Down Arrow' to select a column | Ctrl+D

Notes

- If the deleted database Table column is involved in any constraints it will automatically be removed from them

Reorder Database Table Columns

If you have several columns defined in a database Table, you can change the order in which they are listed. The order in the list is the order in which the columns appear in the generated DDL.

Access

Use one of the methods outlined here to display a list of columns for a Table, then select a column and reposition it within the list.

Ribbon	Develop > Data Modeling > Database Builder > Click on Table
Context Menu	In diagram, right-click on required Table | Features | Columns
Keyboard Shortcuts	F9

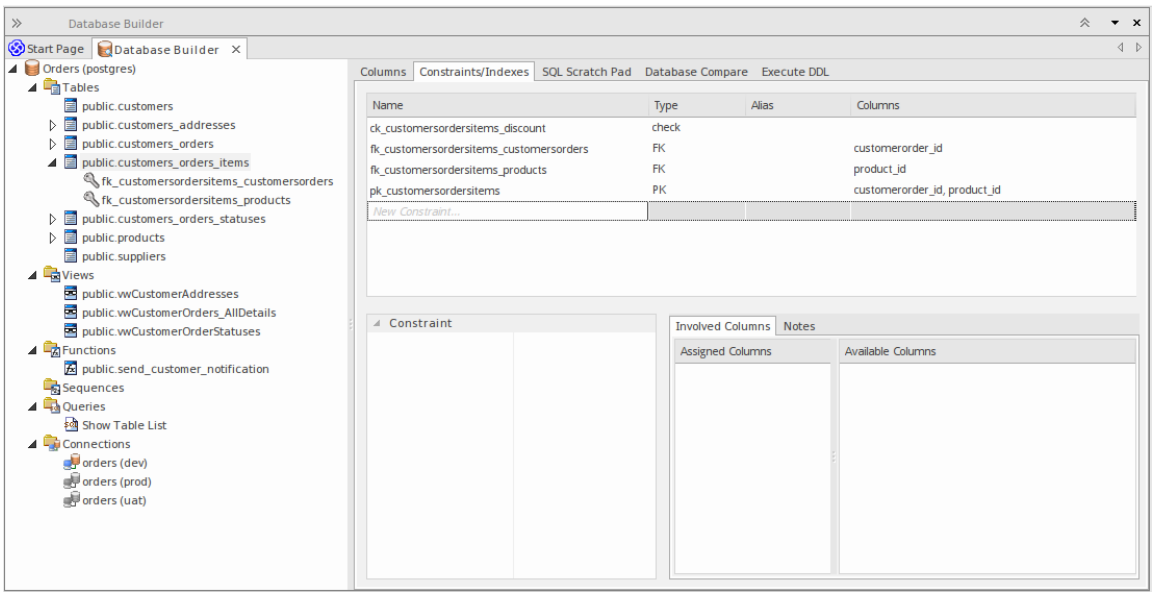
Change the column order

Step	Action
1	In the 'Columns' tab, click on the required column name in the list.
2	Right-click and select the: • 'Move column <name> up' option (or press Ctrl+Up Arrow) to move the column up one position • 'Move column <name> down' option (or press Ctrl+Down Arrow) to move the column down one position These options have an immediate effect both in the 'Columns' tab and on a diagram.

Constraints/Indexes

Tables are the fundamental database object, and Constraints and Indexes (and their properties) are the second most frequently modified Table feature updated and changed by data modelers, therefore the 'Constraints/indexes' page is conveniently located as the second page of the Database Builder's interface.

Once a Table is selected in the Database Builder's tree, the 'Constraints/Indexes' page is populated with the currently defined list of constraints and indexes for the selected Table. The data modeler can then make changes to main properties directly in the list. As the data modeler selects individual constraints or indexes in the list, the constraint's extended properties (and Comments) are shown immediately under the list, allowing modification of these extended properties.



This figure show the Database Builder interface showing the

tree of objects and the Columns tab showing the Columns for the selected table.

Notes

- The 'Constraints/Indexes' page will only be populated when a Table item in the Database Builder's tree is selected

Database Table Constraints/Indexes

Within Enterprise Architect, Table Constraints and Indexes are modeled on the same screen; collectivity they are referred to as Constraints. Database Constraints define the conditions imposed on the behavior of a database Table. They include:

- Primary Key - uniquely identifies a record in a Table, consisting of one or more columns
- Index - improves the performance of retrieval and sort operations on Table data
- Unique Constraints - a combination of values that uniquely identify a row in the Table
- Foreign Key - a column (or collection of columns) that enforce a relationship between two Tables
- Check Constraints - enforces domain integrity by limiting the values that are accepted by a column
- Table Trigger - SQL or code automatically executed as a result of data in a Table being modified

In Enterprise Architect, you can define and maintain Table Constraints using either the purpose-designed 'Constraints/Indexes' page of the Database Builder or the Columns and Constraints screen.

Access

Ribbon	Develop > Data Modeling > Database Builder > Click on Table name > Constraints/Indexes | Right-click | Add New Constraint
Context Menu	In diagram | Right-click on Table | Features | Constraints/Indexes | Right-click | Add New Constraint
Keyboard Shortcuts	Click on Table: F9 > Constraints/Indexes: Ctrl+N

Create a Constraint

The process of creating any of these constraint types is the same and is achieved in one of the ways described here.

Create a Constraint - Using the context menu or keyboard

Step	Action

1	A new constraint is automatically created and assigned the default name *constraint n* (where *n* is a counter) and a 'Type' of 'index'. Overtype the default name with your own constraint name.
2	If necessary, in the 'Type' field click on the drop-down arrow and select the appropriate constraint type.
3	If you prefer, type an alias for the constraint, in the 'Alias' field. The 'Columns' field is read-only; it is populated with the columns that you assign to the 'Involved Columns' tab.

Create a Constraint - Overtype the template text

Step	Action
1	On the 'Constraints/Indexes' tab for the selected Table, the list of constraints ends with the template text *New Constraint*.

	Overtype this text with the appropriate constraint name, and press the Enter key.
2	The new constraint is automatically created and assigned the default Type of index. If necessary, in the 'Type' field click on the drop-down arrow and select the appropriate constraint type.
3	If you prefer, type an alias for the constraint, in the 'Alias' field. The 'Columns' field is read-only; it is populated with the columns that you assign to the 'Involved Columns' tab.

Assign Columns to a Constraint

The constraint types of Primary Key, Foreign Key, Index and Unique all must have at least one column assigned to them; this defines the columns that are involved in the constraint.

Step	Action
1	On the 'Constraints/Indexes' tab for the selected Table, click on the constraint to which you are

	assigning columns.
2	The 'Available Columns' panel lists all columns defined for the Table. For each column to assign to the constraint, right-click on the column name and select 'Assign column <name>'. The column name is transferred to the 'Assigned Columns' list.

Unassign Columns from a Constraint

Step	Action
1	On the 'Constraints/Indexes' tab for the selected Table, click on the constraint from which you are unassigning columns.
2	In the 'Assigned Columns' list, right-click on the name of the column to unassign from the constraint and select 'Unassign column <name>'. The column name is transferred to the 'Available Columns' list.

Reorder the Assigned Columns in a Constraint

If you have a number of columns in the constraint, you can re-arrange the sequence by moving a selected column name one place up or down the list at a time. To do this:

- Right-click on the column name to move and select either:
  - Move column '<name>' up (Ctrl+Up Arrow) or
  - Move column '<name>' down (Ctrl+Down Arrow)

Delete a constraint

To delete a constraint you no longer require, right-click on the constraint name in the list on the 'Constraints/Indexes' tab and select the 'Delete constraint <name>' option. If all validation rules for the given constraint type are met, the constraint is immediately removed from the repository along with all related relationships (if there are any).

Primary Keys

A Primary Key is a column (or set of columns) that uniquely identifies each record in a Table. A Table can have only one Primary Key. Some DBMSs support additional properties of Primary Keys, such as Clustered or Fill Factor.

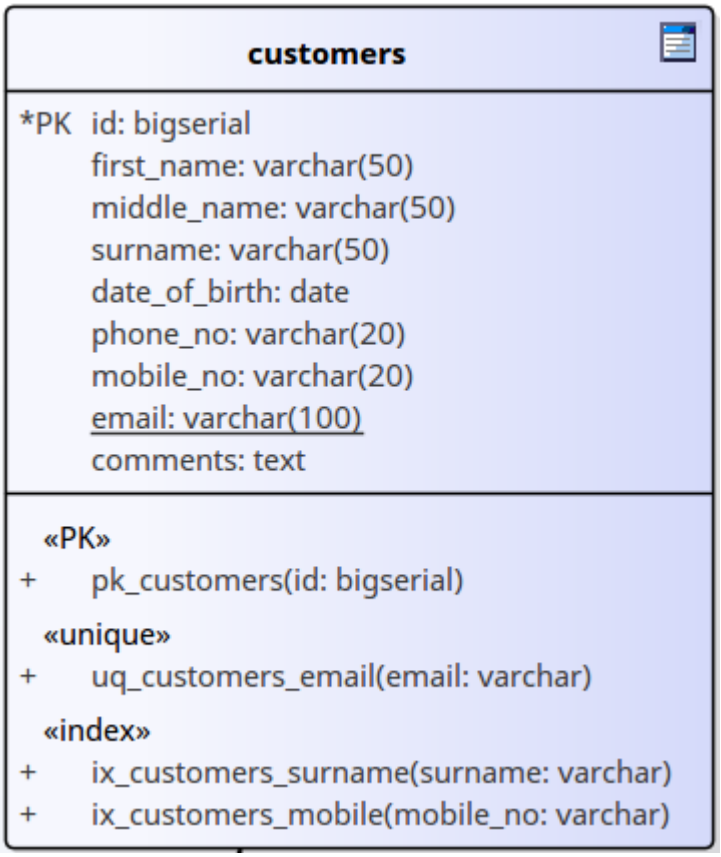
Access

Ribbon	Develop > Data Modeling > Database Builder > Click on Table name
Context Menu	In diagram | Right-click on Table | Features | Constraints/Indexes

Create a Primary Key

In Enterprise Architect you can create a Primary Key from either the 'Columns' tab or the 'Constraints/Indexes' tab. In either case, when you add a column to a Primary Key constraint, the column is automatically set to be 'Not Null'. Additionally any diagram (assuming the 'Show Qualifiers and Visibility Indicators' option is set) containing the Table element will show the 'PK' prefix against the column name.

In this image, see the first column 'id: bigserial'.



Create a Primary Key - from the Columns tab

Step	Action
1	Either: • In the Database Builder, click on a Table with one or more defined columns, and click on the 'Columns' tab, or • On a diagram, click on a Table and press F9 to display the 'Columns' tab

2	For each column to include in the Primary Key, select the 'PK' checkbox. If a Primary Key constraint is not previously defined for the current Table, the system will create a new constraint using the Primary Key Name template.

Create a Primary Key - from the Constraints tab

Step	Action
1	Either: • In the Database Builder, click on a Table with one or more defined columns, and click on the 'Constraints/Indexes' tab, or • On a diagram, click on a Table and press F10 to display the 'Constraints/Indexes' tab
2	Overtype the *New Constraint* text with the Primary Key name, press the Enter key and click on the 'Type' field drop-down arrow, and select 'PK'.
3	Assign the required columns to the PK constraint.

4	Set the Primary Key's extended properties using the property panel. • Fill Factor is a numeric value between 0 and 100 • Is Clustered is a Boolean value that determines the physical order of how the data is stored; for most DBMSs the Is Clustered property defaults to True for Primary Keys

Remove columns from a Primary Key

You can remove columns from a Primary Key using either the 'Columns' tab or the 'Constraints/Indexes' tab.

Remove columns from a Primary Key - using the Columns tab

Step	Action
1	Either: • In the Database Builder, click on the Table with the Primary Key, and click on the 'Columns' tab, or

	• On a diagram, click on a Table and press F9 to display the 'Columns' tab
2	Against each column you want to remove from the Primary Key, deselect the 'PK' checkbox. If you have removed all columns from the Primary Key constraint and the Primary Key is no longer needed, it must be manually deleted.

Remove columns from a Primary Key - using the Constraints/Indexes tab

Step	Action
1	Either: • In the Database Builder, click on the Table with the Primary Key, and click on the 'Constraints/Indexes' tab, or • On a diagram, click on a Table and press F10 to display the 'Constraints/Indexes' tab
2	Unassign the columns on the PK constraint, as necessary.

Notes

- Warning: Enterprise Architect assumes that Primary Key constraints have at least one column assigned to them; however, Enterprise Architect does not enforce this rule during modeling
  If DDL is generated for a Table whose Primary Key has no column assigned, that DDL will be invalid

Database Indexes

Database indexes are applied to Tables to improve the performance of data retrieval and sort operations. Multiple indexes can be defined against a Table; however, each index imposes overheads (in the form of processing time and storage) on the database server to maintain them as information is added to and deleted from the Table

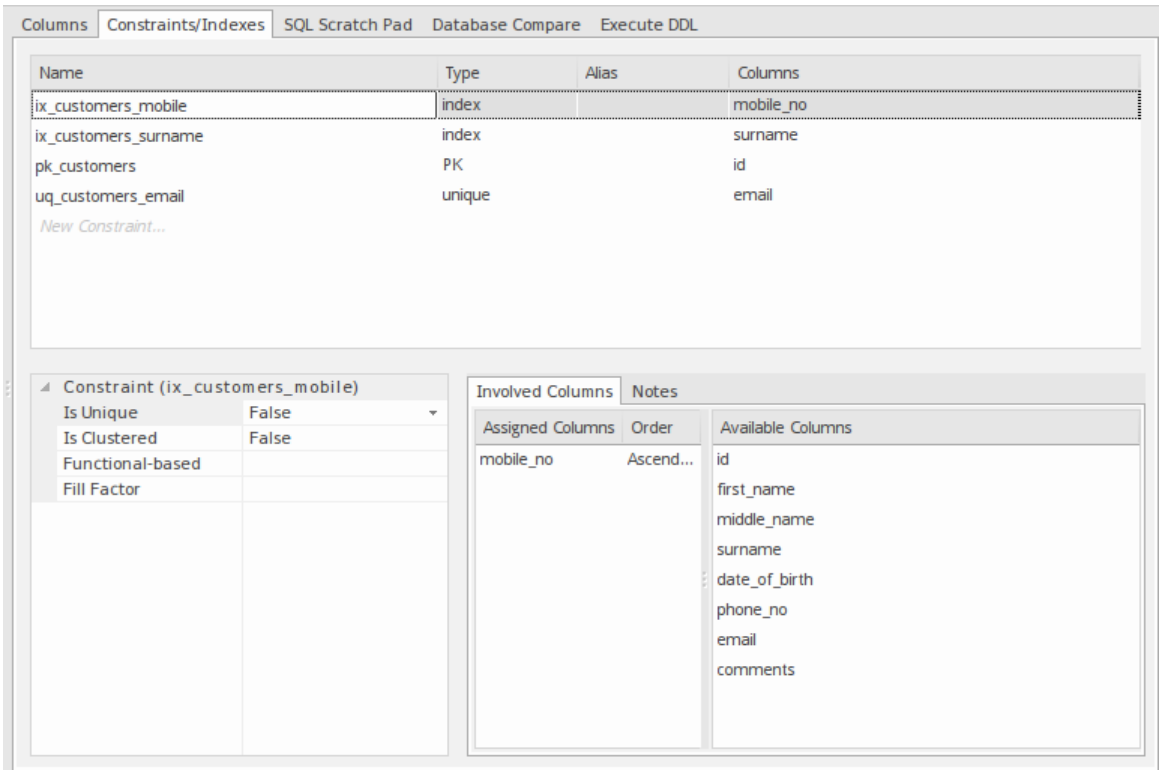
In Enterprise Architect an index is modeled as a stereotyped operation.

Some DBMSs support special types of index; Enterprise Architect defines these using additional properties such as function-based, clustered and fill-factor.

Access

Ribbon	Develop > Data Modeling > Database Builder > Click on Table name > Constraints/Indexes
Context Menu	In diagram | Right-click on Table | Features | Constraints/Indexes
Keyboard Shortcuts	Click on Table: F9 > Constraints/Indexes

Work on an index



Step	Action
1	On the 'Constraints/Indexes' tab for the Table, right-click and select 'Add new constraint'. The new constraint is added with the default name 'constraint1' and the Type of 'index'. Overtype the name with your preferred index name.
2	Assign the appropriate columns to the Index. The 'Assigned Columns' list has an additional 'Order' field that specifies the order (Ascending or

	Descending) in which each assigned column is stored in the index. You can toggle the order for each column, as required. Additionally, for MySQL indexes, a 'Len' field will be visible in which you can define Partial Indexes; that is, an index that uses the leading 'n' number of characters of a text based field. The 'Len' field takes only whole number numeric values of between 0 and the column's defined length. A value of 0 (which is the default) indicates that the entire column is to be indexed.
3	In the 'Property' panel, review the settings of the extended properties that are defined for the current DBMS.

Additional Properties

Property	Description
Is Unique	(True / False) indicates whether the current index is a 'Unique Index'. A Unique Index ensures that the indexed column (or columns) does not contain duplicate values, thereby ensuring that each row has a unique value (or

	combination of values when the index consists of multiple columns).
Is Clustered	(True / False) indicates whether the current index is a 'Clustered Index'. With a clustered index, the rows of the table are physically stored in the same order as in the index, therefore there can be only one clustered index per table. By default a table's Primary Key is clustered. Not all DBMS's support clustered indexes, therefore the 'Is Clustered' Index property will only be visible for DBMSs that support it.
Is Bitmap	(True / False) indicates whether the current index is a 'Bitmap' index. Bitmap indexes are meant to be used on columns that have relatively few unique values (referred to as 'low cardinality' columns) and that physically consist of a bit array (commonly called bitmaps) for each unique value. Each of the arrays will have a bit for each row in the table. Consider this example: a bitmap index is created on a column called 'Gender', which has the options 'Male' or 'Female'. Physically, the index will consist of two bit arrays, one for 'Male' and one for

	'Female'. The female bit array will have a 1 in each bit where the matching row has the value 'Female'. The 'Is Bitmap' and 'Is Unique' properties are mutually exclusive, and so the DDL generation will ignore the 'Is Unique' property when the 'Is Bitmap' property is True. Bitmap Indexes are only supported by Oracle; therefore, this property is only visible while modeling Oracle indexes.
Fill Factor	A numeric value between 0 and 100, that defines the percentage of available space that should be used for data. Not all DBMSs support fill factor, therefore the 'Fill Factor' index property will only be visible for DBMSs that support it.
Functional-based	A SQL statement that defines the function/statement that will be evaluated and the results indexed; for example: LOWER("field") Not all DBMSs support functional-based indexes, therefore the 'Functional-based' Index property will only be visible for DBMSs that support them, such as PostgreSQL and Oracle.

Include	Identifies a comma-separated list (CSV) of non-key Columns from the current table. Not all DBMSs support the 'Include' property on indexes, therefore this property will only be visible for DBMSs that support it.

Notes

- Warning: Enterprise Architect assumes that Indexes have at least one column assigned to them; however, Enterprise Architect does not enforce this rule during modeling
  If DDL is generated for a Table that has an Index defined without column(s) assigned, that DDL will be invalid, unless the index is functional-based
- Any columns assigned to a functional-based index are ignored

Unique Constraints

Unique Constraints enforce the 'uniqueness' of a set of fields in all rows of a Table, which means that no two rows in a Table can have the same values in the fields of a Unique Constraint. Unique Constraints are similar to Primary Keys (in that they also enforce 'uniqueness') but the main difference is that a Table can have multiple Unique Constraints defined but only one Primary Key.

Access

Ribbon	Develop > Data Modeling > Database Builder > Click on Table name > Constraints/Indexes > Right-click > Add New Constraint
Context Menu	In diagram or Browser window | Right-click on Table element | Features | Constraints/Indexes
Keyboard Shortcuts	Click on Table: F9 > Constraints/Indexes: Ctrl+N

Create a Constraint

Step	Action
1	On the 'Constraints/Indexes' tab, a new constraint is automatically created and assigned the default constraint name and a 'Type' of index. Overtype the constraint name with a name that identifies this as a unique constraint.
2	In the 'Type' field, change the value from 'index' to 'unique'.

Notes

- Warning: Enterprise Architect assumes that Unique Constraints have at least one column assigned to them; however, Enterprise Architect does not enforce this rule during modeling
  If DDL is generated for a Table that has a unique constraint defined without column(s) assigned, that DDL will be invalid

Foreign Keys

A Foreign Key defines a column (or a collection of columns) that enforces a relationship between two Tables. It is the responsibility of the database server to enforce this relationship to ensure data integrity. The model definition of a Foreign Key consists of a parent (primary) Table containing a unique set of data that is then referred to in a child (foreign) Table.

In Enterprise Architect, a Foreign Key is modeled with two different (but related) UML components:

- A Foreign Key constraint (a UML operation with the stereotype of <<FK>>) stored on the child Table and
- An Association connector (stereotype of <<FK>>) defining the relationship between the two Tables

Create a Foreign Key

Although the definition of a Foreign Key can be complex, the Foreign Key Constraint screen simplifies the modeling of Foreign Keys. This screen is purpose-designed to help you select which constraint in the parent Table to use, and will automatically match the child Table columns to those in the parent Table that are part of the constraint. Different aspects of the process of developing a Foreign Key are described here separately for illustration, but the overall process should be a smooth transition.

A number of conditions must be met before a Foreign Key definition can be saved:

- Both Tables must have matching DBMSs defined
- The parent Table must have at least one column
- The parent Table must have a Primary Key, unique constraint or unique index defined

Create a Foreign Key - using the Database Builder

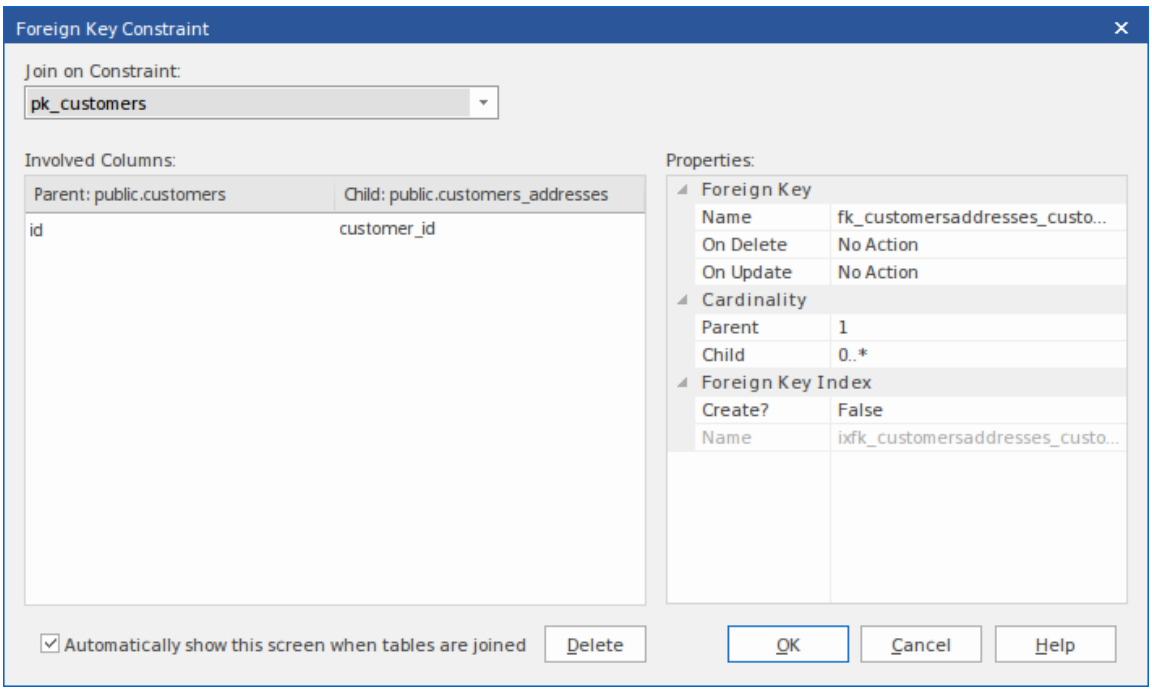
Step	Action
1	In the Database Builder tree, right-click on the child Table name and click on 'Add new Foreign Key on <table name>'. A dialog displays listing all the possible parent Tables.
2	Double-click on the required parent Table name in the list or select it and click on the OK button. The 'Foreign Key Constraint' screen displays.

Create a Foreign Key - using a relationship on a diagram

Step	Action
1	In the Data Modeling diagram, locate the required child (Foreign Key) Table and parent (Primary Key) Table.
2	Select an Association connector in the 'Data Modeling' page of the Diagram Toolbox.
3	Click on the child Table and draw the connector to the parent Table.
4	If the Foreign Key Constraint screen has been set to display automatically when two Tables are joined, it displays now. Otherwise, either: • Double-click on the connector or • Right-click on the connector and select the 'Foreign Keys' option The Foreign Key Constraint screen displays.

The Foreign Key Constraint Screen

As an example this image shows the Foreign Key Constraint screen loaded with the details of 'fk_customersaddresses_customers' (as defined in the Example model).



Option	Action
Join on Constraint	This combo box lists all defined constraints in the parent Table that could be used as the basis of a Foreign Key. (These constraints can be Primary Keys, Unique Constraints or Unique Indexes.) The first constraint in the list is selected by default; if this is not the constraint you want, select the correct constraint from the combo box. When you select the constraint, its columns are automatically listed in the

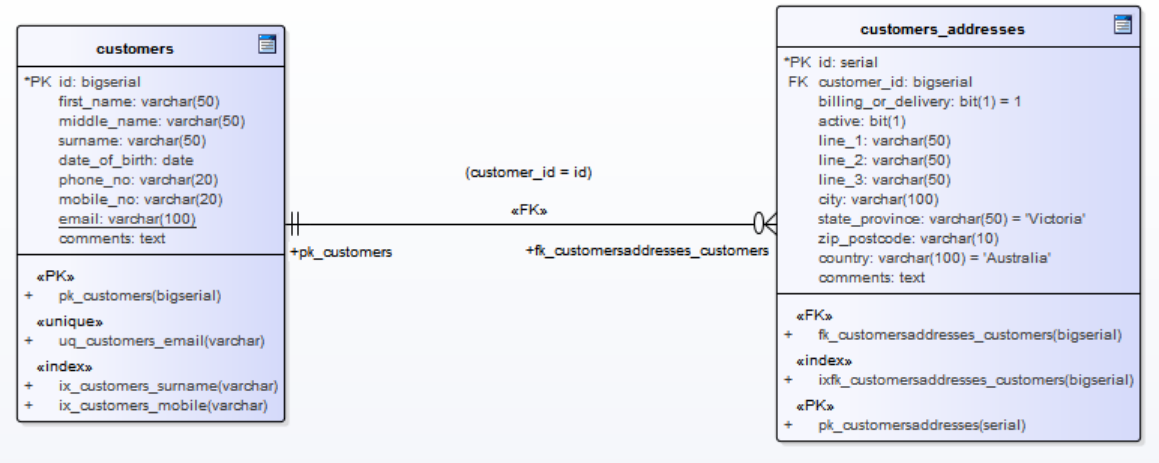
	'Involved Columns' panel, under the 'Parent: <tablename>' column.
Involved Columns	This list is divided into two: the columns involved in the selected constraint are listed on the left, and the child columns that are going to be paired to the parent columns are listed on the right. When a constraint is selected (in the 'Join on constraint' field) the parent side is refreshed to display all columns assigned to the selected constraint. On the child side the system will automatically attempt to match each parent column to one of the same name in the child Table. If the child Table does not have a column of the same name, a new column of that name will be added to the list, flagged with (*) to indicate that a new column will be created in the Table. However, if you want to force the pairing to an existing child Table column or a new column with a different name, click on the column name field and either: • Type in the replacement name, or • Select an existing column (click on the drop-down arrow and select the name from the list)

Name	This field defines the name of the Foreign Key constraint, and defaults to a name constructed by the Foreign Key Name Template. To change the name to something other than the default, simply overtype the value.
On Delete	Select the action that should be taken on the data in the child Table when data in the parent is deleted, so as to maintain referential integrity.
On Update	Select the action that should be taken on the data in the child Table when data in the parent is updated, so as to maintain referential integrity.
Parent	Click on the drop-down arrow and select the cardinality of the parent Table in the Foreign Key.
Child	Click on the drop-down arrow and select the cardinality of the child Table in the Foreign Key.
Create?	If you want to create a Foreign Key Index at the same time as the Foreign Key, set this property to True.

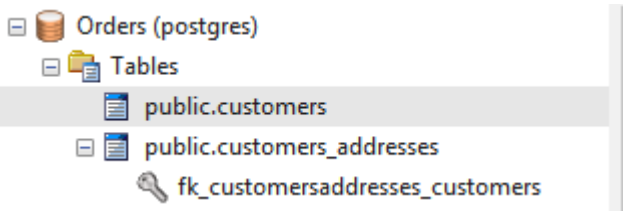
	The name of the Foreign Key Index is controlled by the Foreign Key Index template, and the generated name is shown in the 'Name' field underneath the 'Create?' field.
Automatically show this screen when tables are joined	(For diagrammatic modeling) Select this checkbox to automatically display this screen whenever an Association is created between two Tables.
Delete	Click on this button to delete the currently selected existing (saved) Foreign Key. A prompt is displayed to confirm the deletion (and the deletion of the Foreign Key Index, if one exists) - click on the Yes button. Deleting a Foreign Key leaves an Association connector in place, which you can either edit or delete (right-click and select 'Delete association: to <Table name>').
OK	Click on this button to save the Foreign Key.

Examples

This example shows simple Foreign Keys in a diagram:



The same Foreign Key will be shown in the Database Builder's tree as a child node under the Table 'customers.addresses'.



Check Constraints

A Check Constraint enforces domain integrity by limiting the values that are accepted by a column.

Access

Ribbon	Develop > Data Modeling > Database Builder > Click on Table name > Constraints/Indexes > Right-click > Add New Constraint
Context Menu	In diagram | Right-click on Table | Features | Constraints/Indexes | Right-click | Add New Constraint
Keyboard Shortcuts	Click on Table: F9 > Constraints/Indexes: Ctrl+N

Create a Constraint

Step	Action

1	On the 'Constraints/Indexes' tab of the Columns and Constraints screen, a new constraint is automatically created and assigned the default constraint name and a 'Type' of index. Overtype the constraint name with a name that identifies the constraint as a check constraint, such as 'CHK_ColumnName' (the CHK_ prefix is optional).
2	In the 'Type' field, change the value from 'index' to 'check'.
3	In the 'Properties' panel for the Condition property, type the SQL statement that will be used as the Check Condition; for example, column1 < 1000. If the condition is long, click on the ... button to display a SQL editor (with syntax highlighting).

Delete a Check Constraint

If you do not want to keep a check constraint, either:

- Right-click on it in the list and select 'Delete constraint <name>', or
- Click on the item and press Ctrl+D

The constraint is immediately deleted.

Notes

- Any columns assigned to a check constraint are ignored

Table Triggers

A Table trigger is SQL or code that is automatically executed as a result of data being modified in a database Table. Triggers are highly customizable and can be used in many different ways; for example, they could be used to stop certain database activities from being performed during business hours, or to provide validation or perform deletions in secondary Tables when a record in the primary Table is deleted.

In Enterprise Architect, a Table trigger is modeled as a stereotyped operation and managed using the Table's 'Constraints' screen.

Access

Ribbon	Develop > Data Modeling > Database Builder > Click on Table name > Constraints/Indexes | Right-click | Add New Constraint
Context Menu	In diagram | Right-click on Table | Features | Constraints/Indexes | Right-click | Add New Constraint
Keyboard	Click on Table: F9 > Constraints/Indexes:

Shortcuts	Ctrl+N

Create a Table Trigger

Step	Action
1	On the 'Constraints/Indexes' tab, a new constraint is automatically created and assigned the default constraint name and a 'Type' of index. Overtype the constraint name with a name that identifies the constraint as a trigger, such as TRG_OnCustomerUpdate. (The TRG_ prefix is optional.)
2	In the 'Type' field, change the value from 'index' to 'trigger'.
3	In the 'Properties' panel for the Statement property, type in the complete SQL statement (including CREATE TRIGGER) that will define the Trigger. If the condition is long, click on the ... button to display a SQL editor (with syntax highlighting).
4	The properties Trigger Time and Trigger Event are currently information-only values and are not used in

	DDL generation.

Delete a Table Trigger

If you do not want to keep a trigger, either:

- Right-click on it in the list and select 'Delete constraint <name>', or
- Click on the item and press Ctrl+D

The trigger is immediately deleted.

Notes

- Any columns assigned to table triggers are ignored

SQL Scratch Pad

The SQL Scratch Pad provides a mechanism to develop and run ad-hoc SQL Queries against a live database. While you develop your data model you might want to execute and test ad-hoc SQL Queries for a DDL script, or run enquiries on the live database; all of this is possible within the Enterprise Architect Database Builder interface.

The SQL Scratch Pad requires the Database Builder to have a valid connection to a live database. This database connection is shared between the 'SQL Scratch Pad', 'Database Compare' and 'Execute DDL' tabs of the Database Builder.

The Scratch Pad consists of:

- A toolbar providing facilities for importing, saving, executing and clearing the SQL Queries
- An editor panel in which you create or import the SQL Queries - this panel provides SQL-based syntax highlighting for the current data model
- A tabbed panel consisting of two pages, one to show the results of executing the Query and one to display any messages generated during the execution

Access

Open the Database Builder window, then display the 'SQL Scratch Pad' tab.

Ribbon	Develop > Data Modeling > Database Builder > SQL Scratch Pad

The Scratch Pad Toolbar

The functionality of each button on the Scratch Pad Toolbar is described in this table, working from left to right.

Button	Action
Run SQL	Executes the SQL Query currently shown in the Scratch Pad. Check the 'Results' and 'Messages' tabs for the output of executing the Query.
New	Clears the SQL Query editor fields so that you can enter a new query.
Open	Loads an SQL Query from file. A source file browser displays, defaulted to display SQL files. Click on the file name and on the Open button to display the file contents in the Scratch Pad.
Save to SQL	Saves this SQL statement to the SQL

Query	Query object it came from.
Save to New SQL Query	Creates a new SQL Query object and saves this statement to that object.
Save to File	Saves the currently-displayed Query to the file it came from. If you created the Query from scratch, a source file browser displays in which you type the new file name and click on the Save button to save the Query.
Save to New File	Saves the currently-displayed Query to a new .sql file. A source file browser displays on which you type in the new file name and click on the Save button to save the Query.
Clear	Clears the contents of the Scratch Pad. Any Query displayed in the Scratch Pad remains there until you either replace it with another Query from file or you close the model.
Toggle Comment	Applies the SQL comment characters '--' to the beginning of each selected line or, if the selected lines are already commented, removes the comment characters. Alternatively, press

	Ctrl+Shift+C.
Statement Separator	Type in the character(s) to use to mark the end of each statement.
Help	Displays the Help on the SQL Query Scratch Pad.
Query Description	Displays a label providing a description of the current SQL, whether there are pending changes (indicated by a leading *), and the name of the loaded SQL Query object or Filename.

Notes

- The SQL Scratch Pad does not manipulate your SQL in any way, so you must use the correct syntax for the current DBMS
- While the SQL Scratch Pad can execute multiple SQL statements, and the status and messages of each statement are shown in the 'Messages' list, only the results of one SELECT statement can be shown in the 'Results' list at a time; all subsequent SELECT statements will be ignored

Database Compare

The 'Database Compare' tab provides a mechanism for comparing the current data model with a live database, and optionally synchronizing any differences in either direction. Differences 'pushed' into a live database are performed using 'Alter DDL' statements, while changes imported from the live database can be directly 'pulled' into the model.

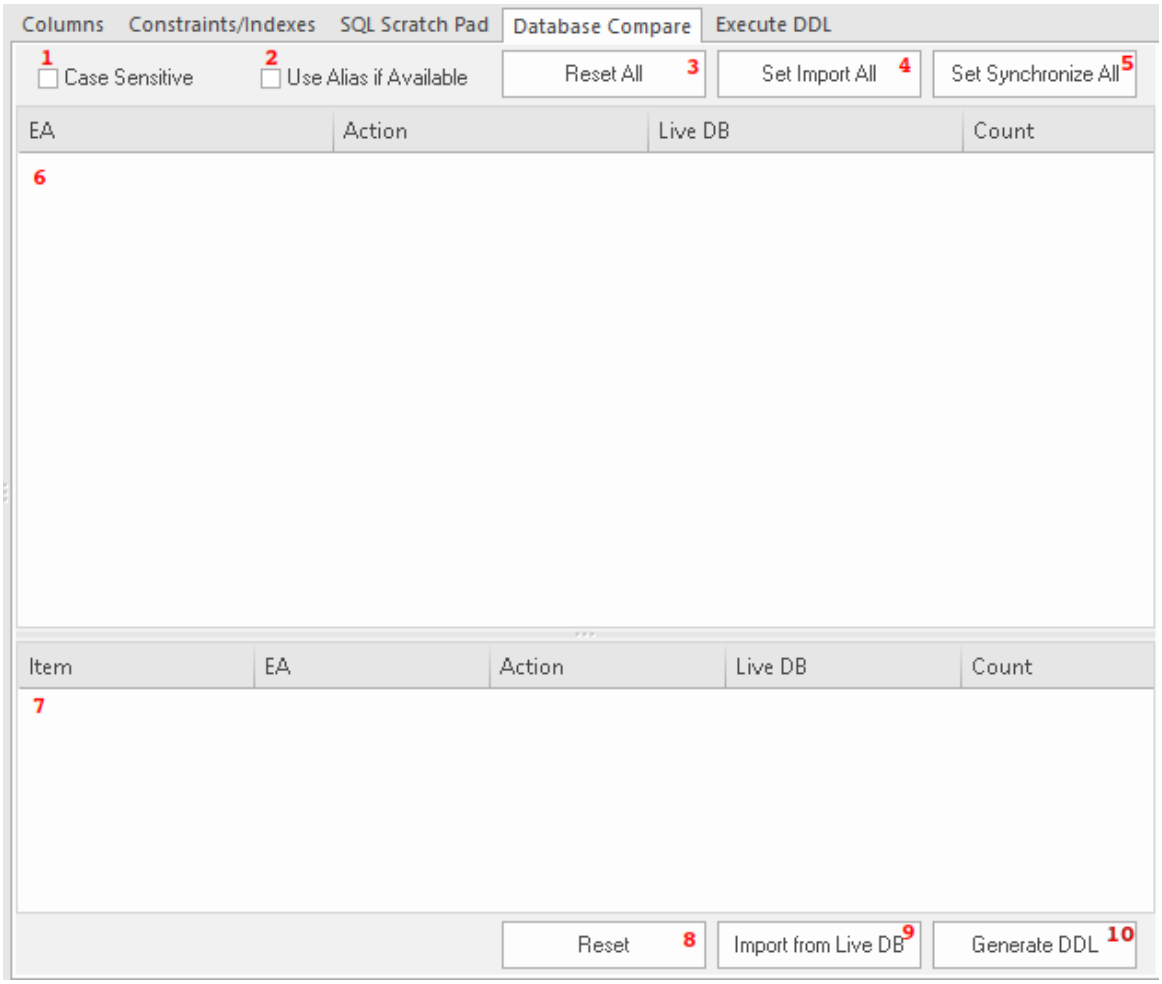
The Database Compare functionality requires the Database Builder to have a valid connection to a live database. This database connection is shared by the 'SQL Scratch Pad', 'Database Compare' and 'Execute DDL' tabs of the Database Builder.

Access

Open the Database Builder window, then display the 'Database Compare' tab.

Ribbon	Develop > Data Modeling > Database Builder > Database Compare

The DDL Compare Tab



The 'Database Compare' tab has a number of controls, as described here.

Number & Name	Description
1 Case Sensitive	Click on this checkbox to make all comparisons of properties recognize differences in letter-case in the property text.
2 Use Alias if Available	Click on this checkbox to indicate that any defined aliases should be used instead of object names (at both object

	and column level).
3 Reset All	Click on this button to set the 'Action' flag for all objects back to the default value.
4 Set Import All	Click on this button to set the 'Action' flag of all detected differences to <====; that is, update the model with the value(s) from the live database.
5 Set Synchronize All	Click on this button to set the 'Action' flag of all detected differences to ====>; that is, update the live database with the value(s) from the model.
6 Differences	Review the list of objects found to have mis-matches between the model and the live database. Selecting an item in this list will populate the 'Components' list. (See the *Differences List* table for a detailed description of each column.)
7 Components	Review this list of properties of the selected object that differ between the model and the live database. (See the *Component List* table for a detailed description of each column.)

8 Reset	Click on this button to set the 'Action' flag for all properties of the current object back to the default value.
9 Import from Live DB	Click on this button to import all properties' values (with the 'Action' of <===) from the live database into the model.
10 Generate DDL	Click on this button to generate the 'Alter DDL' statements for all objects with an 'Action' of ====>, and send the statements to the 'Execute DDL' tab.

Differences List

Column	Description
EA	Displays the name of each object in the model that has one or more detected differences. Blank values indicate that the object is missing in the model but exists in the live database.
Action	Defaults to 'No Action' as the action to take considering this object's

	difference(s). Click on the drop-down arrow and select a specific action. The list of available actions in the list will depend on whether or not the given object is paired in the model and live database. **Paired objects** • No Action - do not update the database or model with this change • ===> - update the object in the database from the model • <=== - update the object in the model from the database • Customize - set the items to No Action prior to setting different actions on each item in the lower panel • Unpair - separate the paired objects so that they are not compared with each other or updated from each other **Unpaired objects** • Create <object name> - create the missing database object in the database or model, as appropriate • Delete <object name> - delete the object from the model • Drop <object name> - delete the object from the database • Pair with <object name> - pair the

	object in the database with the named (unpaired) object in the model, so that they are compared for differences between them The 'Action' fields in the 'Components List' (the lower panel) will be updated based on the selection of this field. For example, if the live database has a Table column 'Address1' and the model doesn't, setting the object 'Action' to '===>' (update the object in the database from the model) sets the column 'Item Action' to 'Drop Address1', which will remove the column from the live database.
Live DB	Shows the name of each object in the live database that has one or more detected differences. Blank values indicate that the object exists in the model but is missing in the live database.
Count	Shows the total number of detected differences for the object (and all of its components) between the model and live database.

Component List

Column	Description
Item	Shows the component name or description for each detected difference. The differences are grouped into three categories: Properties, Columns and Constraints, in a tree structure.
EA	Shows the value of the given component as detected in the model. Blank values indicate that the value is missing in the model but exists in the live database.
Action	Defaults to the action corresponding to the setting of the object 'Action' field in the 'Differences' list, to indicate the action to take regarding the difference detected for the component. Click on the drop-down arrow to select an alternative action; the available options in the list depend on the component's type and the detected difference. • No Action - do not update the database or model • ===> - update the object in the live database from the model

	• <=== - update the object in the model from the live database • Add <item name> - create the missing item in the database or model, as appropriate • Delete <item name> - delete the item from the model • Drop <item name> - delete the item from the live database
Live DB	Shows the value for the selected component in the live database. Blank values indicate that the value exists in the model but is missing in the live database.
Count	Shows the number of differences between the model and the live database detected in the selected component.

Working with the Database Comparison

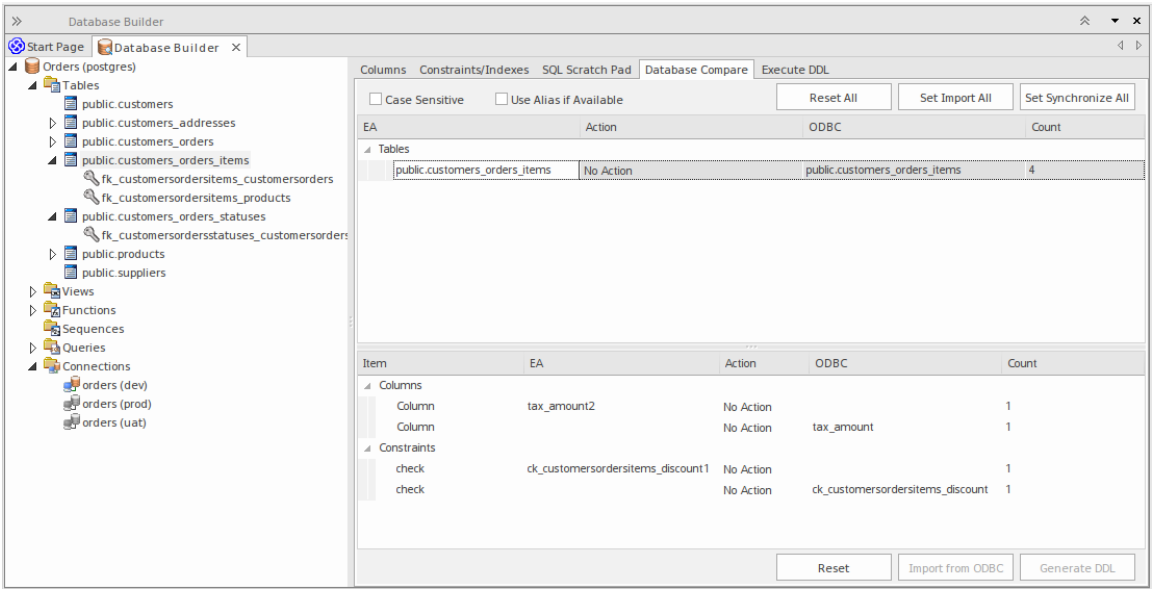
Whenever you perform a comparison, Enterprise Architect reads the definition from both the live database and the model, and then attempts to 'pair' each object from one source with the other, using its name (and schema, if relevant for the current DBMS).

If a match is found, the object name is shown in both the 'EA' and 'Live DB' columns with a default action of 'No Action'. The 'Count' column indicates the total number of differences found for the object and its components or properties.

If a match is not found between the systems, the object name is shown in the source column (either 'EA' or 'Live DB') while the other column is blank. In this state it is possible to pair the object with an object of a different name; the 'Action' dropdown list will present the available objects. If a new pairing is made the two objects' definitions are compared for differences and the results are shown in the 'Components' list, with the default action of '====>' selected.

If you select an action at the object level, this will set the matching action for all of the object's components and properties. However, if you select the 'Customize' action at the object level, you can determine a different action for each component.

As an example, both a column (tax_amount) and constraint (ck_customersordersitems_discount) were renamed in Table 'public.customers_order_items' (in the Example model) and a database compare performed; this image shows the differences found:



In the image there is only one Table that had detected differences - 'public.customers_order_items'; selecting this populates the 'Components' list. From the detected results it can be determined that the data model contains a column (tax_amount2) and a check constraint (ck_customerordersitems_discount1) that the live database doesn't and in turn the live database contains a column (tax_amount) and a check constraint (ck_customerordersitems_discount) that the data model doesn't.

Comparing with Options

The 'Compare with Options' functionality works in the same manner as for a direct comparison, except that you are prompted to choose which object/property comparisons should be performed. This enables you to ignore particular

differences that are not of relevance at the current time.

These tables describe the different comparisons that can be enabled or disabled.

All Objects, Owner

Comparison	Action
Owner	Select to indicate that the 'Owner' property of all database objects should be compared, after the objects have been 'paired'.

Table Options

Option	Action
Tables	Select this parent option to enable all of the Table comparison options. Deselect to disable all the other options. You would then deselect or select specific options in the list.
Table -	Select to indicate that extended properties

Extended Properties	of Tables (such as DB Version and Tablespace) should be compared.
Table - Remarks	Select to indicate that remarks applied to Tables should be compared.
Columns	Select this parent option to enable all of the 'Column comparison' options. Deselect to disable all the other 'Column' options. You would then deselect or select specific options in the list.
Column - Type	Select to indicate that the datatype name for the Table Columns should be compared.
Column - Size	Select to indicate that the datatype size for the Table Columns should be compared.
Column - Default Value	Select to indicate that the default values of the Table Columns should be compared.
Column - Position	Select to indicate that the Table Column positions should be compared.
Column - Not Null	Select to indicate that the not null property of the Table Columns should be

	compared.
Column - Auto Numbering	Select to indicate that the autonumbering properties for the Table Columns should be compared (such as AutoNum, StartNum and Increment).
Column - Unmatched Columns	Select to indicate that Table Columns that are unmatched between the model and the live database should be compared. Typically these are columns that exist in one system but do not exist in the other.
Column - Extended Properties	Select to indicate that extended properties of Table Columns (such as Unsigned and Zerofill) should be compared.
Column - Remarks	Select to indicate that remarks applied to Table Columns should be compared.
Constraints	Select this parent option to enable all of the 'Table Constraint comparison' options. Deselect to disable all the 'Table Constraint' options. You would then deselect or select specific options in the list.
Constraint - Primary Keys	Select to indicate that properties related to Primary Keys should be compared.

Constraint - Foreign Keys	Select to indicate that properties related to Foreign Keys should be compared.
Constraint - Indexes	Select to indicate that properties related to Indexes should be compared.
Constraint - Unique Constraints	Select to indicate that properties related to Unique Constraints should be compared.
Constraint - Check Constraints	Select to indicate that properties related to Check Constraints should be compared.
Constraint - Table Triggers	Select to indicate that properties related to Table Triggers should be compared.
Constraint - Unmatched Constraints	Select to indicate that Table Constraints that are unmatched between the model and the live database should be compared. Typically these are constraints that exist in one system but do not exist in the other.
Constraints - Extended Properties	Select to indicate that extended properties of Table Constraints (such as Fill Factor and Clustered) should be compared.

Constraints - Remarks	Select to indicate that remarks applied to Table Constraints should be compared.

Notes

- The Database Compare functionality currently can perform comparisons on Table, View, Procedure, Function and Sequence object types

Execute DDL

The 'Execute DDL' tab provides a mechanism to easily execute generated DDL statements against a live database, and provides instant feedback on their success, all within the Enterprise Architect interface and without the need for other products.

There are two different types of DDL statement that Enterprise Architect can generate and send to the 'Execute DDL' tab:

- Create DDL statements, created by the Generate DDL screen, and
- Alter DDL statements, created by the Database Compare window

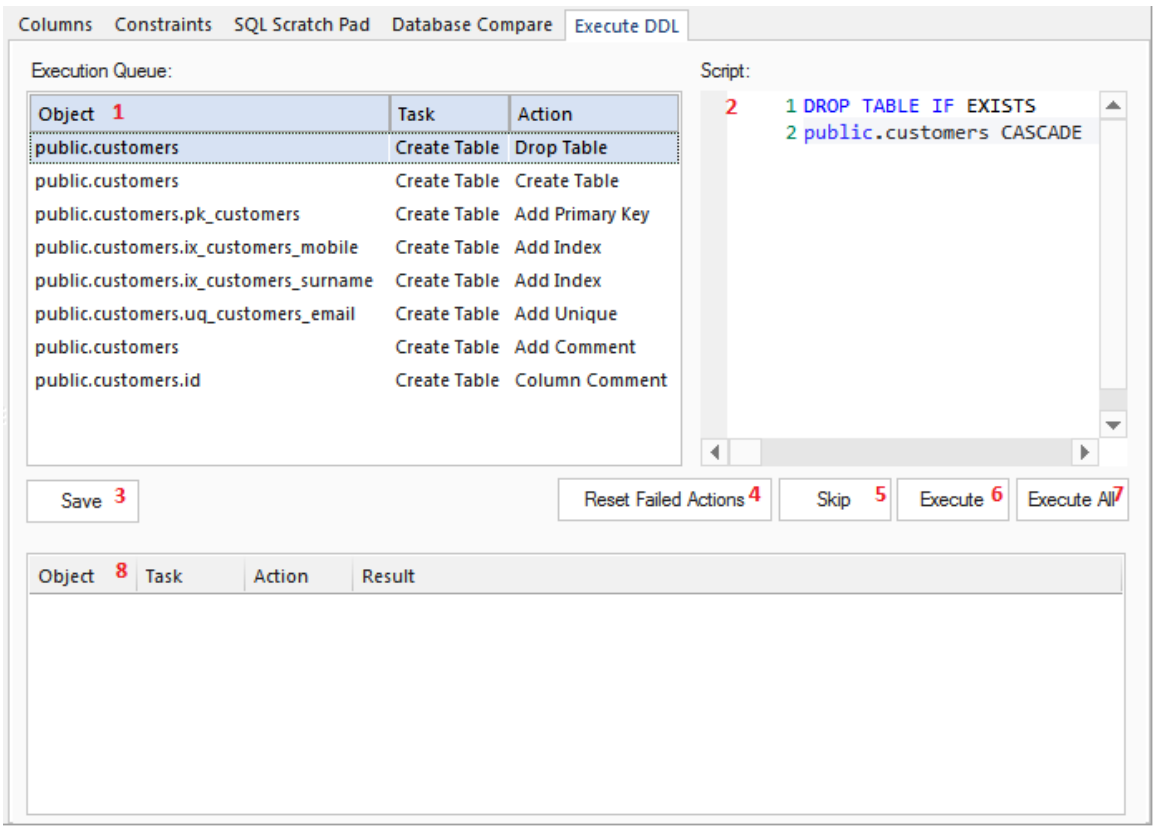
The Execute DDL functionality requires the Database Builder to have a valid connection to a live database. This database connection is shared between the SQL Scratch Pad, Database Compare and 'Execute DDL' tabs of the Database Builder.

Access

Open the Database Builder window, then display the 'Execute DDL' tab.

Ribbon	Develop > Data Modeling > Database Builder > Execute DDL

Execute the DDL



The 'Execute DDL' tab has these fields and buttons:

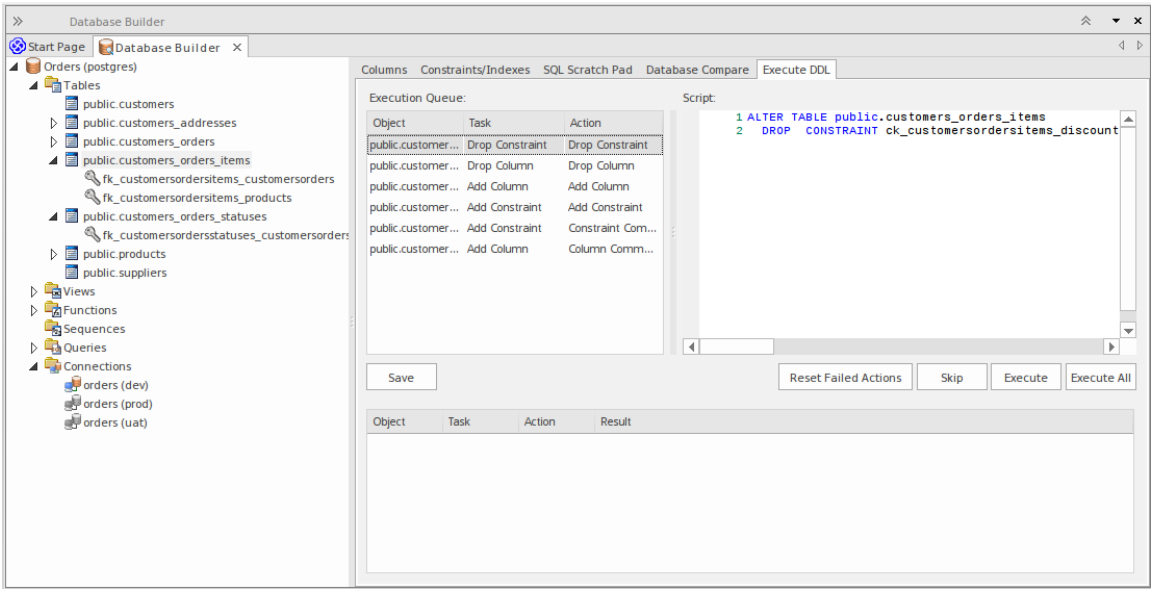
Field/Button	Action
1 Execution Queue	Lists the tasks (each with an associated DDL statement) that are yet to be executed. The list has three columns that specify the name of the object involved, the task and the action being performed. Selecting an item in the list will display the associated DDL statement (in the

	'Script' field) for the given task.
2 Script	A text box with SQL syntax highlighting, showing the DDL statement for the selected task.
3 Save	Click on this button to save all the individual DDL statements from both the 'Execution Queue' and the 'Results List' into a single file.
4 Reset Failed Actions	Click on this button to re-queue any failed or skipped tasks from the 'Results List' to the bottom of the 'Execution Queue'.
5 Skip	Click on this button to skip over the next task in the 'Execution Queue' and not execute it. The task will be moved into the 'Results List' and not given a result. When you click on the Reset Failed Actions button, skipped tasks are returned to the Execution Queue along with any failed tasks.
6 Execute	Click on this button to execute the next task in the 'Execution Queue'. The task is removed from the top of the 'Execution Queue' and added to the end of the

	'Results List' with the execution result.
7 Execute All	Click on this button to execute all tasks in the 'Execution Queue'. When execution is complete, the 'Results List' will display the results of each individual task.
8 Results List	Lists the executed tasks with the results of execution for each task. Selecting an item in this list will display the DDL statement that was executed, in the 'Script' field.

Example

In the example used in the earlier section on Database Comparison (when a column and constraint were renamed), if the defaults are used to 'push' the data model changes into the live database the Execute DDL screen is populated with the details shown here.



In summary, DDL is generated to drop both the old column and the old constraint (tasks 'Drop Column' and 'Drop Constraint'), then the column and constraint are created with the new names (tasks 'Add Column' and 'Add Constraint') and finally each has their comments/remarks applied (tasks 'Add Constraint - Constraint Comment' and 'Add Column - Column Comment').

Database Objects

Whilst Tables are the fundamental components of a relational database and allow the definition of Columns, Data Types, Keys and Indexes, there are a number of other Objects that are important in RDBM systems including:

- Views - a View represents the result-set of a pre-defined query; they are dynamically derived from the data stored in one or more Tables (or other Views)
- Procedures - a feature that some DBMS products implement to provide subroutines that can contain one or more SQL statements to perform a specific task such as data validation, access control, or to reduce network traffic between clients and the DBMS servers
- Functions - a feature that some DBMS products implement to provide a mechanism to extend the functionality of the database server; each is a routine that can accept parameters, perform an action (such as a complex calculation) and return the result of that action as a value
- Sequences - a feature that some DBMS products implement to provide a mechanism to generate unique values - the Sequence ensures that each call to it returns a unique value

The UML itself does not specify how data modeling is performed, but Enterprise Architect has a fully integrated UML profile for data modeling and a range of features built in to the core product that will make data modeling easy.

The profile uses stereotypes and Tagged Values to extend standard UML elements into data modeling constructs. This is achieved by adding the database object stereotype to a UML Class; so that you would model:

- Data Modeling diagrams as extended UML Class diagrams
- Tables as UML Class objects with a stereotype of <<table>>
- Views as UML Class objects with a stereotype of <<view>>
- Procedures as UML Class objects with a stereotype of <<procedure>>
- Functions as UML Class objects with a stereotype of <<function>>
- Sequences as UML Class objects with a stereotype of <<dbsequence>>

You can quickly create and configure all of these objects in your database model with Enterprise Architect.

Database Tables

Tables are the fundamental components of a relational database, representing multiple rows of structured data elements (referred to as Columns). Every individual item of data entered into a relational database is represented by a value in a column.

Enterprise Architect's UML Profile for Data Modeling represents:

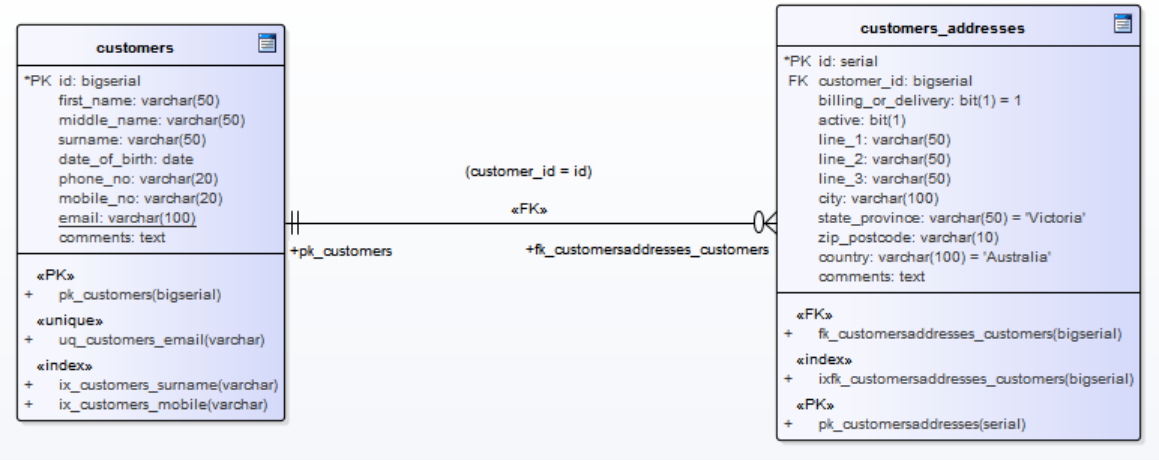
- Database Tables as UML Class objects with a stereotype of <<table>>
- Table columns as UML attributes of a Table, with a stereotype of <<column>>
- Primary Keys as UML operations/methods of a Table, with a stereotype of <<PK>>
- Foreign Keys as UML operations/methods of a Table, with a stereotype of <<FK>>
- Indexes as UML operations/methods of a Table, with a stereotype of <<index>>
- Unique Constraints as UML operations/methods of a Table, with a stereotype of <<unique>>
- Check Constraints as UML operations/methods of a Table, with a stereotype of <<check>>
- Table Triggers as UML operations/methods of a Table, with a stereotype of <<trigger>>

Enterprise Architect refers to all of the UML operations of a Table collectively as Constraints, hence the screen you use

to maintain a Table's UML attributes and operations is called the Columns and Constraints screen.

Example

This simple example of a Physical Data Model diagram in Enterprise Architect consists of two Database Tables represented by UML Classes, named *customers* and *customer_addresses*.



Each Table defines database columns, using UML attributes typed appropriately for the target DBMS (in this case, PostgreSQL).

Notes

- The Table stereotype is denoted by the icon in the top-right corner of each Class (see the *Data Modeling Notation* topic)
- The Enterprise Architect maintenance screen for

managing Table Columns doesn't allow you to change the attributes stereotype, since <<column>> is the only valid option

- It is possible to hide the <<column>> stereotype label shown in the example Tables (see the *Data Modeling Notation* topic)

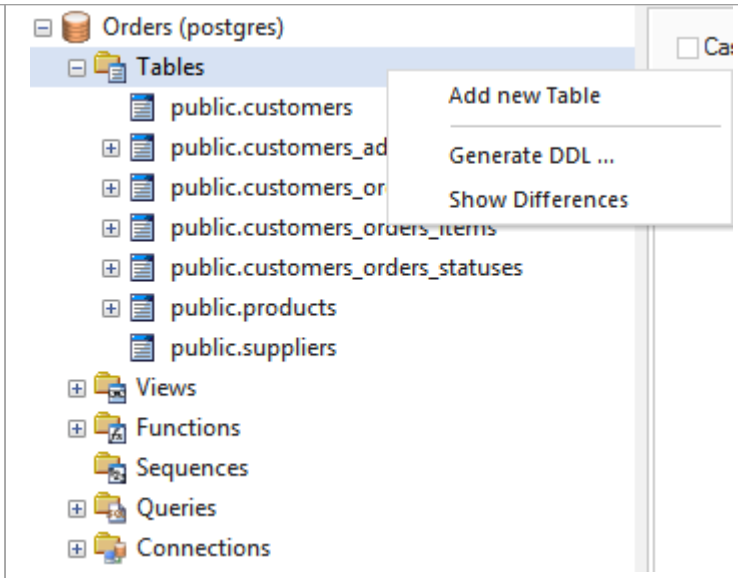
Create a Database Table

Fundamental to data modeling is the creation of Database Tables within the model. There are three ways to create a Table:

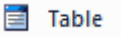
- Within the Database Builder
- On an open Data Model diagram
- Using the Browser New Element option

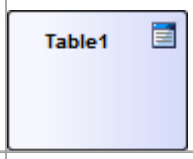
Add a Database Table with the Database Builder

Step	Action
1	Open the Database Builder ('Develop > Data Modeling > Database Builder').
2	Load or create a Data model.
3	Right-click on the Tables Package and select 'Add New Table'.

	
4	Overtype the default name with the appropriate name for the Table, and press the Enter key.
5	Double-click on the Table element to define the Table properties.

Add a Database Table to a diagram

Step	Action
1	Create and/or open a Data Modeling diagram.
2	Drag and drop the 'Table' toolbox icon onto the diagram. 

	This generates a new Table element: 
3	Double-click on the Table element to define the Table properties.

Database Table Columns

In a relational database, a Table column (sometimes referred to as a field) stores a single data value of a particular type in each row of the Table. Table columns can have various individual properties such as a default value or whether the field accepts Null values.

A Database Table Column is represented in the UML Data Modeling Profile as a stereotyped attribute; that is, an attribute with the <<column>> stereotype. In Enterprise Architect you define and maintain Table Columns using the purpose-designed 'Columns' page of the Database Builder, or the 'Columns and Constraints' dialog.

Create Database Table Columns

A database Table column is represented in the UML Data Modeling Profile as an attribute with the <<column>> stereotype. For a selected Table, you can review the existing columns and create new columns, on the 'Columns' page of the Database Builder or on the 'Columns and Constraints' screen.

You can define column details directly on the list of columns on the 'Columns' tab. The changes are automatically saved as you complete each field. Some fields have certain restrictions on the data you can enter, as described here. The tab also contains a 'Properties' panel and a 'Notes' field, which are populated with the existing information on the selected column. Each new column that you create is automatically assigned a set of default values and added to the bottom of the list.

Access

Ribbon	Develop > Data Modeling > Database Builder > Click on Table > Columns > Right-Click > Add new Column
Context Menu	In diagram, right-click on required Table | Features | Columns | Right-Click | Add

	new Column
Keyboard Shortcuts	Select a table | F9 | Tab Key (to set input focus on the 'Columns' tab) | Ctrl+N

Create columns in a Table

Option	Action
Name	Overtype the default name with the appropriate column name text.
Type	Click on the drop-down arrow and select the appropriate datatype for the column. The available datatypes depend on the DBMS assigned to the parent Table.
Length	(Optional) Some datatypes have a length component - for example, VARCHAR has a length that defines the number of characters that can be stored. If the datatype does not have a length component, this field is disabled. If the field is available and if you need to define a number of characters, type the

	value here.
Scale	(Optional) Some datatypes have a scale component - for example, DECIMAL has a scale that defines the number of decimal places that can be held. If the datatype does not have a scale component, this field is disabled. If the field is available and if you need to define a scale, type the value here.
PK	Select the checkbox if the column is part of the Primary Key for this Table.
Not Null	Select the checkbox if empty values are forbidden for this column. The checkbox is disabled if the 'PK' checkbox is selected.
Alias	If required for display and documentation purposes, type in an alternative name for the field.
Initial Value	If required, type in a value that can be used as a default value for this column.
Notes	Type in any additional information necessary to document the column. You can format the text using the Notes

	toolbar at the top of the field.

Column Properties

The appropriate properties for the Table's Database Management System automatically display in the 'Property' panel (expand the 'Column (<name>)' branch if they are not visible).

Property	DBMS
Autonum (Startnum Increment)	Oracle MySQL SQL Server DB2 PostgreSQL Notes: If you require an automatic numbering sequence, set this property to True and, if necessary, define the start number and increment.
Generated	DB2 Notes: Set this additional property for auto numbering in DB2, to 'By Default' or 'Always'.

NotForRep	SQLServer Notes: Set this property to True if you want to block replication.
Zerofill	MySQL Notes: Set this property to True or False to indicate if fields are zerofilled or not.
Unsigned	MySQL Notes: Set this property to True or False to indicate whether or not fields accept unsigned numbers.
LengthType	Oracle Notes: Set this property to define the character semantics as 'None', 'Byte' or 'Char'.

Delete Database Table Columns

For a selected database Table, you can review the existing columns and delete any individual column, on the 'Columns' tab of the Columns and Constraints screen.

Access

Use one of the methods outlined here to display a list of columns for a table, then select a column and delete it.

When you select the 'Delete column '<name>" option, if all validation rules are satisfied the column is immediately deleted.

Ribbon	Develop > Data Modeling > Database Builder > Click on Table > Columns > Right-click on column name > Delete column <name>
Context Menu	In diagram, right-click on required Table | Features | Columns | Right-click on column name | Delete column <name>
Keyboard Shortcuts	F9 | Use 'Up Arrow' or 'Down Arrow' to select a column | Ctrl+D

Notes

- If the deleted database Table column is involved in any constraints it will automatically be removed from them

Reorder Database Table Columns

If you have several columns defined in a database Table, you can change the order in which they are listed. The order in the list is the order in which the columns appear in the generated DDL.

Access

Use one of the methods outlined here to display a list of columns for a Table, then select a column and reposition it within the list.

Ribbon	Develop > Data Modeling > Database Builder > Click on Table
Context Menu	In diagram, right-click on required Table | Features | Columns
Keyboard Shortcuts	F9

Change the column order

Step	Action
1	In the 'Columns' tab, click on the required column name in the list.
2	Right-click and select the: • 'Move column <name> up' option (or press Ctrl+Up Arrow) to move the column up one position • 'Move column <name> down' option (or press Ctrl+Down Arrow) to move the column down one position These options have an immediate effect both in the 'Columns' tab and on a diagram.

Working with Database Table Properties

Once you have created a Database Table, you can review its properties and check that the DBMS and Owner values are correct. To display the 'Properties' dialog for a Table, either double-click on the Table name in the 'Database Builder Tables' Package or on the Table element on a diagram.

Important

A DBMS must be assigned to a Table before you can add columns in it. If you are using the Database Builder then the DBMS of the data model will be automatically applied to all new Tables; however, if you have added a Table by other means (such as working on a diagram) then this is a manual step.

Tasks

Once the Database Table properties are defined, you are ready to add columns.

Task
Set the database type for a Table - other than the Table

name, the most important property to set for a Database Table is the database type.
Set the database Table Owner - For some DBMSs all Tables must be assigned an Owner/Schema; in Enterprise Architect this property is defined as a Tagged Value with the name Owner.
Set extended options - some DBMSs have extended options that are only relevant to that DBMS. These extended properties are stored as Tagged Values.

Default DBMS

Prior to creating a Physical Data Model it is advisable for you to set the default DBMS, which will be automatically applied to new database objects that you create outside of the Database Builder. You can set the default DBMS type in one of these ways:

- Select 'Start > Appearance > Preferences > Preferences > Source Code Engineering > Code Editors', then set the field 'Default Database'
- Select 'Settings > Reference Data > Settings > Database Datatypes', then select a Product Name and select the 'Set as Default' checkbox
- Set the DBMS in the second field of the Code Generation

Toolbar

Set the Database Type

The most important property to set for a Database Table (after its name) is the database type or DBMS. The DBMS value selected will control how Enterprise Architect will determine:

- How the Table name will be shown (with or without an Owner)
- What set of validation rules will be applied while database modeling
- The data types that are available when creating columns,
- What set of DDL templates will be used in DDL Generation

Access

Select a Table in the Browser window or on a diagram then, using any of the methods outlined here, open the Table's 'Properties' dialog, display the 'General' tab, then display the 'Main' child tab.

Ribbon	Design > Element > Editors > Properties Dialog > General > Main
Context Menu	Right-click on the Table element | Properties | Special Action | General | Main

Keyboard Shortcuts	Shift+Enter | General | Main
Other	Double-click on the Table element |General | Main

Options

Field/Button	Action
Database	Click on the drop-down button and select the required database type from the list.
Apply	Click on the Apply button to save any pending changes.
OK	Click on the OK button to save any pending changes and close the screen.

Set Database Table Owner/Schema

For some DBMSs all Tables must be assigned an Owner/Schema. In Enterprise Architect this property is physically defined as a Tagged Value with the name Owner. However, a special properties page is provided to help you easily manage the Owner property.

Access

Select a Table in the Browser window or on a diagram then, using any of the methods outlined here, open the Table's 'Properties' dialog, display the 'General' tab and display the 'Table Detail' child tab.

Ribbon	Design > Element > Editors > Properties > << table >>
Context Menu	Right-click on the Table element | Properties |Special Action > General > Table Detail
Keyboard Shortcuts	Shift+Enter | General | Table Detail
Other	Double-click on the Table element |'General' | 'Table Detail'

Set the Database Table owner

Step	Action
1	In the 'Owner' field, type the name of the owner or schema of the Table.

Set MySQL Options

To make use of Foreign Keys in MySQL, you must declare the Database Table type as InnoDB.

Declare the Table type as InnoDB

Step	Action
1	Add a Tagged Value named *Type* to the Table.
2	Set the 'Value' field to 'InnoDB'.

Generate DDL

When you generate DDL for this Table, the Table type is included in the SQL script.

To allow for later versions of MySQL, additional Table options that can be added in the same way include:

Tag	Value (Example)
ENGINE	InnoDB

CHARACTER SET	latin1
CHARSET	latin1
COLLATE	latin1_german2_ci

Set Oracle Database Table Properties

To set additional Oracle Database Table properties, you use the Table's Tagged Values.

Set Properties

The same properties can be added to indexes and constraints, by highlighting the index or constraint Operation and adding the appropriate Tagged Values.

Step	Action
1	Add one or more Tagged Values to the Table, using the names provided in the 'Property/Tag' column of the 'Properties' Table.
2	Specify the appropriate value for each tag. Examples are provided in the 'Value' column of this Properties Table. • CACHE - NOCACHE • DBVERSION - 9.0.111 • FREELISTS - 1 • GRANT OWNER1 - SELECT • GRANT OWNER2 - DELETE, INSERT, SELECT, UPDATE

	• INITIAL - 65536 • INITRANS - 1 • LOGGING - LOGGING • MAXEXTENTS - 2147483645 • MAXTRANS - 255 • MINEXTENTS - 1 • MONITORING - MONITORING • OWNER - OWNER1 • PARALLEL - NOPARALLEL • PCTFREE - 10 • PCTINCREASE - 0 • PCTUSED - 0 • SYNONYMS - PUBLIC:TABLE_PUB;OWNER2:TABLE_OWNER2 • TABLESPACE - MY_TABLESPACE • TEMPORARY - YES

Database Table Constraints/Indexes

Within Enterprise Architect, Table Constraints and Indexes are modeled on the same screen; collectivity they are referred to as Constraints. Database Constraints define the conditions imposed on the behavior of a database Table. They include:

- Primary Key - uniquely identifies a record in a Table, consisting of one or more columns
- Index - improves the performance of retrieval and sort operations on Table data
- Unique Constraints - a combination of values that uniquely identify a row in the Table
- Foreign Key - a column (or collection of columns) that enforce a relationship between two Tables
- Check Constraints - enforces domain integrity by limiting the values that are accepted by a column
- Table Trigger - SQL or code automatically executed as a result of data in a Table being modified

In Enterprise Architect, you can define and maintain Table Constraints using either the purpose-designed 'Constraints/Indexes' page of the Database Builder or the Columns and Constraints screen.

Access

Ribbon	Develop > Data Modeling > Database Builder > Click on Table name > Constraints/Indexes | Right-click | Add New Constraint
Context Menu	In diagram | Right-click on Table | Features | Constraints/Indexes | Right-click | Add New Constraint
Keyboard Shortcuts	Click on Table: F9 > Constraints/Indexes: Ctrl+N

Create a Constraint

The process of creating any of these constraint types is the same and is achieved in one of the ways described here.

Create a Constraint - Using the context menu or keyboard

Step	Action

1	A new constraint is automatically created and assigned the default name *constraint n* (where *n* is a counter) and a 'Type' of 'index'. Overtype the default name with your own constraint name.
2	If necessary, in the 'Type' field click on the drop-down arrow and select the appropriate constraint type.
3	If you prefer, type an alias for the constraint, in the 'Alias' field. The 'Columns' field is read-only; it is populated with the columns that you assign to the 'Involved Columns' tab.

Create a Constraint - Overtype the template text

Step	Action
1	On the 'Constraints/Indexes' tab for the selected Table, the list of constraints ends with the template text *New Constraint*.

	Overtype this text with the appropriate constraint name, and press the Enter key.
2	The new constraint is automatically created and assigned the default Type of index. If necessary, in the 'Type' field click on the drop-down arrow and select the appropriate constraint type.
3	If you prefer, type an alias for the constraint, in the 'Alias' field. The 'Columns' field is read-only; it is populated with the columns that you assign to the 'Involved Columns' tab.

Assign Columns to a Constraint

The constraint types of Primary Key, Foreign Key, Index and Unique all must have at least one column assigned to them; this defines the columns that are involved in the constraint.

Step	Action
1	On the 'Constraints/Indexes' tab for the selected Table, click on the constraint to which you are

	assigning columns.
2	The 'Available Columns' panel lists all columns defined for the Table. For each column to assign to the constraint, right-click on the column name and select 'Assign column <name>'. The column name is transferred to the 'Assigned Columns' list.

Unassign Columns from a Constraint

Step	Action
1	On the 'Constraints/Indexes' tab for the selected Table, click on the constraint from which you are unassigning columns.
2	In the 'Assigned Columns' list, right-click on the name of the column to unassign from the constraint and select 'Unassign column <name>'. The column name is transferred to the 'Available Columns' list.

Reorder the Assigned Columns in a Constraint

If you have a number of columns in the constraint, you can re-arrange the sequence by moving a selected column name one place up or down the list at a time. To do this:

- Right-click on the column name to move and select either:
  - Move column '<name>' up (Ctrl+Up Arrow) or
  - Move column '<name>' down (Ctrl+Down Arrow)

Delete a constraint

To delete a constraint you no longer require, right-click on the constraint name in the list on the 'Constraints/Indexes' tab and select the 'Delete constraint <name>' option. If all validation rules for the given constraint type are met, the constraint is immediately removed from the repository along with all related relationships (if there are any).

Primary Keys

A Primary Key is a column (or set of columns) that uniquely identifies each record in a Table. A Table can have only one Primary Key. Some DBMSs support additional properties of Primary Keys, such as Clustered or Fill Factor.

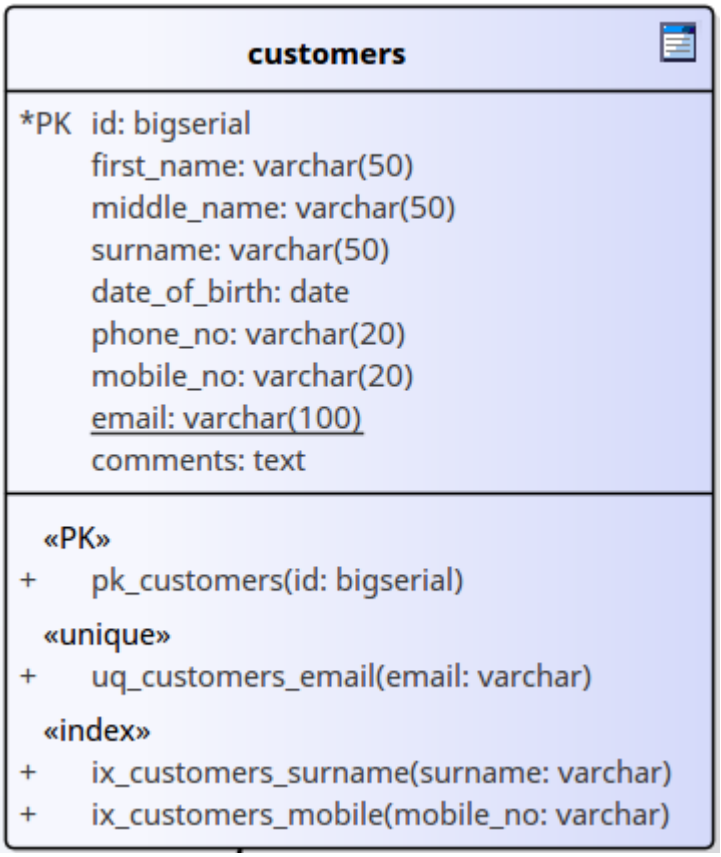
Access

Ribbon	Develop > Data Modeling > Database Builder > Click on Table name
Context Menu	In diagram | Right-click on Table | Features | Constraints/Indexes

Create a Primary Key

In Enterprise Architect you can create a Primary Key from either the 'Columns' tab or the 'Constraints/Indexes' tab. In either case, when you add a column to a Primary Key constraint, the column is automatically set to be 'Not Null'. Additionally any diagram (assuming the 'Show Qualifiers and Visibility Indicators' option is set) containing the Table element will show the 'PK' prefix against the column name.

In this image, see the first column 'id: bigserial'.



Create a Primary Key - from the Columns tab

Step	Action
1	Either: • In the Database Builder, click on a Table with one or more defined columns, and click on the 'Columns' tab, or • On a diagram, click on a Table and press F9 to display the 'Columns' tab

2	For each column to include in the Primary Key, select the 'PK' checkbox. If a Primary Key constraint is not previously defined for the current Table, the system will create a new constraint using the Primary Key Name template.

Create a Primary Key - from the Constraints tab

Step	Action
1	Either: • In the Database Builder, click on a Table with one or more defined columns, and click on the 'Constraints/Indexes' tab, or • On a diagram, click on a Table and press F10 to display the 'Constraints/Indexes' tab
2	Overtype the *New Constraint* text with the Primary Key name, press the Enter key and click on the 'Type' field drop-down arrow, and select 'PK'.
3	Assign the required columns to the PK constraint.

4	Set the Primary Key's extended properties using the property panel. • Fill Factor is a numeric value between 0 and 100 • Is Clustered is a Boolean value that determines the physical order of how the data is stored; for most DBMSs the Is Clustered property defaults to True for Primary Keys

Remove columns from a Primary Key

You can remove columns from a Primary Key using either the 'Columns' tab or the 'Constraints/Indexes' tab.

Remove columns from a Primary Key - using the Columns tab

Step	Action
1	Either: • In the Database Builder, click on the Table with the Primary Key, and click on the 'Columns' tab, or

	• On a diagram, click on a Table and press F9 to display the 'Columns' tab
2	Against each column you want to remove from the Primary Key, deselect the 'PK' checkbox. If you have removed all columns from the Primary Key constraint and the Primary Key is no longer needed, it must be manually deleted.

Remove columns from a Primary Key - using the Constraints/Indexes tab

Step	Action
1	Either: • In the Database Builder, click on the Table with the Primary Key, and click on the 'Constraints/Indexes' tab, or • On a diagram, click on a Table and press F10 to display the 'Constraints/Indexes' tab
2	Unassign the columns on the PK constraint, as necessary.

Notes

- Warning: Enterprise Architect assumes that Primary Key constraints have at least one column assigned to them; however, Enterprise Architect does not enforce this rule during modeling
  If DDL is generated for a Table whose Primary Key has no column assigned, that DDL will be invalid

Non Clustered Primary Keys

When you create a Primary Key in some DBMSs (such as SQL Server or ASA), it is automatically created with the 'Is Clustered' property set to True. Therefore when you model a Primary Key in an Enterprise Architect data model, the same behavior occurs.

Clustered indexes provide improved performance for accessing the column(s) involved, by physically organizing the data by those columns. There can be only one clustered index per Table.

In some situations, you might be more interested in the performance of columns other than the ones assigned to the Primary Key, and therefore you would need to change the default assignment so that the Primary Key is not clustered.

Access

Ribbon	Develop > Data Modeling > Database Builder > Click on Table name > Constraints/Indexes
Context Menu	In diagram or Browser window | Right-click on Table | Features | Constraints/Indexes

Keyboard Shortcuts	Click on Table: F9 > Constraints/Indexes

Define Primary Key as non-clustered

Subsequently, you can model an index for the same Table as clustered.

Step	Action
1	Highlight the existing Primary Key constraint. The Primary Key properties display in the 'Property' panel.
2	For the *Is Clustered* property, in the 'Value' field click on the drop-down arrow and change the value to False.

Database Indexes

Database indexes are applied to Tables to improve the performance of data retrieval and sort operations. Multiple indexes can be defined against a Table; however, each index imposes overheads (in the form of processing time and storage) on the database server to maintain them as information is added to and deleted from the Table

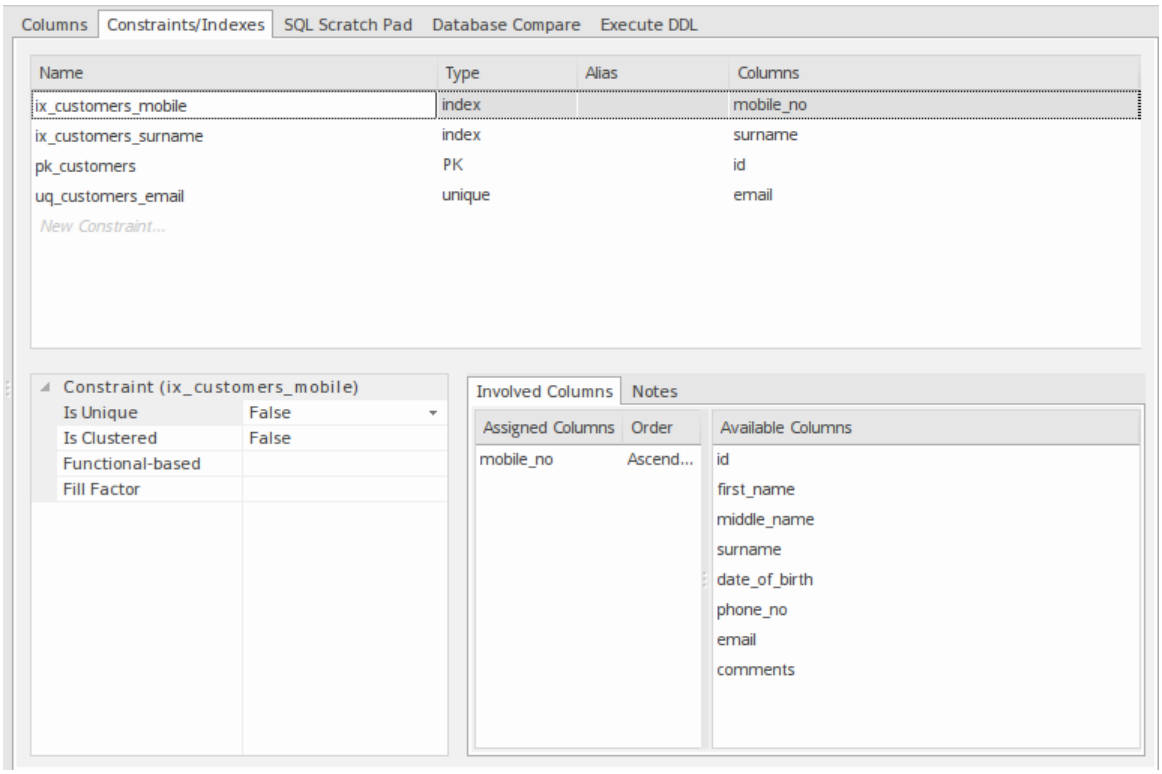
In Enterprise Architect an index is modeled as a stereotyped operation.

Some DBMSs support special types of index; Enterprise Architect defines these using additional properties such as function-based, clustered and fill-factor.

Access

Ribbon	Develop > Data Modeling > Database Builder > Click on Table name > Constraints/Indexes
Context Menu	In diagram | Right-click on Table | Features | Constraints/Indexes
Keyboard Shortcuts	Click on Table: F9 > Constraints/Indexes

Work on an index



Step	Action
1	On the 'Constraints/Indexes' tab for the Table, right-click and select 'Add new constraint'. The new constraint is added with the default name 'constraint1' and the Type of 'index'. Overtype the name with your preferred index name.
2	Assign the appropriate columns to the Index. The 'Assigned Columns' list has an additional 'Order' field that specifies the order (Ascending or

	Descending) in which each assigned column is stored in the index. You can toggle the order for each column, as required. Additionally, for MySQL indexes, a 'Len' field will be visible in which you can define Partial Indexes; that is, an index that uses the leading 'n' number of characters of a text based field. The 'Len' field takes only whole number numeric values of between 0 and the column's defined length. A value of 0 (which is the default) indicates that the entire column is to be indexed.
3	In the 'Property' panel, review the settings of the extended properties that are defined for the current DBMS.

Additional Properties

Property	Description
Is Unique	(True / False) indicates whether the current index is a 'Unique Index'. A Unique Index ensures that the indexed column (or columns) does not contain duplicate values, thereby ensuring that each row has a unique value (or

	combination of values when the index consists of multiple columns).
Is Clustered	(True / False) indicates whether the current index is a 'Clustered Index'. With a clustered index, the rows of the table are physically stored in the same order as in the index, therefore there can be only one clustered index per table. By default a table's Primary Key is clustered. Not all DBMS's support clustered indexes, therefore the 'Is Clustered' Index property will only be visible for DBMSs that support it.
Is Bitmap	(True / False) indicates whether the current index is a 'Bitmap' index. Bitmap indexes are meant to be used on columns that have relatively few unique values (referred to as 'low cardinality' columns) and that physically consist of a bit array (commonly called bitmaps) for each unique value. Each of the arrays will have a bit for each row in the table. Consider this example: a bitmap index is created on a column called 'Gender', which has the options 'Male' or 'Female'. Physically, the index will consist of two bit arrays, one for 'Male' and one for

	'Female'. The female bit array will have a 1 in each bit where the matching row has the value 'Female'. The 'Is Bitmap' and 'Is Unique' properties are mutually exclusive, and so the DDL generation will ignore the 'Is Unique' property when the 'Is Bitmap' property is True. Bitmap Indexes are only supported by Oracle; therefore, this property is only visible while modeling Oracle indexes.
Fill Factor	A numeric value between 0 and 100, that defines the percentage of available space that should be used for data. Not all DBMSs support fill factor, therefore the 'Fill Factor' index property will only be visible for DBMSs that support it.
Functional-based	A SQL statement that defines the function/statement that will be evaluated and the results indexed; for example: LOWER("field") Not all DBMSs support functional-based indexes, therefore the 'Functional-based' Index property will only be visible for DBMSs that support them, such as PostgreSQL and Oracle.

Include	Identifies a comma-separated list (CSV) of non-key Columns from the current table. Not all DBMSs support the 'Include' property on indexes, therefore this property will only be visible for DBMSs that support it.

Notes

- Warning: Enterprise Architect assumes that Indexes have at least one column assigned to them; however, Enterprise Architect does not enforce this rule during modeling
  If DDL is generated for a Table that has an Index defined without column(s) assigned, that DDL will be invalid, unless the index is functional-based
- Any columns assigned to a functional-based index are ignored

Unique Constraints

Unique Constraints enforce the 'uniqueness' of a set of fields in all rows of a Table, which means that no two rows in a Table can have the same values in the fields of a Unique Constraint. Unique Constraints are similar to Primary Keys (in that they also enforce 'uniqueness') but the main difference is that a Table can have multiple Unique Constraints defined but only one Primary Key.

Access

Ribbon	Develop > Data Modeling > Database Builder > Click on Table name > Constraints/Indexes > Right-click > Add New Constraint
Context Menu	In diagram or Browser window | Right-click on Table element | Features | Constraints/Indexes
Keyboard Shortcuts	Click on Table: F9 > Constraints/Indexes: Ctrl+N

Create a Constraint

Step	Action
1	On the 'Constraints/Indexes' tab, a new constraint is automatically created and assigned the default constraint name and a 'Type' of index. Overtype the constraint name with a name that identifies this as a unique constraint.
2	In the 'Type' field, change the value from 'index' to 'unique'.

Notes

- Warning: Enterprise Architect assumes that Unique Constraints have at least one column assigned to them; however, Enterprise Architect does not enforce this rule during modeling
  If DDL is generated for a Table that has a unique constraint defined without column(s) assigned, that DDL will be invalid

Foreign Keys

A Foreign Key defines a column (or a collection of columns) that enforces a relationship between two Tables. It is the responsibility of the database server to enforce this relationship to ensure data integrity. The model definition of a Foreign Key consists of a parent (primary) Table containing a unique set of data that is then referred to in a child (foreign) Table.

In Enterprise Architect, a Foreign Key is modeled with two different (but related) UML components:

- A Foreign Key constraint (a UML operation with the stereotype of <<FK>>) stored on the child Table and
- An Association connector (stereotype of <<FK>>) defining the relationship between the two Tables

Create a Foreign Key

Although the definition of a Foreign Key can be complex, the Foreign Key Constraint screen simplifies the modeling of Foreign Keys. This screen is purpose-designed to help you select which constraint in the parent Table to use, and will automatically match the child Table columns to those in the parent Table that are part of the constraint. Different aspects of the process of developing a Foreign Key are described here separately for illustration, but the overall process should be a smooth transition.

A number of conditions must be met before a Foreign Key definition can be saved:

- Both Tables must have matching DBMSs defined
- The parent Table must have at least one column
- The parent Table must have a Primary Key, unique constraint or unique index defined

Create a Foreign Key - using the Database Builder

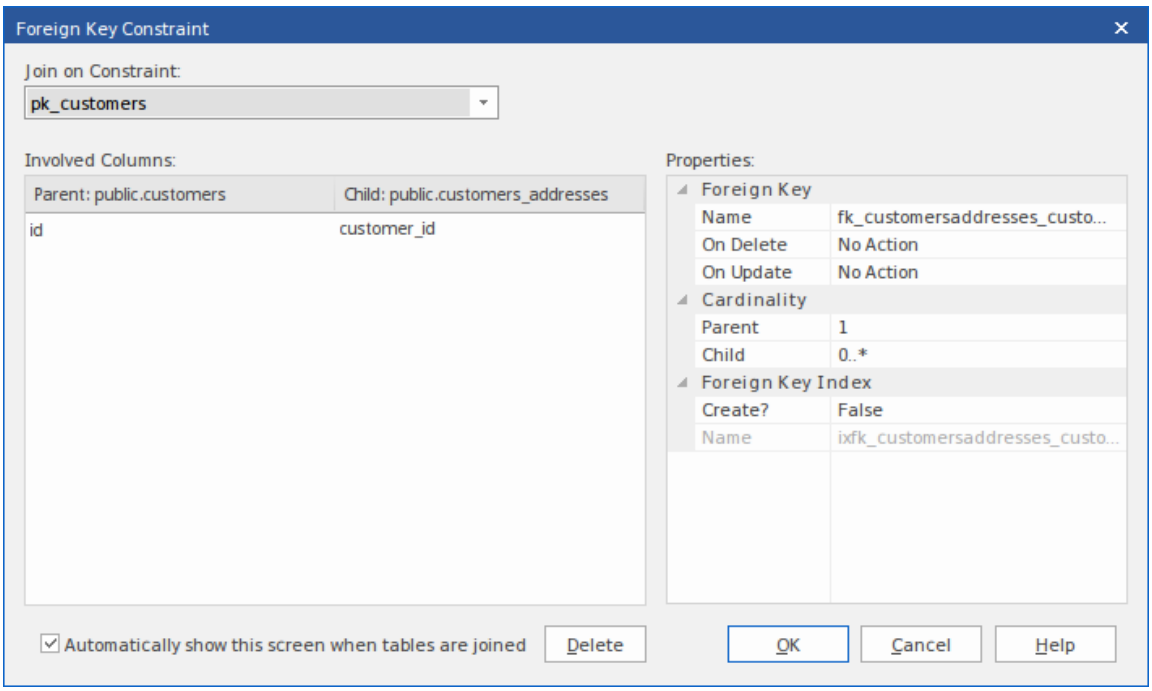
Step	Action
1	In the Database Builder tree, right-click on the child Table name and click on 'Add new Foreign Key on <table name>'. A dialog displays listing all the possible parent Tables.
2	Double-click on the required parent Table name in the list or select it and click on the OK button. The 'Foreign Key Constraint' screen displays.

Create a Foreign Key - using a relationship on a diagram

Step	Action
1	In the Data Modeling diagram, locate the required child (Foreign Key) Table and parent (Primary Key) Table.
2	Select an Association connector in the 'Data Modeling' page of the Diagram Toolbox.
3	Click on the child Table and draw the connector to the parent Table.
4	If the Foreign Key Constraint screen has been set to display automatically when two Tables are joined, it displays now. Otherwise, either: • Double-click on the connector or • Right-click on the connector and select the 'Foreign Keys' option The Foreign Key Constraint screen displays.

The Foreign Key Constraint Screen

As an example this image shows the Foreign Key Constraint screen loaded with the details of 'fk_customersaddresses_customers' (as defined in the Example model).



Option	Action
Join on Constraint	This combo box lists all defined constraints in the parent Table that could be used as the basis of a Foreign Key. (These constraints can be Primary Keys, Unique Constraints or Unique Indexes.) The first constraint in the list is selected by default; if this is not the constraint you want, select the correct constraint from the combo box. When you select the constraint, its columns are automatically listed in the

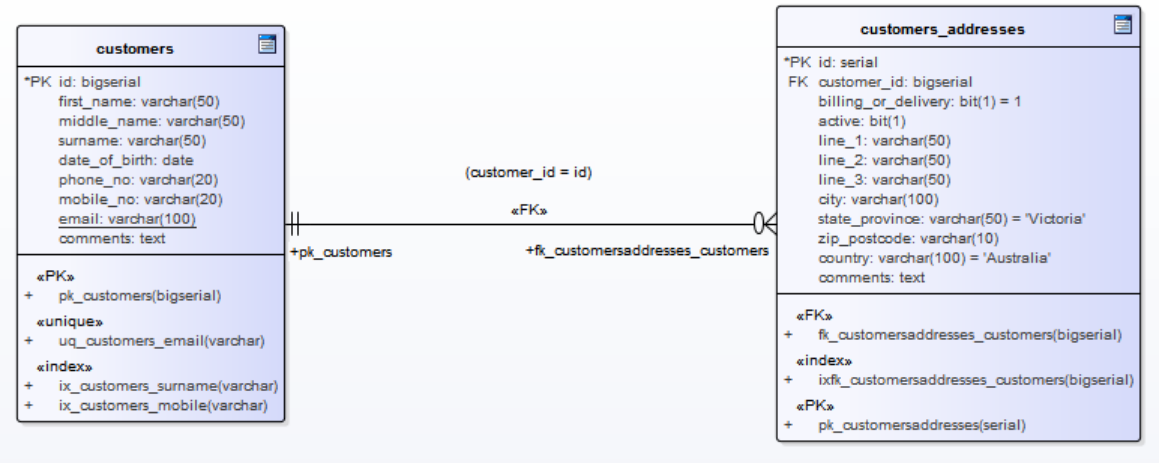
	'Involved Columns' panel, under the 'Parent: <tablename>' column.
Involved Columns	This list is divided into two: the columns involved in the selected constraint are listed on the left, and the child columns that are going to be paired to the parent columns are listed on the right. When a constraint is selected (in the 'Join on constraint' field) the parent side is refreshed to display all columns assigned to the selected constraint. On the child side the system will automatically attempt to match each parent column to one of the same name in the child Table. If the child Table does not have a column of the same name, a new column of that name will be added to the list, flagged with (*) to indicate that a new column will be created in the Table. However, if you want to force the pairing to an existing child Table column or a new column with a different name, click on the column name field and either: • Type in the replacement name, or • Select an existing column (click on the drop-down arrow and select the name from the list)

Name	This field defines the name of the Foreign Key constraint, and defaults to a name constructed by the Foreign Key Name Template. To change the name to something other than the default, simply overtype the value.
On Delete	Select the action that should be taken on the data in the child Table when data in the parent is deleted, so as to maintain referential integrity.
On Update	Select the action that should be taken on the data in the child Table when data in the parent is updated, so as to maintain referential integrity.
Parent	Click on the drop-down arrow and select the cardinality of the parent Table in the Foreign Key.
Child	Click on the drop-down arrow and select the cardinality of the child Table in the Foreign Key.
Create?	If you want to create a Foreign Key Index at the same time as the Foreign Key, set this property to True.

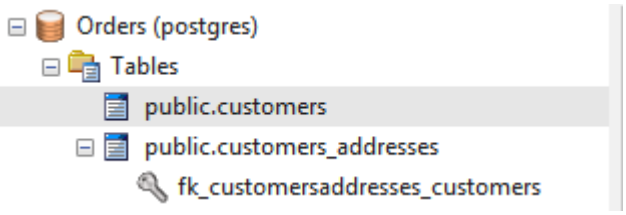
	The name of the Foreign Key Index is controlled by the Foreign Key Index template, and the generated name is shown in the 'Name' field underneath the 'Create?' field.
Automatically show this screen when tables are joined	(For diagrammatic modeling) Select this checkbox to automatically display this screen whenever an Association is created between two Tables.
Delete	Click on this button to delete the currently selected existing (saved) Foreign Key. A prompt is displayed to confirm the deletion (and the deletion of the Foreign Key Index, if one exists) - click on the Yes button. Deleting a Foreign Key leaves an Association connector in place, which you can either edit or delete (right-click and select 'Delete association: to <Table name>').
OK	Click on this button to save the Foreign Key.

Examples

This example shows simple Foreign Keys in a diagram:



The same Foreign Key will be shown in the Database Builder's tree as a child node under the Table 'customers.addresses'.



Check Constraints

A Check Constraint enforces domain integrity by limiting the values that are accepted by a column.

Access

Ribbon	Develop > Data Modeling > Database Builder > Click on Table name > Constraints/Indexes > Right-click > Add New Constraint
Context Menu	In diagram | Right-click on Table | Features | Constraints/Indexes | Right-click | Add New Constraint
Keyboard Shortcuts	Click on Table: F9 > Constraints/Indexes: Ctrl+N

Create a Constraint

Step	Action

1	On the 'Constraints/Indexes' tab of the Columns and Constraints screen, a new constraint is automatically created and assigned the default constraint name and a 'Type' of index. Overtype the constraint name with a name that identifies the constraint as a check constraint, such as 'CHK_ColumnName' (the CHK_ prefix is optional).
2	In the 'Type' field, change the value from 'index' to 'check'.
3	In the 'Properties' panel for the Condition property, type the SQL statement that will be used as the Check Condition; for example, column1 < 1000. If the condition is long, click on the  button to display a SQL editor (with syntax highlighting).

Delete a Check Constraint

If you do not want to keep a check constraint, either:

- Right-click on it in the list and select 'Delete constraint <name>', or
- Click on the item and press Ctrl+D

The constraint is immediately deleted.

Notes

- Any columns assigned to a check constraint are ignored

Table Triggers

A Table trigger is SQL or code that is automatically executed as a result of data being modified in a database Table. Triggers are highly customizable and can be used in many different ways; for example, they could be used to stop certain database activities from being performed during business hours, or to provide validation or perform deletions in secondary Tables when a record in the primary Table is deleted.

In Enterprise Architect, a Table trigger is modeled as a stereotyped operation and managed using the Table's 'Constraints' screen.

Access

Ribbon	Develop > Data Modeling > Database Builder > Click on Table name > Constraints/Indexes | Right-click | Add New Constraint
Context Menu	In diagram | Right-click on Table | Features | Constraints/Indexes | Right-click | Add New Constraint
Keyboard	Click on Table: F9 > Constraints/Indexes:

Shortcuts	Ctrl+N

Create a Table Trigger

Step	Action
1	On the 'Constraints/Indexes' tab, a new constraint is automatically created and assigned the default constraint name and a 'Type' of index. Overtype the constraint name with a name that identifies the constraint as a trigger, such as TRG_OnCustomerUpdate. (The TRG_ prefix is optional.)
2	In the 'Type' field, change the value from 'index' to 'trigger'.
3	In the 'Properties' panel for the Statement property, type in the complete SQL statement (including CREATE TRIGGER) that will define the Trigger. If the condition is long, click on the ... button to display a SQL editor (with syntax highlighting).
4	The properties Trigger Time and Trigger Event are currently information-only values and are not used in

	DDL generation.

Delete a Table Trigger

If you do not want to keep a trigger, either:

- Right-click on it in the list and select 'Delete constraint <name>', or
- Click on the item and press Ctrl+D

The trigger is immediately deleted.

Notes

- Any columns assigned to table triggers are ignored

Database Views

A Database View represents the results of a pre-defined query. Unlike a Table, a View is dynamically derived from data in one or more Tables (or other Views). Enterprise Architect supports the definition of Views both with and without this statement:

"Create View {viewName} As" statement

The system will automatically add it dynamically (if missing) whenever DDL generation is performed. The advantage of not defining this statement is that when a view object is renamed the 'View Definition' property does not have to be manually updated.

You can create a Database View either:

- Within the Database Builder or
- By dragging the 'View' icon from the Data Modeling Toolbox onto a diagram

Add a Database View with the Database Builder

Step	Action
1	Open the Database Builder.

2	Load or create a Data model.
3	Right-click on the 'Views' Package and select 'Add New View'.
4	Overtype the default name with the appropriate name for the View, and press the Enter key.
5	Double-click on the new View, or right-click on it and select 'SQL Object Properties'. The 'SQL Object Editor' dialog displays.

Add a Database View to a diagram

Step	Action
1	Open your Data Modeling diagram and, if necessary, display the 'Data Modeling' page of the Diagram Toolbox (click on  to display the 'Find Toolbox Item' dialog and specify 'Data Modeling').
2	Drag the 'View' icon onto the diagram.  This generates the View element:

	
3	Right-click on the new View element and select 'SQL Object Properties'. The 'SQL Object Editor' dialog displays.

SQL Object Editor

The 'SQL Object Editor' dialog is shared by a number of SQL-based database objects (Views, Procedures, Functions and Sequences); it helps the data modeler manage the various properties of the SQL-based object.

Option	Action
Database	If it has already been set, the default database type displays. If the default has not been set, or you want to change the database type for this View, click on the drop-down arrow and select the target DBMS to model.
Dependencies	A list of objects that the current object depends on. The 'Dependencies' list shows:

	• Each Depends connector between this View and another Table or View • Any object names (specified as a CSV list) in the 'parents' Tagged Values
Notes	If necessary, type in a comment on the current View.
Definition	Type the full SQL View definition. For releases of Enterprise Architect up to 12.1 (Build 1227), this must include the CREATE_VIEW syntax as appropriate for the target DBMS (for later versions this is not needed). For example: CREATE VIEW 'MyViewName' AS [view definition] The code editor provides Intelli-sense for basic SQL keywords, functions and names of all objects in the current data model.

Database Procedures

Database Procedures (sometimes referred to as Stored Procedures or Procs) are subroutines that can contain one or more SQL statements that perform a specific task. They can be used for data validation, access control, or to reduce network traffic between clients and the DBMS servers. Extensive and complex business logic can be embedded into the subroutine, thereby offering better performance.

Database Procedures are similar to Database Functions. The major difference is the way in which they are invoked - Database Functions can be used in the same way as for any other expression within SQL statements, whereas Database Procedures must be invoked using the CALL or EXEC statement, depending on the DBMS.

In Enterprise Architect, Database Procedures can be modeled in one of two ways:

- As individual objects (the default method) or
- As operations in a container

Functionally the two methods result in the same DDL being produced. The main difference is visual - by having several Operations in one container, you have fewer elements and less clutter on the diagram.

Individual objects

Database Procedures modeled as individual objects are

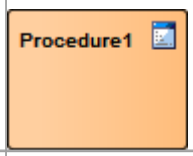
UML Classes with the stereotype «procedure»; you create these either:

- Within the Database Builder or
- By dragging the 'Procedure' icon from the Data Modeling Toolbox onto a diagram

Add a Database Procedure using the Database Builder

Step	Action
1	Open the Database Builder.
2	Load or create a Data model.
3	Right-click on the Procedures Package and select 'Add New Procedure'.
4	Overtype the default name with the appropriate name for the Procedure, and press the Enter key.
5	Double-click on the new Procedure, or right-click on it and select 'SQL Object Properties'. The SQL Object Editor screen displays.

Add a Database Procedure to a diagram

Step	Action
1	Open your Data Modeling diagram and, if necessary, display the 'Data Modeling' page of the Diagram Toolbox (click on to display the 'Find Toolbox Item' dialog and specify 'Data Modeling').
2	Drag the 'Procedure' icon onto the diagram.  This generates the Procedure element: 
3	Right-click on the new Procedure element and select 'SQL Object Properties'. The SQL Object Editor screen displays.

SQL Object Editor

The 'SQL Object Editor' dialog is shared by a number of

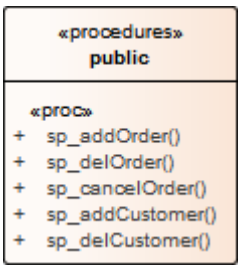
SQL-based database objects (Views, Procedures and Functions); it helps you to manage the various properties of the SQL-based object.

Option	Action
Database	If it has already been set, the default database type displays. If the default has not been set, or you want to change the database type for this Procedure, click on the drop-down arrow and select the target DBMS to model.
Notes	If necessary, type in a comment on the current Procedure.
Definition	Type the full SQL Procedure definition, including the CREATE PROCEDURE syntax. The code editor provides Intelli-sense for basic SQL keywords, functions and names of all objects in the current data model.

Operations in a Container

Database Procedures modeled as operations have a

container object, this being a UML Class with the stereotype «procedures» (with an 's' on the end). Each Database Procedure is an operation with the stereotype «proc». The system provides a dedicated Maintenance window through which you can easily manage the Database Procedures defined as operations.



Database Functions

Database Functions provide you with a mechanism to extend the functionality of the database server. A Database Function is a routine that accepts parameters, performs an action (such as a complex calculation) and returns the result of that action as a value. Depending on the Function, the return value can be either a single value or a result set.

Once created, a Database Function can be used as an expression in an SQL statement.

In Enterprise Architect, Database Functions can be modeled in one of two ways:

- As individual objects (the default method) or
- As Operations in a container

Functionally the two methods result in the same DDL being produced. The main difference is visual - by having several Operations in one container, you have fewer elements and less clutter on the diagram.

Individual objects

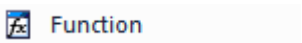
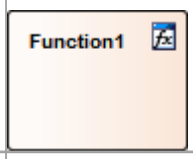
Database Functions modeled as individual objects are UML Classes with the stereotype «function»; you create these either:

- Within the Database Builder or
- By dragging the Function icon from the Data Modeling Toolbox onto a diagram

Add a Database Function using the Database Builder

Step	Action
1	Open the Database Builder.
2	Load or create a Data model.
3	Right-click on the Functions Package and select 'Add New Function'.
4	Overtype the default name with the appropriate name for the Function, and press the Enter key.
5	Double-click on the new Function, or right-click on it and select 'SQL Object Properties'. The SQL Object Editor screen displays.

Add a Database Function to a diagram

Step	Action
1	Open your Data Modeling diagram and, if necessary, display the 'Data Modeling' page of the Diagram Toolbox (click on  to display the 'Find Toolbox Item' dialog and specify 'Data Modeling').
2	Drag the 'Function' icon onto the diagram.  This generates the Function element: 
3	Right-click on the new Function element and select 'SQL Object Properties'. The SQL Object Editor screen displays.

SQL Object Editor

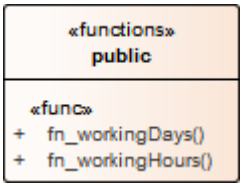
The 'SQL Object Editor' dialog is shared by a number of SQL-based database objects (Views, Procedures and Functions); it helps you to manage the various properties of the SQL-based object.

Option	Action

Database	If it has already been set, the default database type displays. If the default has not been set, or you want to change the database type for this Function, click on the drop-down arrow and select the target DBMS to model.
Notes	If necessary, type in a comment on the current Function.
Definition	Type the full SQL Function definition including the CREATE FUNCTION syntax. The code editor provides Intelli-sense for basic SQL keywords, functions and names of all objects in the current data model.

Operations in a Container

Database Functions modeled as operations have a container object, this being a UML Class with the stereotype «functions» (with an 's' on the end). Each Function is an operation with the stereotype «func». The system provides a dedicated Maintenance window through which you can easily manage the Database Functions stored as operations.


«functions»
public
«func»
+ fn_workingDays()
+ fn_workingHours()

Database Sequences

Sequences are a feature that some DBMS products implement to provide users with a mechanism to generate unique values - the Sequence ensures that each call to it returns a unique value. This is particularly important when the Sequence's result is used as a Primary Key. These can be generated with a schema for loading onto the DBMS server.

Sequences are provided so that database users are not forced to implement their own unique value generator. Not all DBMS products support Sequences; those that do not instead provide functionality for columns to be initialized with an incrementing value.

In Enterprise Architect, Sequences can be modeled in one of two ways:

- As individual objects (the default method) or
- As Operations in a container

Functionally the two methods result in the same DDL being produced. The main difference is visual - by having several Operations in one container, you have fewer elements and less clutter on the diagram.

Individual objects

Sequences modeled as individual objects are UML Classes with the stereotype «dbsequence»; you create these either:

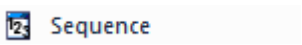
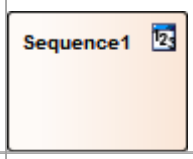
- Within the Database Builder or

- By dragging the 'Sequence' icon from the 'Data Modeling' Toolbox pages onto a diagram

Add a Database Sequence using the Database Builder

Step	Action
1	Open the Database Builder.
2	Load or create a Data model.
3	Right-click on the Sequences Package and select 'Add New Sequence'.
4	Overtype the default name with the appropriate name for the Sequence, and press the Enter key.
5	Double-click on the new Sequence, or right-click on it and select 'SQL Object Properties'. The 'SQL Object Editor' dialog displays.

Add a Database Sequence to a diagram

Step	Action
1	Open your Data Modeling diagram and, if necessary, display the 'Data Modeling' page of the Diagram Toolbox (click on  to display the 'Find Toolbox Item' dialog and specify 'Data Modeling').
2	Drag the 'Sequence' icon onto the diagram.  This generates the Sequence element: 
3	Right-click on the new Sequence element and select 'SQL Object Properties'. The 'SQL Object Editor' dialog displays.

SQL Object Editor

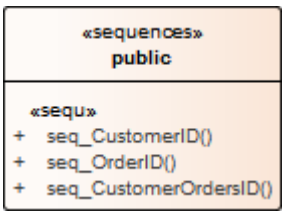
The 'SQL Object Editor' dialog is shared by a number of SQL-based database objects (Views, Procedures and Functions); it helps you to manage the various properties of the SQL-based object.

Option	Action
Database	If it has already been set, the default database type displays. If the default has not been set, or you want to change the database type for this Sequence, click on the drop-down arrow and select the target DBMS to model.
Notes	If necessary, type in a comment on the current Sequence.
Definition	Type the full SQL Sequence definition including the CREATE SEQUENCE syntax. The code editor provides Intelli-sense for basic SQL keywords, functions and names of all objects in the current data model.

Operations in a Container

Database Sequences modeled as operations have a container object, this being a UML Class with the stereotype «sequences» (with an 's' on the end). Each Sequence is an operation with the stereotype «sequ». The system provides a

dedicated Maintenance window through which the modeler can easily manage the Sequences defined as operations.



Database SQL Queries

An SQL Query object provides a convenient mechanism for storing an SQL Statement in the repository, for repeated execution on live database(s).

An SQL Query element is represented in the UML Data Modeling Profile as an Artifact element with the stereotype <<sqlquery>>. You can create these elements either:

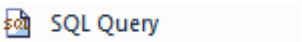
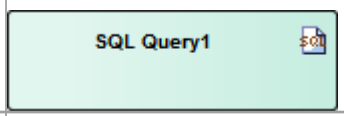
- Within the Database Builder or
- By dragging the 'SQL Query' icon from the 'Data Modeling' Toolbox pages onto a diagram

Add a Database SQL Query using the Database Builder

Step	Action
1	Open the Database Builder.
2	Load or create a Data model.
3	Right-click on the Queries Package and select 'Add New SQL Query'.
4	Overtype the default name with the appropriate

	name for the Query, and press the Enter key.
5	Right-click on the new element and select 'Edit'. The 'SQL Scratch Pad' tab displays, on which you can create the SQL Query statement.
6	When you have finished the SQL statement, click on the Save to SQL Query button in the toolbar to save the changes to the query element.

Add a Database Function to a diagram

Step	Action
1	Open your Data Modeling diagram and, if necessary, display the 'Data Modeling' page of the Diagram Toolbox (click on  to display the 'Find Toolbox Item' dialog and specify 'Data Modeling').
2	Drag the 'SQL Query' icon onto the diagram.  This generates the SQL Query Artifact element: 

3	Double-click on the new element and update the element name and other properties as necessary. To edit the element's SQL statement, access the Database Builder, click on the element in the Queries Package and edit the Query on the 'SQL Scratch Pad' tab.

Create Operation Containers

Whilst the default method of modeling Database Functions, Procedures and Sequences is to create them as individual elements, you can also represent a number of each type of structure as operations of a container Class. You add a stereotype to the Class, which specifies:

- The type of data structure the Class will contain
- The stereotype that will be automatically assigned to each operation created in the Class (for a given data structure, the operations can only be of one stereotype)

Access

Toolbox	Drag the 'Class' icon onto the diagram

Create the Container Class

Step	Action
1	Right-click on the Class element on the diagram and select the 'Design > Element > Editors > Properties

	Dialog' option. The element 'Properties' dialog displays, showing the 'General' tab.
2	In the 'Name' field, type an appropriate name for the container.
3	In the 'Stereotype' field (in the table at the right edge of the dialog) type: • 'functions' for a Database Function container • 'procedures' for a Stored Procedure container • 'sequences' for a Sequence container The 's' at the end of the stereotype name is important.
4	Click on the OK button to save the setting and close the dialog.

Create database structures as operations of the Class

Step	Action
	Click on the Class element on the diagram and press

1	F10. The 'Database <Structure> container: <Classname>' dialog displays.
2	Right-click in the 'Functions' ('Procedures' or 'Sequences') list and select 'Add New <structure>'.
3	In the 'Name' field, type an appropriate name for the operation, such as: • fn_WorkDays • sp_AddOrder or • seq_AddressID
4	In the 'Notes' field type any supporting comments or explanation of the operation. In the 'Function definition' field (or 'Procedure definition', or 'Sequence definition') type the appropriate text.
5	Repeat steps 2 to 4 until you have defined all the operations you require.
6	Click on the list and then on the Close button to close the dialog and show the operations within the Class on the diagram and in the Browser window.

Oracle Packages

Oracle Packages are database objects that are unique to the Oracle DBMS. They are containers that group logically-related objects into a single definition. Packages have two parts - a specification and a body. The:

- Specification section declares the various components
- Body section provides the full definitions of the components

The Package components can consist of Types, Variables, Constants, Exceptions, Cursors and subprograms.

In Enterprise Architect, an Oracle Package is modeled as a UML Class with a stereotype of <<package>>. It has two operations:

- Specification
- Body

For each of these operations the complete SQL syntax is contained in the 'Initial Code' field.

Create an Oracle Package

Step	Action
1	Add a Class element to your data model.

2	Open the Properties window for the element and, in the 'Stereotype' field, type the value 'Package'.
3	Click on the element and press F10, to display the Features window at the 'Operations' page. For the Package specification, press Ctrl+N and create an operation with the name 'Specification' and with no return type.
4	The Properties window displays the properties of the operation; click on the 'Code' tab and type the entire Package specification into the text panel.
5	Return to the Features window at the 'Operations' page and, for the Package body, press Ctrl+N and create an operation with the name 'Body' and no return type.
6	On the Properties window, click on the 'Code' tab and type the entire Package body code into the text panel.

Database Connections

A Database Connection object provides a convenient way of storing the connection details of a live database. Enterprise Architect supports the definition of three different connection types:

- MS Access
- Firebird
- SQLite (introduced in Enterprise Architect v16)
- Native Connection (introduced in Enterprise Architect v16), and
- ODBC

For file based connections (MS Access, Firebird and SQLite) you only have to specify the full path to the database files; for Native connections you will be prompted for the connection details of a database server; for connections of type ODBC you are prompted to select from the list of pre-defined ODBC DSNs on your machine.

Create a Database Connection element

A Database Connection element is represented in the UML Data Modeling Profile as an Artifact element with the stereotype <<database connection>>. You create these either:

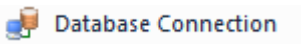
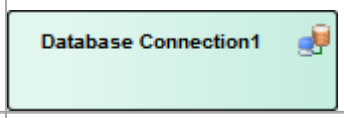
- Within the Database Builder or

- By dragging the 'Database Connection' icon from the 'Data Modeling' Toolbox pages onto a diagram

Add a Database Connection using the Database Builder

Step	Action
1	Open the Database Builder.
2	Load or create a Data model.
3	Right-click on the Connections Package and select 'Add New DB Connection'.
4	Overtype the default name with the appropriate name for the Connection, and press the Enter key.
5	Double-click on the new Connection, or right-click on it and select 'DB Connection Properties'. The 'Database Connection Properties' dialog displays.

Add a Database Connection to a diagram

Step	Action
1	Open your Data Modeling diagram and, if necessary, display the 'Data Modeling' page of the Diagram Toolbox (click on  to display the 'Find Toolbox Item' dialog and specify 'Data Modeling').
2	Drag the 'Database Connection' icon onto the diagram.  This generates the Database Connection element: 
3	Double-click on the new element. The 'Database Connection Properties' dialog displays.

Database Connection Properties

Option	Action

DBMS Type	Click on the radio button for the appropriate type: • MS Access file based database • Firebird file based database • SQLite file based database • Direct Native connection, or • ODBC based database The 'Save Password?' checkbox is only enabled for ODBC connection types, and indicates if Enterprise Architect should store the password for the selected ODBC DSN. The checkbox defaults to selected; that is, passwords are saved. While all connection passwords are encrypted before being saved, there can be occasions when data modelers want to restrict access to only users that have the required permissions.
Filename/DSN	If you have selected a 'DBMS Type' of MS Access or Firebird, type in or browse for the location and name of a physical file. If the file does not already exist it will be created. If you have selected a 'DBMS Type' of ODBC, type in or select a defined ODBC DSN. Depending on the DBMS, you might be prompted for other details such

	as server, connection user ID and password.
Other Schemas	This field acts as a schema filter to limit the number of objects returned by enquiries made against the ODBC connection. Entering a value in this field is particularly important for Oracle databases to reduce the time it takes for making connections to the database, due to the large number of system objects. If you need to enter multiple schemas to be filtered on, separate them with commas.
OK	Click on this button to save the changes you have made.

Delete connection

If a connection is no longer required, you can delete it as for any other element from the Database Builder, the Browser window or a diagram. Right-click on the element and select the corresponding 'Delete <element name>' option.

Notes

- It is advisable that when working in a team environment (that is, multiple users sharing a single Enterprise Architect repository) all ODBC based Database Connection objects are defined as 'DSN-less' so that the Database Connection object contains all necessary details and can therefore be shared between all users, although a Native Connection does this and is easier to setup
- The DBMS type of a Database Connection object cannot be changed once the initial selection has been saved

Manage DBMS Options

Using the 'Manage DBMS Options' dialog, you can quickly change the DBMS Type and/or Owner of an individual database object or several objects within an individual Package or Package hierarchy. You can also create bulk Foreign Key Indexes on all Foreign Keys that do not already have an index.

Access

Ribbon	Design > Package > Manage > DBMS Options Develop > Data Modeling > Database Builder > Right-click on the required database | Load | right-click on the root node | Manage DBMS Options

Options

Option	Action
Package	Displays the name of the Package in the

	Browser window that you are currently working on. If necessary, click on the ... button and select a different Package using the Navigator window (a version of the 'Find Package' dialog).
Include Objects in Child Packages	Select this checkbox to include all the database objects in all sub-Packages. Selecting or deselecting this control will immediately refresh the list of objects.
List of Objects	This list control will display the names of all objects in the current Package (or Package hierarchy) along with its allocated DBMS and owner. By default every object has its checkbox selected whenever the list is loaded or refreshed.
All	Click on this button to select all deselected checkboxes in the 'List of Objects'.
None	Click on this button to deselect all selected checkboxes in the 'List of Objects'.
Change DBMS	Select this checkbox if you want to change the assigned DBMS of objects in the Package. Provide values for the

	'Current DBMS' and 'New DBMS' fields in order to continue. The 'Current DBMS' drop-down list includes the option '<All>', which changes several different DBMS values all to the new value. Note: When performing this function, the data types of all Table columns are automatically converted to the closest match for the selected DBMS; therefore, you should perform a manual review of the data types after running the process.
Change Owner	Select this checkbox if you want to change the Owner of the selected objects in the 'List of Objects'. Specify the current Owner in the 'Current Owner' field in order to continue. Leaving the 'New Owner' field blank will remove the Owner property of all selected objects.
Create Indexes on Foreign Keys	Select this checkbox to create an index on all Foreign Keys in the Package, where one does not already exist.
OK	Click on this button to start the update process. The button is disabled unless at least one object in the list and one of the update options are selected.

Data Types

Every Table column that you define in your data model has a data type assigned that specifies the type of information that can be stored by the column. The available datatypes for a column are dependent on the selected DBMS for the Table, because each DBMS supports its own list of datatypes. Whilst each DBMS supports the same basic types, such as string, whole or decimal numbers, each DBMS calls them by different names and have different properties.

Each Enterprise Architect repository contains the definitions of the core datatypes for a number of standard DBMS products. However, since data types vary from one DBMS product to another, and from one version of a product to another, Enterprise Architect provides you with tools to:

- Define new data types for a new version of your DBMS product
- Define data types for a new, non-standard database product
- Automatically convert data types from one defined DBMS product to another
- Import and export datatypes between repositories

Map Data Types Between DBMS Products

Whilst modeling physical data models provides a great deal of detail about all Tables and their columns, this level of detail does make it harder to change the target technology or platform. For example, after reverse engineering your database into a physical data model, you must remap the data types before generating the schema for the new DBMS product.

Enterprise Architect provides a set of default mappings for standard, supported DBMS products, to help you automate the conversion process.

However, you might want to customize the default mappings to suit your specific project requirements, or where the mapping of one data type to another is not currently defined. For example, in your migration from one DBMS platform to another, one of the platforms might be non-standard or otherwise not supported by Enterprise Architect.

Access

Ribbon	Settings > Reference Data > Settings > Database Datatypes : Datatype Map

Database Data Types Mapping

Repeat this process for each of the data types to map.

Once you are satisfied with the data type mappings, you can convert either individual Tables or an entire Package of Tables to the new target DBMS product.

Field/Button	Action
From Product Name	Click on the drop-down arrow and select the DBMS product to map data types from.
Defined Datatypes for Databases	Displays all the defined data types for the product and, where appropriate, their sizes and values. Click on the data type to map - this must have a defined size unit and value. The 'Datatype' and 'Common Type' fields under the 'From Product Name' field display this data type.
To Product Name	Click on the drop-down arrow and select the DBMS product to map data types to. The 'Datatype' and 'Common Type' fields under this field display the values corresponding to those in the fields for

	the 'From' product.
Size	Click on the radio button for the appropriate size unit and type the default values in the corresponding data fields.
Save	Click on this button to save the mapping.

DBMS Product Conversion for a Package

Using the DBMS Package mapper, you can automatically convert a Package of database Tables from one supported DBMS type to another supported DBMS type. You can also change the DBMS type for individual Tables.

If one of the DBMS types is non-standard or otherwise not supported by Enterprise Architect, you should check that the mapping of datatypes from one DBMS type to the other has been defined.

Access

Ribbon	Design > Package > Manage > DBMS Options Develop > Data Modeling > Database Builder > Right-click on the required database | Load | right-click on the root node | Manage DBMS Options

Map the DBMS data types of a Package to the data types of another DBMS

Field/Button	Action
Include Objects in Child Packages	If there are objects in child Packages that also require changing, select the checkbox.
Change DBMS	Select the checkbox.
Current DBMS	Click on the drop-down arrow and select the current DBMS.
New DBMS	Click on the drop-down arrow and select the target DBMS.
OK	Click on this button to map all Tables in the selected Packages to the new DBMS.

Data Type Conversion For a Table

Once a database schema has been set up on an Enterprise Architect diagram (either by importing through ODBC or manually setting up the Tables), the DBMS can be changed to another type and the column datatypes are mapped accordingly for each Table.

You might use this procedure if you have copied a small number of Tables into the project from elsewhere, but if you have many Tables you can also convert all of them at once within their parent Package.

If one of the DBMS types is non-standard or otherwise not supported by Enterprise Architect, you should check that the mapping of datatypes from one DBMS type to the other has been defined.

Map the DBMS type of a Table to another DBMS type

Step	Action
1	Double-click on the Table element in a diagram. The Table 'Properties' dialog displays, with the 'Database' field showing the current DBMS for this Table.

2	To map the datatypes to another DBMS, click on the 'Database' drop-down arrow and select the target DBMS.
3	Click on the Apply button. The datatypes are converted to match those of the new DBMS, and these are reflected in any DDL generated from this Table.

Database Datatypes

Using Enterprise Architect's 'Database Datatypes' dialog, you can add to the set of data types that are available for a particular DBMS. You can:

- Identify the DBMS in use and, if required, set this as the model default
- Include any new data types that are supported by later versions of the DBMS and not yet included with Enterprise Architect
- Remove any previously-added data types that are no longer relevant
- Add a new DBMS product and its built-in data types if, for example, you want to create a physical data model for a DBMS product that is not yet supported natively by Enterprise Architect

Access

Ribbon	Settings > Reference Data > Settings > Database Datatypes or Develop > Data Modeling > Datatypes

Manage Datatypes

You can transport these database data types between Enterprise Architect models using the 'Export Reference Data' and 'Import Reference Data' options.

Field/Button	Action
Product Name	Click on the drop-down arrow and select an existing DBMS. Once a product is selected, all defined data types will be shown in the 'Defined Datatypes for Databases' list.
Add Product	If your DBMS is not listed, click on this button to add it. An 'Input' prompt displays, in which you type the DBMS name; click on the OK button to add the name to the drop-down list.
Set as Default	Select the checkbox to set the selected DBMS as the default for your database engineering and modeling. Once you set the default database, when you create any new Table elements the database type is automatically pre-set to this default. You can also set the default database type

	in the second data entry field of the Code Generation toolbar.
New	Click on this button to clear the data type fields on the dialog so that you can define another data type.
Datatype	Type a name for the data type.
Size	Select the appropriate radio button for the required size and, if appropriate, specify the default and maximum values: • None – for data types without a size component, such as INT • Length – for data types that require a single size that defines the Length, such as VARCHAR(10) • Precision & Scale – for data types that require two numeric values, such as DECIMAL(18,2)
Common Type	Click on the drop-down arrow and select the generic name of each data type. This is used when a Table's DBMS is changed.
Save	Click on the button to immediately save your data type to the repository (and add it to the 'Defined Datatypes for Databases' list).

Defined Datatypes for Databases	This panel lists the data types currently defined for the selected DBMS, either system-supplied or user-defined.
Delete	Select a data type in the 'Defined Datatypes for Databases' list and click on this button to remove the data type.
Datatype Map	If you have changed the DBMS or technology for which you have defined the data types from or to an unsupported DBMS type, click on this button to define how to automatically remap the data types to your new DBMS or technology.

MySQL Data Types

MySQL supports the ENUM and SET data types, which must be added to your Enterprise Architect model before you can use them as the types for columns.

Access

Ribbon	Settings > Reference Data > Settings > Database Datatypes

Add the ENUM and SET data types for MySQL

When using these data types later in a column's 'Initial' field, type the values as a comma-separated list, in the format:

('one','two','three')

If one value is the default, use the format:

('one','two','three') default 'three'

Step	Action

1	The 'Database Datatypes' dialog displays.
2	In the 'Product Name' field select 'MySQL'.
3	Add the data types ENUM and SET.

Oracle Data Types

The Oracle data types NUMBER and VARCHAR have additional properties that you can model.

Access

Ribbon	Settings > Reference Data > Settings > Database Datatypes

Data Types

Data Type	Detail
NUMBER	The NUMBER data type requires precision and scale properties. The 'Precision' and 'Scale' fields are displayed on the 'Attributes' page of the Features window when the data type is set to NUMBER; if you enter information into these fields, it is displayed on your diagrams. For example:

	create NUMBER by setting 'Precision' = 0 and 'Scale' = 0 create NUMBER(8) by setting 'Precision' = 8 and 'Scale' = 0 create NUMBER(8,2) by setting 'Precision' = 8 and 'Scale' = 2
VARCHAR	Oracle VARCHAR2(15 CHAR) and VARCHAR2(50 BYTE) data types can be created by adding the Tagged Value LengthType with the value CHAR or BYTE.

Data Modeling Settings

Enterprise Architect provides data modeling settings that can be used to configure the way database systems are modeled in Enterprise Architect. These include the ability to define the data modeling language, which determines the way that connectors are displayed, and settings to configure the naming of Primary Keys, Foreign Keys and Indexes. The settings are global and will affect any Enterprise Architect repository.

Access

Ribbon	Start > Appearance > Preferences > Preferences > Source Code Engineering > Code Editors > DDL

DDL Editor

In this field you browse for the full execution file path and name of an external program that Enterprise Architect should use to open files that are created by its Generate DDL functionality. If you leave this field empty, Enterprise Architect uses the default code editor.

Default Database

In this field you select the DBMS that will be automatically assigned to database objects that are created outside a Data Model workspace (see the *Create a Data Model from Model Pattern* Help topic).

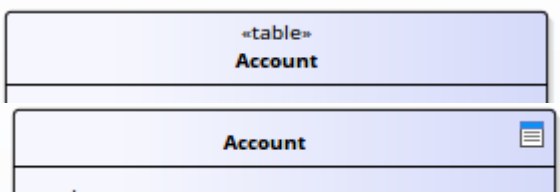
MySQL Storage

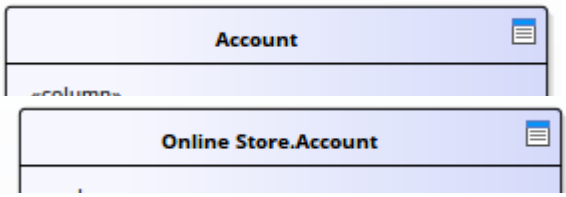
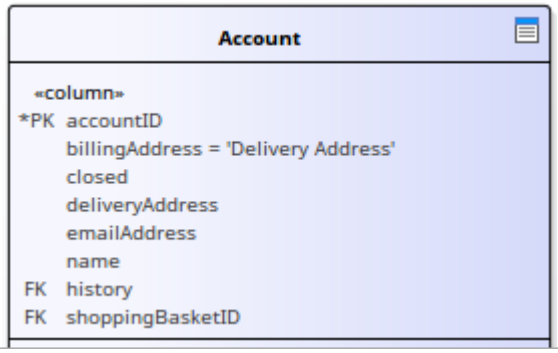
In this field you select the default storage engine to be assigned to MySQL Tables; from MySQL v 5.5 onwards the default value is InnoDB.

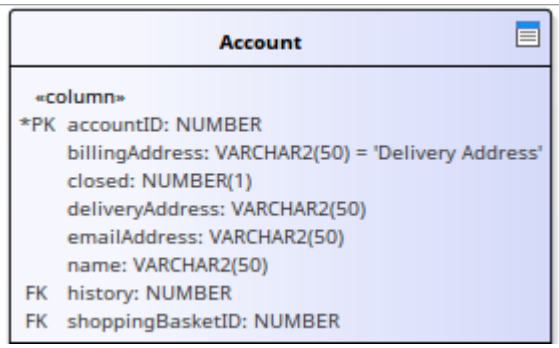
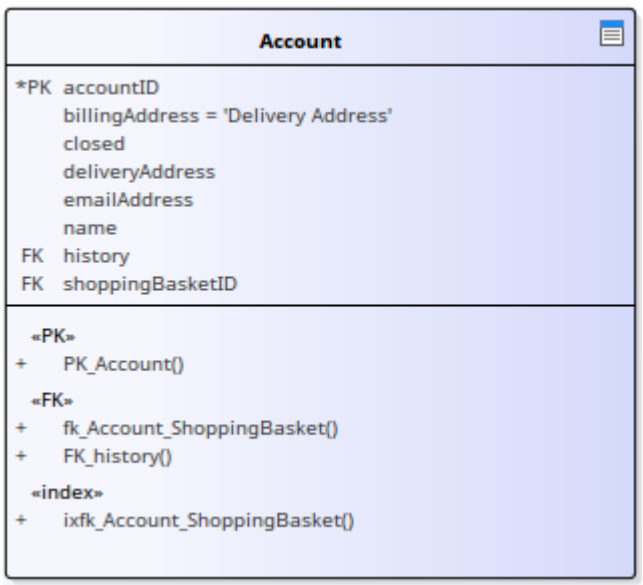
Data Modeling Notations

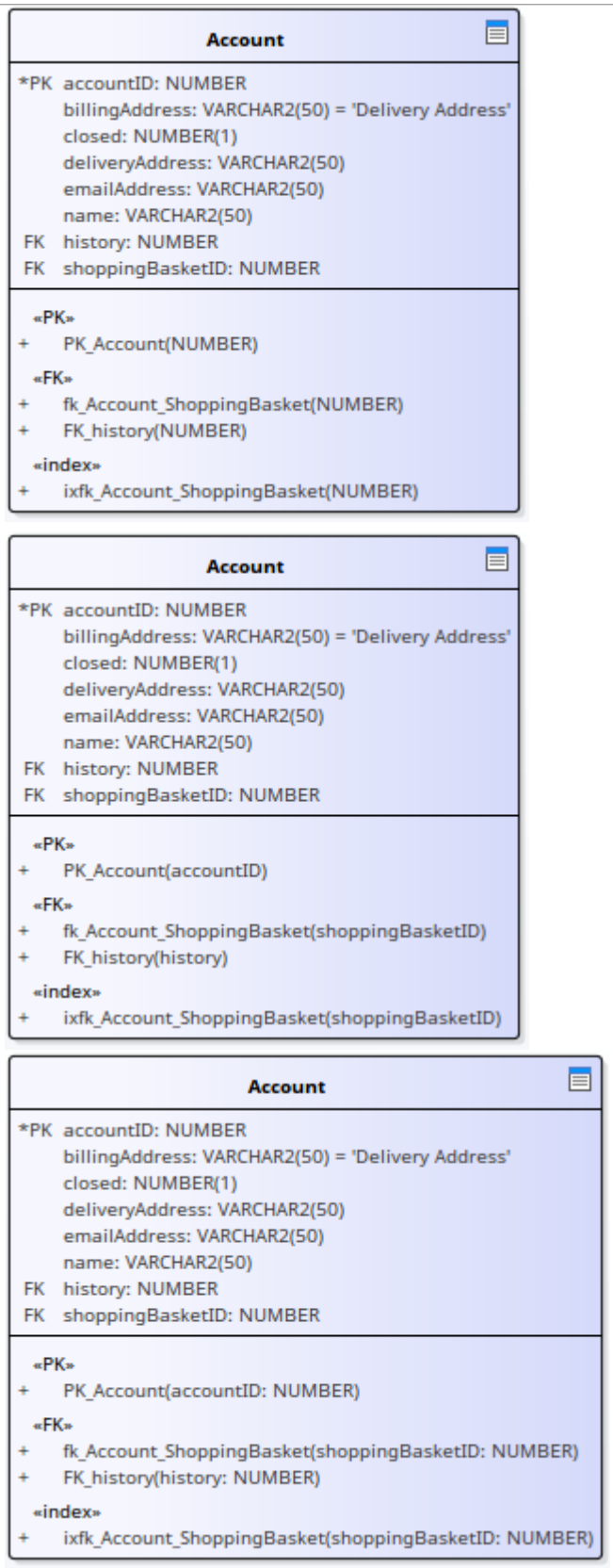
Enterprise Architect supports numerous settings related to data modeling that can influence how database objects are represented on diagrams. These settings, and how they can affect the representation of database objects, are described here.

Settings

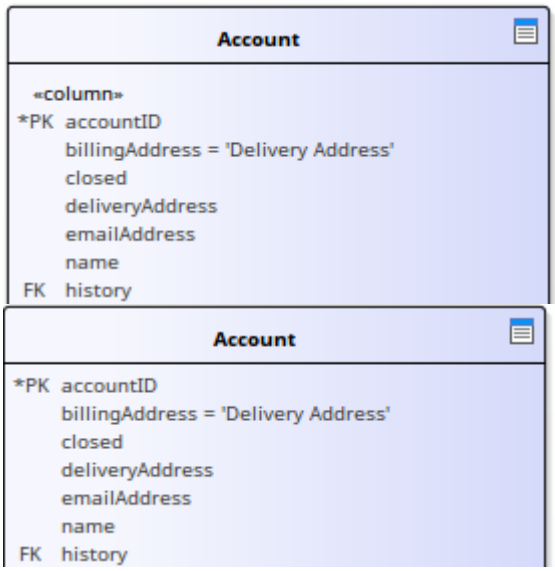
Setting	Detail
Stereotype Icons	Access: 'Design > Diagram > Manage > Properties > Elements : Use Stereotype Icons' Default Value: True Enterprise Architect provides a diagram-level setting for the display of stereotyped objects. When the checkbox is selected, database objects on the diagram are displayed with an icon representing their stereotype instead of the stereotype name. 

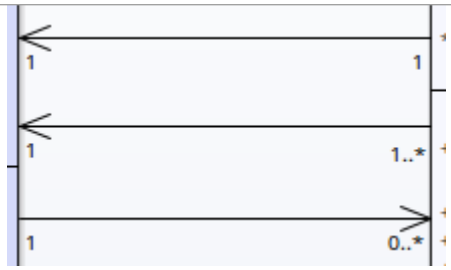
Show Data Model Owner	Access: 'Design > Diagram > Manage > Properties > Elements : Show Data Model Owner' Default Value: True The system provides a diagram-level setting for the display of Owners. When the checkbox is selected, database objects on the current diagram will be displayed with their full name '{Owner.}ObjectName'. 
Show Column Details	Access: 'Design > Diagram > Manage > Properties > Features : Show Attribute Detail' Default Value: Name Only The system provides a diagram-level setting for the display of Table column names and datatypes. The available options are: 'Name Only' or 'Name and Type'. 

	
Show Involved Column Details	Access: 'Design > Diagram > Manage > Properties > Features | Show Parameter Detail' Default Value: Type Only The system provides a diagram-level setting for the display of columns involved in a Table constraint. The available options are: 'None', 'Type Only', 'Name Only' and 'Full Details'. In these examples, the Primary Key (PK) constraint 'PK_account' involves the column 'accountID'. 

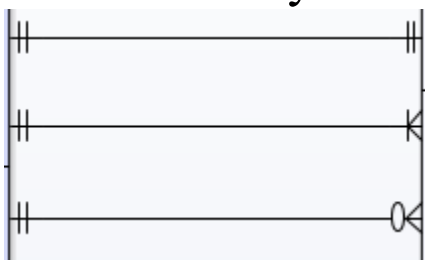


Show Column	Access: Start > Application > Preferences > Preferences > Objects: Show

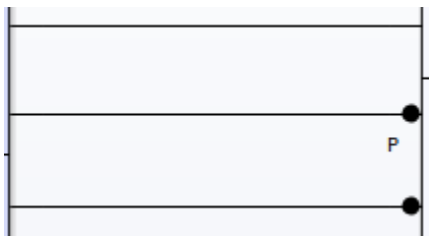
Stereotype	<<column>> stereotype Default Value: True Enterprise Architect provides a global-level setting that controls whether or not the <<column>> stereotype is displayed above each Table's columns. You can therefore hide the stereotype if you prefer, considering that attributes with a stereotype of <<column>> are the only valid option for Tables. 
Connector Notation	Access: 'Design > Diagram > Manage > Properties > Connectors : Connector Notation' Default Value: UML 2.1 Enterprise Architect supports three diagram notations for data modeling: • UML 2.1 - the standard UML 2.1 notation for connectors



- Information Engineering - the Information Engineering (IE) connection style



- IDEF1X - the Integrated Definition Methods IDEF1X connection style



(These are the same three connectors using the different notations.) The default notation for the Data Modeling diagram is 'Information Engineering', whilst the default notation for models created from Database Engineering Patterns is 'IDEF1X'.

DDL Name Templates

At various times during the process of data modeling, Enterprise Architect is required to automatically generate Table constraints. The naming standard for these generated constraints is defined in and applied by the DDL Name Templates, which you are free to change at any time. These Name templates are defined at repository level, so whenever they are changed all users of the repository will use the new templates.

Access

Ribbon	Settings > Model > Options > Source Code Engineering : DDL Name Templates ...

DDL Name Templates

Option	Action
Primary Key	Define the name template used when Primary Key constraints are created.

Unique Constraint	Define the name template used when Unique Constraints are created.
Foreign Key	Define the name template used when Foreign Key constraints are created.
Foreign Key Index	Define the name template used when Foreign Key indexes are created.
Save	Click on this button to save the name template(s) you have defined.

Template Macros

These recognized macros will be replaced by name text during the creation of a constraint name.

Macro	Applies to
%tablename%	Primary Key Unique Constraint Description: The string that is replaced by the Table's name.
%columnnam	The string that is replaced by the

e%	constraint's column name(s).
%primarytablename%	Foreign Key Description: The string that is replaced by the primary (parent) Table's name.
%foreigntablename%	The string that is replaced by the foreign (child) Table's name.
%foreignkeyname%	Foreign Key Index Description: The string that is replaced by the Foreign Key name.

Import Database Schema

The power of model-based engineering is the ability to visualize, analyze and design all aspects of a system. Being able to view the database schemas alongside other models of a system provides great clarity and reduces the chance of error. Enterprise Architect can reverse engineer a DBMS schema and its objects into a model under a number of different standards, including UML, Information Engineering and IDEF 1X. A wide range of database objects are supported including Tables, Views, Procedures, Functions and Sequences. Enterprise Architect achieves this by interrogating the DBMS's information schema and importing the definition into a UML objects. As modifications are made to the Live database the changes can be synchronized into the model.

Once the schema is in Enterprise Architect, the database objects can be traced to other elements, ensuring the integrity of design and architecture. When systems target multiple DBMSs, these can all be reverse engineered into a model and elements and datatypes can be compared between these models. The sophisticated reporting engine can produce high quality documentation, including data dictionaries, diagrams and relationships back to other models such as architecture and information requirements, and ultimately to business goals and drivers.

Database schema information can be imported via the Database Builder (recommended) or from the 'Develop'

ribbon.

Import Database Schema

Step	Action
1	Open the Database Builder (Develop > Data Modeling > Database Builder)
2	Load or create a Data Model.
3	• Right-click on the loaded Data Model in the Database Builder and select 'Import DB schema' or • From the ribbon select 'Develop > Data Modeling > Import' The 'Import DB Schema' dialog displays, showing the details of the current active database connection.

The Import DB Schema dialog

Option	Description
	This field shows a description of the

Database	current Live connection, in the format: dbms.database_server.database_name If necessary, click on the  button and, select an alternative connection.
Import to	This field shows the target Package that the new objects will be saved to. If you want to specify a different Package, click on the  button and select an alternative Package.
Only include objects from Schema(s)	If the database type supports multiple schemas (such as SQL Server, Oracle, PostgreSQL and DB2 Express) you can filter objects to be retrieved from the database by schema. The available schemas are automatically listed in this panel. Select the checkbox against each schema to include in the import. (You can click on the All button to select all the schemas, or the None button to clear all selected checkboxes.) If you suspect that the schema list might have changed since you loaded them, you can refresh the list by clicking on the Reload Schemas button.

Name Filter	The 'Name Filter:' field allows filtering of objects using SQL wildcards appropriate to the DBMS of the schema being imported. For example, for Oracle: • LIKE 'A%' - list objects with a name starting with the letter 'A' • NOT LIKE '%_%' ESCAPE '\' - list objects with a name that does not include an underscore (_) • IN ('TABLE1','TABLE2') - list objects with names that are included in the parentheses • NOT IN ('TABLE1','TABLE2') - list objects with names that are not included in the parentheses Note that only one filter can be entered. You cannot add a second filter using the AND clause. Filtering is not available for MS Access
Filter Options	The 'Filter Options' panel controls what object types and properties are read in from the database schema. Values changed on this screen are saved to the registry so that they are re-applied in the next work session. The available options are briefly described here; select the

	checkbox against an option to activate it. **Tables** • Tables - Select to import Tables • Table Primary Keys - Select to import Primary Key definitions on Tables • Table Foreign Keys - Select to import Foreign Key definitions on Tables • Table Indexes - Select to import Table Indexes • Unique Constraints - Select to import Unique Constraint definitions on Tables • Check Constraints - Select to import Check Constraint definitions on Tables • Table Triggers - Select to import Trigger definitions on Tables • Table Properties - Select to import extended Table properties • Constraint Properties - Select to import Constraint Properties for Tables • Length Semantics - Select to import length semantic definitions on Oracle string columns **Objects** • Views - Select to import Views • Procedures - Select to import Procedures

	- As Operations - Select to import Procedures as operations (methods) of a single Class; you can view and edit them through the Database object container 'Properties' dialog (the option defaults to unselected, where the selected items are imported as separate Classes) • Functions - Select to import Functions - As Operations - Select to import Functions as operations (defaults to unselected) • Sequences - Select to import Sequences - As Operations - Select to import Sequences as operations (defaults to unselected) • Package - Select to import Oracle Packages **Advanced** • System Objects - Select to import system Tables, Views and other system objects **Warning**: With the 'As Operations' option for Procedures, Functions and Sequences, if objects have been imported under one setting (selected or unselected) and then you change the setting and import further objects, the objects imported under the first setting are

	removed.
Synchronization	Select the appropriate radio button to indicate whether the existing Classes are to be updated, or the database objects imported as new objects. If you select the 'Synchronize existing classes' option, also select the appropriate checkboxes to determine whether model comments, column default values and/or Table constraints are to be retained or overwritten with the comments, values and constraints of the imported objects.
Import To	Select the appropriate radio button to indicate whether to update the Package and currently-open data model diagrams, or just the Package. If no diagrams are open, the 'Package Only' radio button defaults to selected and the options are disabled; if the open diagrams are in the selected Package, you can select either option.
Import	Click on this button to start the import. The 'Select Database Objects to Import' dialog displays, listing all the database objects found that match the selection criteria.

	Select the checkbox against each schema (or object type) to automatically select all objects in that group or to import each object individually. Click on the All button to select all types and objects, or on the None button to clear all selected checkboxes. When you have selected all the objects to import, click on the OK button to continue the import.

Notes

- Within Windows, ODBC DSN can be defined for either 32 or 64 bit applications, therefore care must be taken to ensure that all ODBC DSNs for Enterprise Architect's use are defined sharing the same architecture. This is particular important from Enterprise Architect version 16 onwards because it is now available in both 32 and 64 bit versions. An alternative solution (and what Sparx Systems recommend) is to make use of Native connections, since they work for both architectures.
- The ODBC connection should use the ODBC driver available from the DBMS vendor, such as MySQL's ODBC driver for MySQL, and Oracle's ODBC driver for Oracle; drivers provided by third-party vendors are not

supported, including the Microsoft ODBC driver for Oracle

- You can import a suitable ODBC driver for SQLite from http://www.ch-werner.de/sqliteodbc/
- Due to the limitations of SQLite, round tripping of SQLite Table and column comments is not possible; to retain comments entered in an SQLite data model when importing from ODBC, deselect the 'Overwrite Object Comments' checkbox in the 'Synchronization' section of the 'Import DB Schema from ODBC Source' dialog
- If setting up an ODBC connection for reverse engineering, the default settings are sufficient
- If you are importing database schema from an MS Access Jet 4.0 database, check that you have selected the 'Use Jet 4.0' checkbox on the 'General' page of the 'Preferences' dialog (the 'Start > Appearance > Preferences > Preferences' ribbon option), otherwise the Jet 3.5 routines are loaded; you must restart Enterprise Architect after selecting the checkbox
- The list of Data Modeling Data types is defined as static data (in each repository), so depending on the age of your repository, there could be additional data types available from the 'Data Modeling Data Types' section of the 'Resources' page on the Sparx Systems website

Generate Database Definition Language (DDL)

Once a physical model has been defined and the objects modeled, Enterprise Architect can generate Database Definition Language (DDL) for a variety of objects including database Tables, Views, Functions, Sequences and Procedures. This is a time saving mechanism and reduces the errors that can be introduced by doing this by hand in other tools. Forward engineering is governed by a set of templates that define how UML constructs are converted to the objects in the targeted DBMS. Standard templates are provided for all supported DBMSs, and these can be edited to customize the way the DDL is generated. In the case that a DBMS is not supported out-of-the-box, a new set of templates can be created using the existing ones as a starting point and reference.

When forward engineering DDL, the output can be directed to a file (or a series of files, one for each object) or to the DDL execution engine. The execution engine allows you to execute the DDL immediately, targeting a live database through the active connection. If you direct the output to a file you can execute the DDL against a live database later, at your convenience. The generated files can be opened using the code editor, by selecting F12, Ctrl+E or Alt+7, allowing you to view the DDL inside Enterprise Architect.

Generate DDL For Objects

As you create your database model, you can generate the DDL for an individual object, a Package of objects or the complete data model. The only difference is how you invoke the generate DDL process.

Access

Open the Database Builder window, then use the context menu and select 'Generate DDL'.

Ribbon	Develop > Data Modeling > Database Builder > Click on an object, Package or Data Model node : Generate DDL

Generate Tab

Field/Button	Action
Package	Click on the  button and browse for the Package for which you want to generate DDL, using the Navigator window (a version of the 'Find Package' dialog).

	(Note: This field might not be displayed in all situations.)
Include All Child Packages	Select this checkbox to include the objects in sub-Packages in the 'Select Objects to Generate' list.
Delete Target Files	When objects are generated to single files, the full filename is stored with the object, and displayed in the 'Target File' column of the 'Select Objects to Generate' list. Click on this button to remove all the existing filenames and prompt for new ones.
Select Objects to Generate	This field displays the list of objects that DDL will be generated for, in the displayed order. If you need to change this order to resolve object dependencies, click on an object to move and click on the  buttons to move that object one position up or down in the sequence. Select each object for which to generate DDL. Click on: • The All button to select every item • The None button to clear all selections • Each of several objects while you press Ctrl, to select a number of individual

	objects • The first and last objects in a block while you press Shift, to select every object in the block
Save Generated Order	If you have changed the order in which the objects are listed, select the checkbox to save the new sequence when you click on the Generate button.
Refresh	Reload the list of objects, restoring each object to their previous positions (if object positions have been changed).
Single File	Select this radio button if you want to save the generated DDL to a single file. Click on the  button to browse for the file path and file name.
Individual file for each table	Select this radio button if you want to save the DDL generated for each object to a separate file. When you click on the Generate button, the system prompts you for the target file name for each object in turn (if it is not specified already).
Generate to DDL	Select this radio button if you want to save the DDL to the execution engine

Execution Engine	(the 'Execute DDL' tab of the Database Builder). The DDL Execution Engine provides the facilities for executing the generated SQL script and responding to errors in execution immediately, without having to create an external file and load it into another tool. 'Generate to DDL Execution Engine' is the default option if the Database Builder is open.
Generate	Click on this button to run the Generate DDL process with the options you have selected.
View	If you have generated the DDL to a single external file, click on this button to view the output. By default Enterprise Architect uses the default code editor. However, you can define an alternative default DDL editor on the 'Preferences' dialog ('Start > Application > Preferences > Preferences > Source Code Engineering > Code Editors > DDL').
Close	Click on this button to close the dialog. If you did not generate the DDL, this button

	also abandons DDL generation for the object.

Options Tab

Set any of these flags to False if you do not want to take the action they initiate.

Group	Options
Table Generation Options	**Tables** - indicates that DDL for Table elements should be generated (*) **Primary Keys** - indicates that DDL for Primary Keys should be generated ($) **Foreign Keys** - indicates that DDL for Foreign Keys should be generated ($) **Indexes** - indicates that DDL for Indexes should be generated ($) **Unique Constraints** - indicates that DDL for Unique Constraints should be generated ($) **Check Constraints** - indicates that DDL for Check Constraints should be generated ($) **Table Triggers** - indicates that DDL for Table Triggers should be generated ($)

	Table properties - indicates that DDL for extended table properties should be generated ($) **Length Semantics** - indicates that DDL for Oracle Length Semantic should be generated ($)
Object Generation Options	**Views** - indicates that DDL for View elements should be generated (*) **Procedures** - indicates that DDL for Procedure elements should be generated (*) **Functions** - indicates that DDL for Function elements should be generated (*) **Sequences** - indicates that DDL for Sequence elements should be generated (*) **Packages** - indicates that DDL for Oracle Packages elements should be generated (*)
Formatting	**Include pre/post queries** - indicates that the generated DDL should include the SQL statements defined in the '_PreStatements' and '_PostStatements' SQL Queries **Include Owners** - indicates that the generated DDL should include the

	schema/owner of all elements **Include Comments** - indicates that the generated DDL should include any comments **Include Header Comments** - indicates that the generated DDL should include any header comments (#) **Include Object Comments** - indicates that the generated DDL should include any object (such as Table or View) comments (#) **Include Column Comments** - indicates that the generated DDL should include any columns comments (#) **Generate DROP statements** - indicates that the generated DDL should include the DROP statement for objects **Use Database** - indicates that the generated DDL should include a USE Database statement **Use Alias** - indicates that the generated DDL makes use of any object or column aliases **Separate Constraint from Table** - indicates that the generated DDL should define the creation of constraints as separate statements from the Table definition

	Include NULL in column definitions - indicates that the generated DDL should apply the NULL keyword to each column definition that is defined as nullable; that is, columns with their 'NOT NULL' flag unchecked (this option only applies to the DBMSs that support the 'NULL' syntax)

Notes

- (*) - options with this mark will be automatically set to True if you have specified to generate DDL for an individual element of that type; that is, if you select a Table and your 'Generate Table' option is False, Enterprise Architect will change the option to True
- ($) - options with this mark will be disabled if the 'Tables' option is set to False
- (#) - options with this mark will be disabled if the 'Include Comments' option is set to False

- In the Corporate, Unified and Ultimate Editions of Enterprise Architect, if security is enabled you must have 'Generate Source Code and DDL' permission to generate DDL
- For a PostgreSQL database, you must set the 'Sequences' option to True to enable auto increment columns to be

created

- If generating Oracle sequences, you must always set the 'Table Triggers' and 'Sequences' options to True, so that a pre-insert trigger is generated to select the next sequence value to populate the column; also, in the column properties, set the 'AutoNum' property to True
- You can edit the DDL templates that the system uses to generate the DDL; these are stored at the repository level so that all other users of the same repository will automatically use the updated templates

Edit DDL Templates

The DDL Template Editor provides the ability to change the templates that the system uses while generating DDL from a data model. It applies the facilities of the Common Code Editor, including Intelli-sense for the various macros. For more information on Intelli-sense and the Common Code Editor, see the *Editing Source Code* Help topic.

Access

Ribbon	Develop > Data Modeling > Templates

Select and Edit Templates

Option	Action
Language	Click on the drop-down arrow and select the database type (Database Management System).
New Database	Click on this button to create a new set of templates for a non-standard DBMS.

	The 'Input' dialog displays, on which you type the name of the new DBMS for which you are creating templates. This updates the 'Language' field.
Template	Displays the contents of the selected template, and provides the editor for modifying these contents.
Templates	Lists the base DDL templates, Click on a template name to display and edit the template contents; the name of the selected template is highlighted. The 'Modified' field indicates whether you have modified the default template originally supplied with the system.
Stereotype Overrides	Lists any stereotyped templates that exist for the currently-selected base template. The 'Modified' field indicates whether you have modified a default stereotyped template.
Add New Custom Template	Click on this button to display the 'Create New Custom Template' dialog, on which you select the template type from a drop-down list, and type in a name for the template. The template type becomes a prefix for

	the name; for example: Namespace_MyDDLTemplate
Add New Stereotyped Override	Select a base template and click on this button to display the 'New Template Override' dialog for adding a stereotyped template for the selected template. From the drop-down lists, select the Class and/or Feature stereotype for which to apply the override template.
Get Default Template	Click on this button to refresh the editor display with the default version of the selected template. (This does not delete the changed version of the template.)
Save	Click on this button to overwrite the selected template with the updated contents of the Template panel.
Delete	If you have overridden the selected template, click on this button to delete the overridden template and replace it with the corresponding default DDL template.

Notes

- User-modified and user-defined DDL Templates can be imported and exported as Reference Data (see the *Sharing Reference Data* topic)
- Any user-defined templates for a database type are listed in the 'Export Reference Data' dialog in the 'Code, DDL, Transformation & CSV Templates' table, identified by the DBMS name with the suffix _DDL_Template - if no user-defined templates exist for a DBMS, there is no entry for the DBMS in the dialog
- You must also define any appropriate data types for the DBMS and, if exporting the templates as Reference Data, you must export the 'Model Data Types - Code and DDL' table as well

DDL Template Syntax

DDL Templates are written using Enterprise Architect's Code Template Framework, but they have been extended to support DDL generation.

DDL Template Development

These aspects of DDL Template development are discussed in this section.

Aspect	See also
DDL Templates	DDL Templates
DDL Macros	DDL Macros
DDL Function Macros	DDL Function Macros
DDL Property Macros	DDL Property Macros
DDL Options in Templates	DDL Options in Templates

DDL Templates

The DDL Template Editor operates in the same way as the Code Template Editor, except that the DDL Template Editor displays templates for DDL Generation and templates for Alter DDL Generation at the same time. The Alter DDL Generation templates are shown at the bottom of the list, prefixed by 'DDL Diff'.

Base Templates for DDL Generation

The DDL Template Framework consists of a number of base templates for DDL Generation. Each base template generates a DDL statement (or a partial statement) for a particular aspect of the UML data model.

Templates

This table lists and briefly describes the base templates used for DDL generation.

Template	Description
DDL Check Constraint	Invoked by the DDL Table Constraint template to generate the Check Constraint statements for a Table object.
DDL Column Comment	Normally invoked by the DDL Create Table Extras template to generate COMMENT ON statements (or equivalent) for each Table column.
DDL Column Definition	Invoked by numerous templates to build the statement to create a single Table column, as it appears in the CREATE TABLE statement.

DDL Column Extras	Normally invoked by the DDL Create Table Extras template to generate any extended column properties for each Table column.
DDL Constraint Column Name	Invoked by each of the constraint templates to retrieve the correctly formatted column names involved in the current constraint.
DDL Constraint Comment	Normally invoked by the DDL Create Table Extras template to generate COMMENT ON statements (or equivalent) for each Table constraint.
DDL Create Foreign Keys	Invoked by the DDL Create Table Constraints template to generate Foreign Key constraints for a Table object.
DDL Create Function	Invoked by the DDL Script File template to generate the CREATE FUNCTION statement for a Function object.
DDL Create Package	Invoked by the DDL Script File template to generate the CREATE PACKAGE statements for a Package object (Oracle only).
DDL Create	Invoked by the DDL Script File template

Procedure	to generate the CREATE PROCEDURE statement for a Procedure object.
DDL Create Schema	Currently not used.
DDL Create Sequence	Invoked by the DDL Script File template to generate the CREATE SEQUENCE statement for a Sequence object.
DDL Create Table	Invoked by the DDL Script File template to generate the CREATE TABLE statement for a Table object.
DDL Create Table Constraints	Invoked by the DDL Script File template to generate Table constraints and Indexes for a Table object.
DDL Create Table Extras	Invoked by the DDL Script File template to generate extended Table properties for a Table object.
DDL Create View	Invoked by the DDL Script File template to generate the CREATE VIEW statement for a View object.
DDL Data Type	Invoked by the DDL Column Definition template to generate the correctly formatted data type statement for a Table

	column.
DDL Drop Column Extras	Invoked by the DDL Drop Table Extras template to generate any specialized drop statements for column extended properties.
DDL Drop Foreign Keys	Invoked by the DDL Script File template to generate the statements to DROP all Foreign Keys for a Table object.
DDL Drop Function	Invoked by the DDL Script File template to generate the DROP FUNCTION statement for a Function object.
DDL Drop Procedure	Invoked by the DDL Script File template to generate the DROP PROCEDURE statement for a Procedure object.
DDL Drop Sequence	Invoked by the DDL Script File template to generate the DROP SEQUENCE statement for a Sequence object.
DDL Drop Table	Invoked by the DDL Script File template to generate the DROP TABLE statement for a Table object.
DDL Drop Table Extras	Invoked by the DDL Script File template to generate the statements to DROP all

	extended properties for a Table object.
DDL Drop View	Invoked by the DDL Script File template to generate the DROP VIEW statement for a View object.
DDL Foreign Constraint	Invoked by the DDL Table Constraint template to generate the ADD FOREIGN KEY CONSTRAINT statements for a Table object.
DDL Grant	Invoked by the DDL Create Table Extras template to generate the GRANT statement for the current object (Oracle only).
DDL Index	Invoked by the DDL Table Constraint template to generate the CREATE INDEX statements for a Table object.
DDL Left Surround	Used to define the character (or characters) used as the left-hand delimiter on the name of an object (or object component).
DDL Name	Used by most templates to provide a common way of formatting the name of an object (or object feature). This template accepts four parameters:

	• Object Location (values: EA or LIVE) • Object Type (values: OWNER, TABLE, VIEW, PROCEDURE, FUNCTION, SEQUENCE, PACKAGE, COLUMN, CONSTRAINT, CONSTRAINT_COLUMN, REFERENCE_TABLE, REFERENCE_COLUMN) • Include Owner flag; controls if the name should be prefixed by the Owner name (values: INCLUDE_OWNER or {blank}) • Include Surround flag; controls if the name should be delimited by the left and right surround characters (values: INCLUDE_SURROUND or {blank})
DDL Primary Constraint	Invoked by the DDL Table Constraint template to generate the ADD PRIMARY KEY CONSTRAINT statement for a Table object.
DDL Reference Column Name	Normally invoked by the DDL Name templates to retrieve the correctly formatted reference column names involved in a Foreign Key.
DDL	Invoked by the DDL Foreign Constraint

Reference Definition	template to generate the ON DELETE/ON UPDATE statements for a Foreign Key constraint.
DDL Right Surround	Used to define the character (or characters) used as the right-hand delimiter on the name of an object (or object component).
DDL Script File	A top-level template to generate DDL; all other templates are invoked from this one.
DDL Script Header	Invoked by the DDL Script File template to add a header comment at the start of each DDL file.
DDL Script Separator	Used by all templates that must include a statement separator in the generated DDL.
DDL Statement Term	Used to define the character (or characters) used as the statement terminator. For example, semi-colon (';') for most DBMSs.
DDL Statement Term Alt	Used to define the character (or characters) used as the alternative statement terminator. For example, some

	DBMSs must have the statement terminator changed in order to not cause problems with DDL statements generated for SQL-based objects, such as Views and Procedures.
DDL Synonym	Invoked by the DDL Create Table Extras template to generate the CREATE SYNONYMS statement (Oracle only).
DDL Table Constraint	Invoked by the DDL Create Table Constraints template to generate the Table constraints and Indexes for each Table object, taking into account the generation options.
DDL Table Level Comment	Invoked by the DDL Create Table Extras template to generate COMMENT ON statements (or the equivalent) for an object.
DDL Trigger	Invoked by the DDL Table Constraint template to generate the CREATE TRIGGER statements for a Table object.
DDL Unique Constraint	Invoked by the DDL Table Constraint template to generate the ADD UNIQUE CONSTRAINT statements for a Table object.

DDL Use Database	Invoked by the DDL Script File template to include a USE DATABASE statement at the start of each DDL file.

Base Templates for Alter DDL Generation

The DDL Template Framework consists of a number of base templates for Alter DDL generation. Each base template generates DDL statement(s) based on the detected *Action* that must be undertaken to synchronize the data model and live database.

Templates

This table lists and briefly describes the base templates used for Alter DDL generation.

Template	Description
DDL Diff Column	Invoked directly by Enterprise Architect for each Table Column difference that was detected.
DDL Diff Constraint	Invoked directly by Enterprise Architect for each Table Constraint difference that was detected.
DDL Diff Table	Invoked directly by Enterprise Architect for each Table difference that was detected.

DDL Diff View	Invoked directly by Enterprise Architect for each View difference that was detected.
DDL Diff Procedure	Invoked directly by Enterprise Architect for each Stored Procedure difference that was detected.
DDL Diff Function	Invoked directly by Enterprise Architect for each Function difference that was detected.
DDL Diff Sequence	Invoked directly by Enterprise Architect for each Sequence difference that was detected.

DDL Macros

Field substitution macros provide access to data from your model. In particular, they are used to access data fields from:

- Database objects (such as Tables and Views)
- Columns
- Constraints
- Constraint Columns

Field substitution macros are named according to Camel casing. By convention, all DDL macros are prefixed with 'ddl'.

Macros that represent checkboxes or Boolean values return a string value of 'T' if the checkbox/boolean is true. Otherwise an empty string is returned.

Internal Field Macro - ddlAction

The ddlAction macro is an internal macro available in the 'Alter DDL' templates, providing direct access to Enterprise Architect's internal fields; it has no direct mapping to any stored data.

ddlAction represents the action that must be undertaken to synchronize the live database with the current repository. For example, 'Create Table', 'Drop Table' or 'Change Owner'.

Element Field Macros

This list identifies the macros that are available in DDL templates to access element-level fields, where (in Enterprise Architect) the fields are editable, such as 'Table Name' and 'Table Alias'.

ddlFunctionAlias

Function 'Properties' dialog: 'Main' tab: 'Alias' text field.

ddlFunctionName

Function 'Properties' dialog: 'Name' text field.

ddlOwner

{Table element} 'Properties' dialog: {element} 'Table Detail' tab: 'Owner' text field.

ddlPackageAlias

Package 'Properties' dialog: 'Main' tab: 'Alias' text field.

ddlPackageName

Package 'Properties' dialog: 'Name' text field.

ddlProcedureAlias

Procedure 'Properties' dialog: 'Main' tab: 'Alias' text field.

ddlProcedureName

Procedure 'Properties' dialog: 'Name' text field.

ddlSchemaFunctionName

The name of the Function element's definition read in from the live database.

ddlSchemaOwner

The 'Owner' property of the element's definition read in

from the live database.

ddlSchemaProcedureName

The name of the Procedure element's definition read in from the live database.

ddlSchemaSequenceName

The name of the Sequence element's definition read in from the live database.

ddlSchemaTableName

The 'Table Name' property read in from the live database.

ddlSchemaViewName

The name of the View element's definition read in from the live database.

ddlSequenceAlias

Sequence 'Properties' dialog: 'Main' tab: 'Alias' text field.

ddlSequenceName

Sequence 'Properties' dialog: 'Name' text field.

ddlTableAlias

Table 'Properties' dialog: 'Main' tab: 'Alias' text field.

ddlTableDBMS

Table 'Properties' dialog: 'Main' tab: 'Database' drop down list field.

ddlTableLevelComment

Table 'Properties' dialog: 'Notes' text field.

ddlTableName

Table 'Properties' dialog: 'Name' text field.

ddlViewAlias

View 'Properties' dialog: 'Main' tab: 'Alias' text field.

ddlViewName

View 'Properties' dialog: 'Name' text field.

Column Field Macros

This list identifies the macros that are available in DDL templates to access column-related fields, where (in Enterprise Architect) the fields are editable, such as 'Column Name' and 'Column Alias'.

ddlColumnName

'Columns and Constraints' dialog: 'Column' tab: 'Name' cell.

ddlColumnAlias

'Columns and Constraints' dialog: 'Column' tab: 'Alias' cell.

ddlColumnComment

'Columns and Constraints' dialog: 'Column' tab: 'Notes' text field.

ddlSchemaColumnName

The Column **Name** property read in from the live database.

Note: This field is not editable directly in Enterprise Architect.

Constraint Field Macros

This table lists the macros that are available in DDL templates to access constraint-related fields, where (in Enterprise Architect) the fields are editable, such as 'Constraint Name' and 'Constraint Type'.

ddlConstraintAlias

'Columns and Constraints' dialog: 'Constraints' tab: 'Alias' cell.

ddlConstraintColumnAlias

'Columns and Constraints' dialog: 'Constraints' tab: 'Involved Columns: Assigned' list.

ddlConstraintColumnName

'Columns and Constraints' dialog: 'Constraints' tab: 'Involved Columns: Assigned' list.

ddlConstraintComment

'Columns and Constraints' dialog: 'Constraints' tab: 'Notes' text field.

ddlConstraintName

'Columns and Constraints' dialog: 'Constraints' tab: 'Name' cell.

ddlPKColumnCount

Only relevant if the current constraint has a type of Primary Key, this macro will return a count of assigned columns to the Primary Key.

'Columns and Constraints' dialog: 'Constraints' tab: 'Involved Columns: Assigned' list.

ddlReferenceColumnAlias

Only relevant if the current constraint has a type of Foreign Key, this macro will return the column alias from the reference table.

'Columns and Constraints' dialog: 'Constraints' tab: 'Alias' cell.

ddlReferenceColumnName

Only relevant if the current constraint has a type of Foreign Key, this macro will return the column name from the reference table.

Foreign Key 'Constraint' dialog: 'Involved Columns' list: 'Parent' column.

ddlReferenceTableAlias

Only relevant if the current constraint has a type of Foreign Key, this macro will return the reference table's alias.

Table 'Properties' dialog: 'Main' tab: 'Alias' text field.

ddlReferenceTableName

Only relevant if the current constraint has a type of Foreign Key, this macro will return the reference table's name.

Foreign Key 'Constraint' dialog: 'Involved Columns' list: 'Parent' column header.

ddlReferenceTableOwner

Only relevant if the current constraint has a type of Foreign Key, this macro will return the reference table's owner.

Foreign Key 'Constraint' dialog: 'Involved Columns' list: 'Parent' column header.

ddlSchemaConstraintColumnName

The column names involved in the current constraint read in from the live database.

Note: this field is not editable directly in Enterprise Architect.

ddlSchemaConstraintName

The Constraint **Name** property read in from the live database.

Note: this field is not editable directly in Enterprise Architect.

ddlSchemaConstraintType

The Constraint **Type** property read in from the live database.

Note: this field is not editable directly in Enterprise Architect.

DDL Function Macros

The DDL Function macros provide a convenient way of manipulating, retrieving or formatting element data relevant to DDL generation. These macros, along with the code function macros, are available to the DDL templates. Each Function macro returns a result string and is used in the same manner as a Code Template Function macro.

The available function macros are described here. All parameters have a type of String and are denoted by square brackets; that is: FUNCTION_NAME([param]).

DDL_DATATYPE_SIZE ([productName], [datatype])

Returns the fully formatted datatype of the current column in DDL syntax.

Parameters

- productName - the current Table's assigned DBMS, such as SQL Server 2012, Oracle or PostgreSQL
- datatype - the current column's datatype name, such as VARCHAR or INT

Remarks

Within an Enterprise Architect Table column, datatypes are defined with a Length Type (0, 1 or 2) property that influences the DDL syntax; this function macro takes the

Length Type (and other factors) into consideration when building the return value.

DDL_GET_DEFINITION_PARAS ([definition])

Returns a string representation of the parameters from the supplied function/procedure definition.

Parameters

- definition - the complete SQL definition of the procedure/function

Remarks

Some DBMSs (such as PostgreSQL) support multiple definitions of the same procedure/function name. The definitions differ only in their parameter list, therefore to manipulate such objects the DDL must specify the name and parameters. This function macro gives the DDL templates the ability to extract the parameters so that they can then be used to identify individual objects.

DDL_INCLUDE_SQLQUERY([objectName])

Returns the SQL statement defined in the SQLQuery object.

Parameters

- objectName - the name of the SQL Query object defined in the current data model

Remarks

None.

DDL_INDEX_SORT([product],[columns])

Returns the sort order of a given index.

Parameters

- product - the DBMS (currently, Firebird)
- columns - a CSV of column names involved in the index

Remarks

This macro currently only applies to Firebird indexes.

DDL_RESOLVE_NAME ([productName], [name], [leftSurround], [rightSurround])

Returns the supplied name delimited (with the supplied left and right characters) if the name is a reserved word for the current DBMS.

Parameters

- productName - the current Table's assigned DBMS, such as SQL Server 2012, Oracle or PostgreSQL
- name - the object/column name
- leftSurround - the left character of the pair used to surround the name; for example, single quote {'}

- rightSurround - the right character of the pair used to surround the name; for example, single quote {'}

Remarks

The DDL syntax of some DBMSs requires names that are reserved words to be delimited in a different manner; this function macro can be used to safely format all names for DB2 and Firebird.

DDL_TABLE_TAGVALUE ([tagName])

Returns the value for the supplied tag name in the repository's version of the current Table.

Parameters

- tagName - the tag item's name that is to be retrieved

Remarks

None.

EXECUTE_CURRENT ([objectName], [actionName], [priority])

Adds the return string from the current template to the Execution Engine's execution queue.

Parameters

- objectName - the value that will be shown in the 'Object'

column of the execution queue, which indicates the name of the object being updated

- actionName - the value that will be shown in the 'Action' column of the execution queue, which indicates the action that resulted in the generation of this statement
- priority - a numeric value that represents the priority of the statement; the higher the number, the lower in the queue the statement is placed

Remarks

This function macro can be called at any point throughout the template, but will not execute until the end. Once the template is complete, the DDL it has generated is sent to the execution queue.

This function macro has no effect if the user has elected to generate DDL to a file.

EXECUTE_STRING ([objectName], [actionName], [priority], [ddlStatement])

Adds the supplied DDL statement to the Execution Engine's execution queue.

Parameters

- objectName - the value that will be shown in the 'Object' column of the execution queue, which indicates the name of the object being updated
- actionName - the value that will be shown in the 'Action'

column of the execution queue, which indicates the action that resulted in the generation of this statement

- priority - a numeric value that represents the priority of the statement; the higher the number, the lower in the queue the statement is placed
- ddlStatement - a single DDL statement that performs the required action

Remarks

This function macro has no effect if the user has elected to generate DDL to a file.

EXIST_STRING ([ddlStatement])

Searches the Execution Engine's execution queue for the supplied DDL Statement and returns 'T' if the statement is found.

Parameters

- ddlStatement - a single DDL statement

Remarks

None.

GET_FIRST_SQL_KEYWORD([statement])

Returns the first keyword of the provided SQL statement.

Parameters

- statement - the SQL statement

Remarks

None.

ODBC_TABLE_TAGVALUE ([tagName])

Returns the value for the supplied tag name in the live database's version of the current table.

Parameters

- tagName - the tag item's name that is to be retrieved

Remarks

None.

PROCESS_DDL_SCRIPT ([type], [parameter2], [parameter3], [parameter4])

A generic function macro that returns a formatted string for a specific purpose.

Parameters

- type - specifies the special action to be undertaken
- parameter2 - generic parameter 2, will have a different purpose for each type

- parameter3 - generic parameter 3, will have a different purpose for each type
- parameter4 - generic parameter 4, will have a different purpose for each type

Remarks

For Oracle Synonyms use these parameters:

- type = "SYNONYMS"
- parameter2 = the table name; for example, TBL_EMPLOYEES
- parameter3 = a delimited string of values, separated by semi-colons, specifying the synonym owner and name with full colon between; for example, OE:EMPLOYEES;PUBLIC:PUB_EMPLOYEES;
- parameter4 = the statement terminator

Return Result

Of the format:

CREATE SYNONYM OE.EMPLOYEES FOR TBL_EMPLOYEES;

CREATE PUBLIC SYNONYM PUB_EMPLOYEES FOR TBL_EMPLOYEES;

REMOVE_LAST_SEPARATOR ([ddlStatement], [separator])

Returns the supplied DDL statement with the last separator

removed (if it exists).

Parameters

- ddlStatement - a partial DDL statement
- separator - the separator character that should be removed

Remarks

When building a string that represents a DDL statement, it is common practice to append the separator character after each item; however, the separator is not required after the last item, so this function macro is provided to remove the trailing separator.

REMOVE_STRING ([ddlStatement])

Removes the supplied DDL statement from the Execution Engine's execution queue.

Parameters

- ddlStatement - a single DDL statement

Remarks

None.

SUPPRESS_EXECUTE_CURRENT ([boolean])

A function macro to enable/disable subsequent calls to EXECUTE_CURRENT.

Parameters

- boolean - True or False

Remarks

The default state for this flag is False; that is, calls to EXECUTE_CURRENT are not ignored.

DDL Property Macros

The DDL Property macros provide a convenient way of retrieving element property values (that is, Tagged Values). In the scope of data modeling there are two groups of properties:

- Internal properties (those that Enterprise Architect recognizes and uses in its compares) and
- User-defined properties

These property macros provide access to properties defined against the various elements. All property macros have the same syntax, return a string and require the name of the property to be specified.

Syntax: propertyMacroName:"propertyName"

INTERNAL PROPERTIES

tableBoolProperty:"propertyName"

Returns a Boolean representation ("T" or "") of the value for the internal property in the repository's version of the current Table.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

tableProperty:"propertyName"

Returns the value for the internal property in the repository's version of the current Table.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

columnProperty:"propertyName"

Returns the value for the internal property in the repository's version of the current Column.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

columnBoolProperty:"propertyName"

Returns a Boolean representation ("T" or "") of the value for the internal property in the repository's version of the current Column.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

constraintProperty:"propertyName"

Returns the value for the internal property in the repository's version of the current Constraint.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

constraintBoolProperty:"propertyName"

Returns a Boolean representation ("T" or "") of the value for the internal property in the repository's version of the current Constraint.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

constraintColumnProperty:"propertyName"

Returns the value for the internal property in the repository's version of the current Constraint Column.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

constraintColumnBoolProperty:"propertyName"

Returns a Boolean representation ("T" or "") of the value for the internal property in the repository's version of the current Constraint Column.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

viewProperty:"propertyName"

Returns the value for the internal property in the repository's version of the current View.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

procedureProperty:"propertyName"

Returns the value for the internal property in the repository's version of the current Procedure.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

functionProperty:"propertyName"

Returns the value for the internal property in the repository's version of the current Function.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

sequenceProperty:"propertyName"

Returns the value for the internal property in the repository's version of the current Sequence.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

packageProperty:"propertyName"

Returns the value for the internal property in the repository's version of the current database Package.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

odbcTableProperty:"propertyName"

Returns the value for the internal property in the ODBC's version of the current Table.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

odbcConstraintProperty:"propertyName"

Returns the value for the internal property in the ODBC's version of the current Constraint.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

USER DEFINED PROPERTIES

tableUserProperty:"propertyName"

Returns the value for the user-defined property in the repository's version of the current Table.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

columnUserProperty:"propertyName"

Returns the value for the user-defined property in the repository's version of the current Column.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

constraintUserProperty:"propertyName"

Returns the value for the user-defined property in the repository's version of the current Constraint.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

constraintColumnUserProperty:"propertyName"

Returns the value for the user-defined property in the repository's version of the current Constraint Column.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

viewUserProperty:"propertyName"

Returns the value for the user-defined property in the repository's version of the current View.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

procedureUserProperty:"propertyName"

Returns the value for the user-defined property in the repository's version of the current Procedure.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

functionUserProperty:"propertyName"

Returns the value for the user-defined property in the repository's version of the current Function.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

sequenceUserProperty:"propertyName"

Returns the value for the user-defined property in the repository's version of the current Sequence.

Parameters

- propertyName - the property name that is to be retrieved

Remarks

None.

DDL Options in Templates

The DDL Generation Options macros provide a convenient way for the DDL templates to access the generation options.

This list identifies and briefly describes each of the available option macros. Each option has a value of either 'T' for true or an empty string for false.

ddlGenerateToExecuteEngine

Directs the generated DDL to the Execution Engine.

ddlOptionColumnComments

Include column comments in the generated DDL.

ddlOptionGenerateCheck

Include Check constraints in the generated DDL.

ddlOptionGenerateDrop

Include DROP statements in the generated DDL.

ddlOptionGenerateForeign

Include Foreign Keys in the generated DDL.

ddlOptionGenerateFunction

Include Functions in the generated DDL.

ddlOptionGenerateIndex

Include Indexes in the generated DDL.

ddlOptionGenerateLengthSemantic

(Oracle only) Include length semantics syntax on text columns in the generated DDL.

ddlOptionGenerateNullable

Include the keyword NULL against each column if it hasn't been flagged as a NOT NULL column in the generated DDL.

ddlOptionGeneratePackage

(Oracle only) Include Packages in the generated DDL.

ddlOptionGeneratePrimary

Include Primary Key constraints in the generated DDL.

ddlOptionGenerateProcedure

Include Procedures in the generated DDL.

ddlOptionGenerateSeparateConstraint

Generate Table constraints separately to the CREATE TABLE statement; that is, using an ALTER TABLE statement.

Note: Some DBMSs do not support separate constraints in

all conditions.

ddlOptionGenerateSequence

Include Sequences in the generated DDL.

ddlOptionGenerateTable

Include Tables in the generated DDL.

ddlOptionGenerateTableProperty

Include extended properties on Tables in the generated DDL.

ddlOptionGenerateTrigger

Include Table Triggers in the generated DDL.

ddlOptionGenerateUnique

Include Unique Constraints in the generated DDL.

ddlOptionGenerateView

Include Views in the generated DDL.

ddlOptionHeaderComments

Include header comments in the generated DDL.

ddlOptionTableComments

Include Table comments in the generated DDL.

ddlOptionUseAlias

Use Aliases instead of Names for all objects (object components) as specified on the Generate DDL screen.

ddlOptionUseDatabaseName

Include the USE DATABASE statement at the beginning of each generated file.

ddlUseAlias

Use Aliases instead of Names for all objects (object components) as specified on the Database Builder 'Database Compare' tab.

DDL Limitations

A fundamental feature of a Database Management System (DBMS) is to allow the definition of database objects via a structured language; this language is called DDL (for data definition language, or data description language). The DDL syntax of each DBMS is unique. While there are common DDL statements and keywords across all DBMSs, there are differences that require each DBMS to have its own set of DDL templates within Enterprise Architect.

This page summarizes the main limitations for each of the supported Database Management Systems.

MS Access

- Comments cannot be applied to (or changed in) Tables, Table Columns, Table Constraints or Views, therefore Enterprise Architect ignores these differences
- The CREATE TABLE statement does not support the definition of column defaults, therefore Enterprise Architect excludes the Default definition from all generated DDL; however, it does highlight a Default difference in the comparison logic
- Generally object names in DDL can be enclosed in square brackets ([ ]) so that they can include spaces and other non standard characters, however the CREATE VIEW DDL statement does not support the square bracket

notation; the 'Create View' DDL template replaces all spaces with underscore ('_') characters

MySQL

- Comments can only be applied to Indexes and Unique Constraints, when the MySQL version is greater than 5.5.3
- Comments can only be applied to Indexes and Unique Constraints when they are created, therefore changing an Index or Unique Constraint's comment causes the constraint to be dropped and recreated
- Check Constraints are not supported; whilst the MySQL DDL engine can parse such statements, it simply ignores them
- Comments cannot be applied to (or changed in) Views, Procedures or Functions, therefore Enterprise Architect ignores these differences

Oracle

- Comments cannot be applied to (or changed in) Procedures, Sequences or Functions, therefore Enterprise Architect ignores these differences

PostgreSQL

- Currently Enterprise Architect does not support function parameters, therefore any statements (COMMENT ON or DROP) that refer to a function by name will fail because they must use a combination of function name and parameters

SQL Lite

- Constraints cannot be added to an existing Table; the Table must be dropped and created (including the new Constraint in the Create statement)
- Comments are not supported on any object type, therefore Enterprise Architect ignores all remark differences

Supported Database Management Systems

Enterprise Architect has built in support for a comprehensive range of database management systems, but it also provides the flexibility to extend the product to support other DBMSs. The DDL template editor can be used to define how to generate DDL for an unsupported DBMS, the transformation templates can be used to define a new transformation to a physical model for an unsupported DBMS, and new datatypes can be defined for an existing or new DBMS.

Enterprise Architect provides the modeling constructs and the ability to forward and reverse engineer a database schema for these Database Management Systems:

- DB2 (*)
- Firebird
- MS Access 97, 2000, 2003, 2007, 2013
- MS SQL Server from 2005, all editions including Express and Azure SQL Database
- MariaDB
- MySQL v4, v5
- Oracle from 9i (all editions)
- PostgreSQL (including version 12)
- SQLite
- Informix (#)

- Ingres (#)
- InterBase (#)
- Sybase Adaptive Server Anywhere (Sybase ASA) (#)
- Sybase Adaptive Server Enterprise (Sybase ASE) (#)

(*) - Only compatible for DB2 when hosted in Windows and Linux environments.

(#) - No further development will be undertaken on these DBMSs, as these products are not commonly used by the Enterprise Architect user base. This will allow Sparx Systems to concentrate its efforts on the other areas of Database modeling which are used extensively.

Notes

- To perform data modeling for a particular DBMS, you must have the appropriate data types for that DBMS in your repository; you can download the most up-to-date data definitions from the 'Resources' page of the Sparx Systems web site

More Information

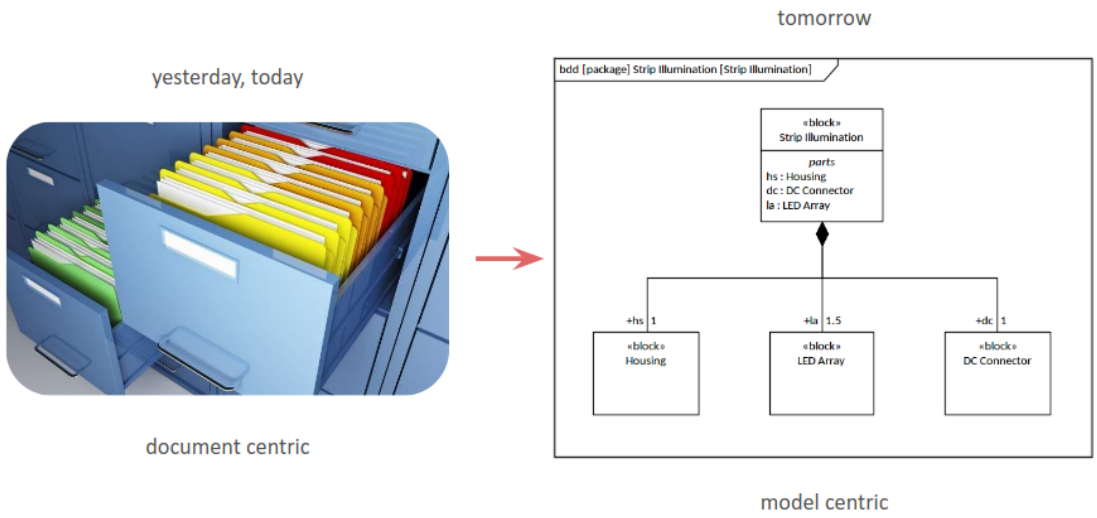
Edition Information

The Database Builder is available in the Corporate, Unified and Ultimate Editions of Enterprise Architect.

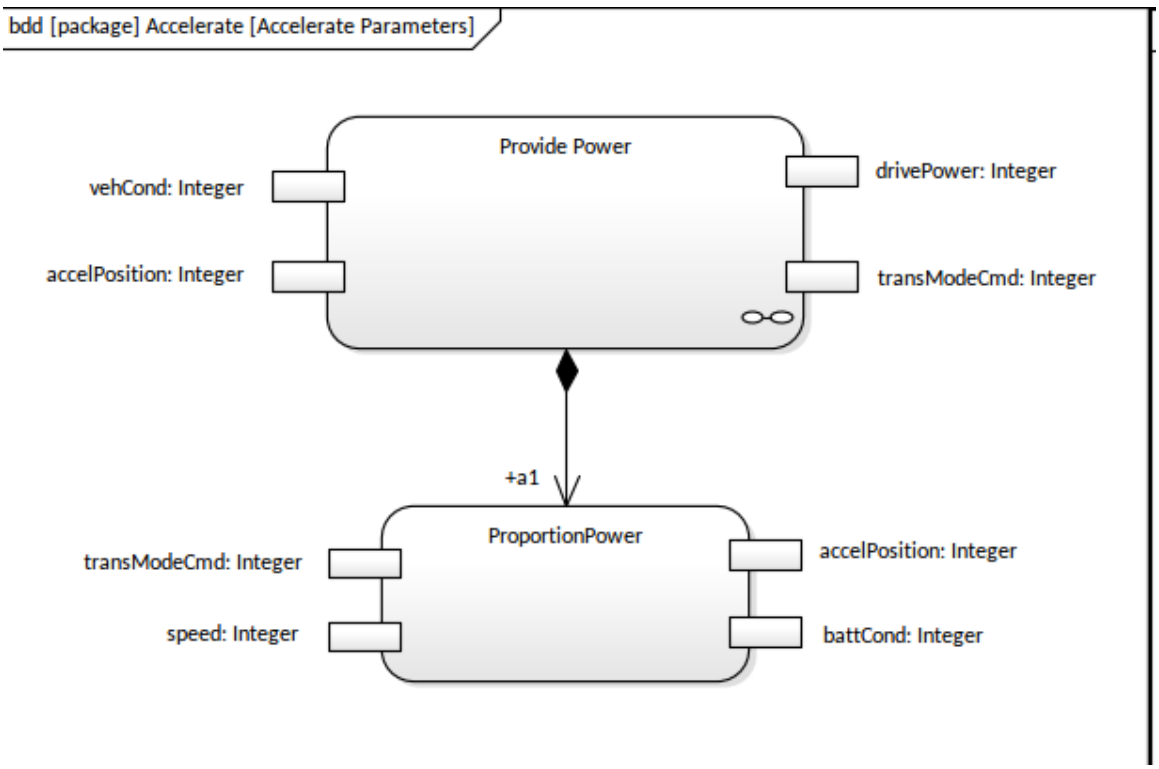
Systems Engineering

Systems Engineering is, very broadly, the work of researching, designing and managing complex physical or electronic systems over their lifecycles. It focuses on the whole system and typically involves a number of sub-disciplines such as requirements, reliability, logistics, design, testing and maintenance; it considers not only the system itself but also processes, optimization and risk management, and requires sophisticated project management techniques.

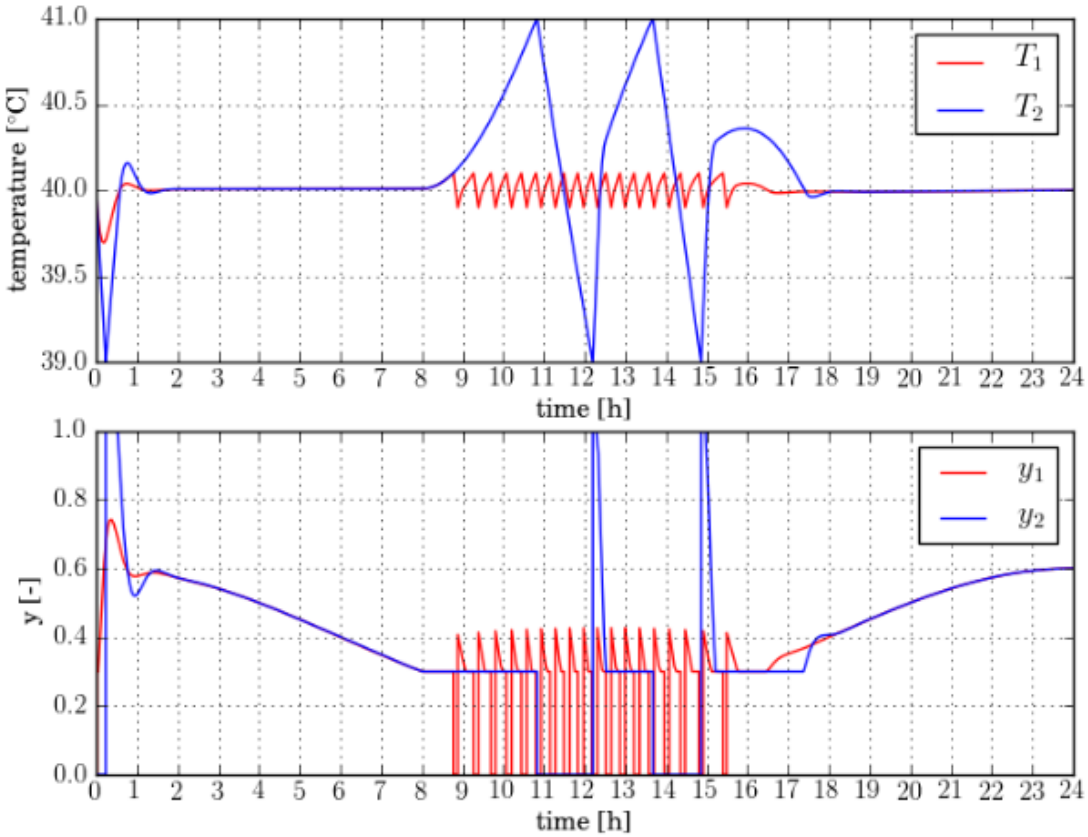
In earlier decades a large but localized team might consider a very specific set of objects within a very specific and controlled environment, to be delivered to a small user base and maintained by perhaps an, again, localized team of experts who each might have responsibility for only a part of the system. Even for such a controlled and structured scenario, a huge volume of documentation was required to define the system requirements, the components, the engineering process, the standards applied and complied with, and the tests to be run on the system. Keeping this documentation cross-referenced, up to date and integrated was a major task.



Advancing into computing and basing Systems Engineering work on graphical models (Model Based Systems Engineering) provided huge benefits, allowing engineers to store and retrieve data from repositories, associate data with documentation also held in the repositories, and develop both master structures and variants from templates, all of which reduced the need to recreate and repeat work. The model initially represented the organization of the developing system, but grew to reflect the development process and the factors that supported and directed that process. As computing capabilities grew, and more specialized and sophisticated applications were made available, it became possible to represent the components of a system with increasingly varied and detailed model elements, and with increasingly varied and detailed relationships between them.



Engineers could 'load' the model components and relationships with an array of properties, characteristics and parameters, which could be varied to reflect different scenarios. The standards that the system must apply or meet could be automatically enforced on the components as constraints, conditions and rules. More and more of the development process - such as testing - could be represented by element or model features, and more and more aspects of the process could be performed on the model by the application - such as automatic generation of code to make the system operational, and simulation of the system in action under various conditions.



Currently, the Systems Engineer is likely to be a member of an interdisciplinary team that has to consider a wide range of factors in architecting, designing and modeling a system - a much broader, diverse and inexpert user base, a much broader maintenance base, how the system interacts with many other systems, how the system operates in many different and sometimes extreme environments, the impact the system has on the global environment - both within its operating framework and within its pre-use production and final disposal - the socio-economic environment controlling its acceptability and popularity, and how the system compares with its increasing range of competitors. To see how the work of the Systems Engineer has become vastly more complex one has only to think of a single development, such as the quantum leap from the relatively recent fixed-site landline telephone handset for making

voice calls, to the modern mobile smartphone used as a camera, computer, cinema, music center, navigator, and audio, visual and text communicator.

Today, large projects and industries are being developed around systems and products for which the use cases are increasingly complex. Controlling this complexity grows further and further beyond the capacity of the engineer, increasing the level of risk to the product, the end user and the manufacturer. Examples of systems with dramatically increased risk include the manufacture of passenger air bags to be fitted to many different brands and types of car manufactured in different parts of the globe; or the requirements for the development of space probes intended to travel to the planets of the solar system and beyond.

It is the advances in Systems Engineering tools and methodologies that have increased this complexity, whilst simultaneously providing the capability to manage and mitigate the associated risk, and reducing the difficulty and effort involved in managing and maintaining highly complex models.

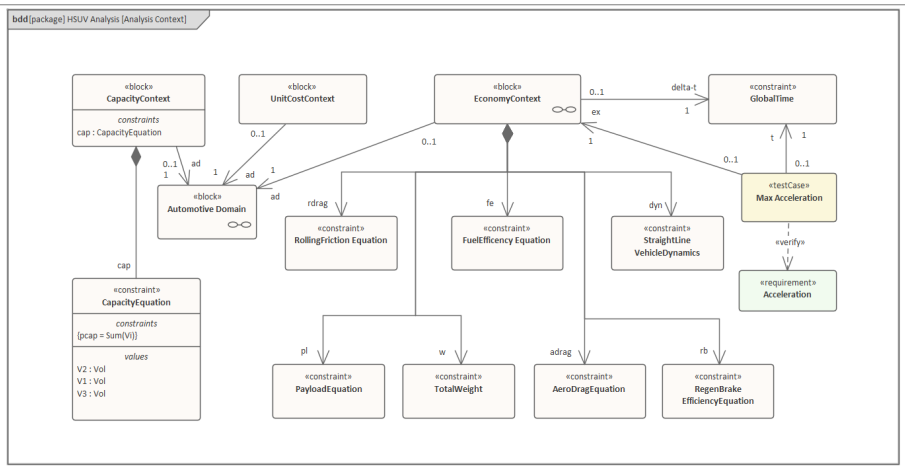
For additional information, see the *Representing Systems*

with Models section of the 'SEBoK - Guide to the Systems Engineering Body of Knowledge' website.

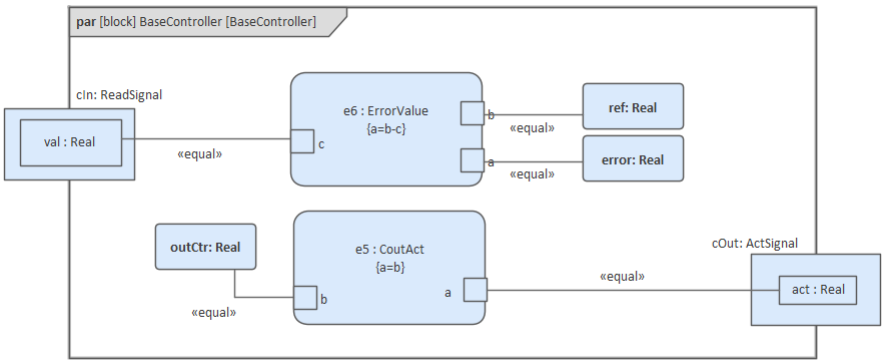
Model-Based Systems Engineering in Enterprise Architect

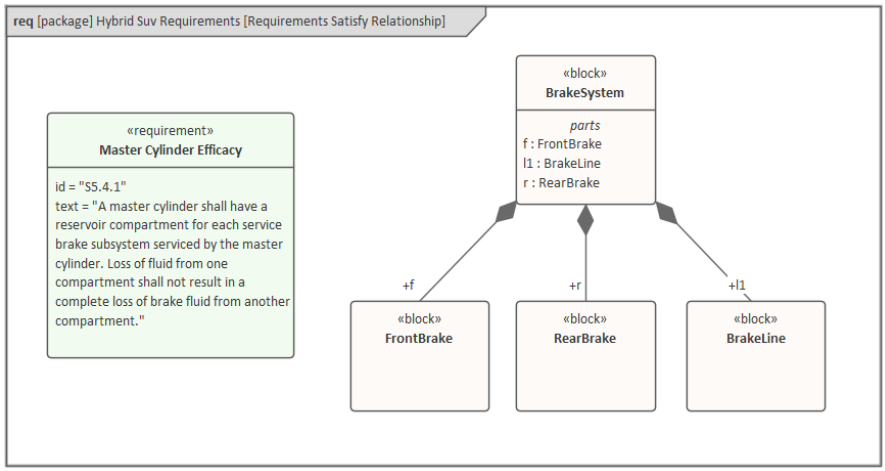
Enterprise Architect provides a Model Based Systems Engineering platform that integrates many high-end features for Systems Engineers and model-based development, with these built-in features.

Feature	Description
SysML	Enterprise Architect is integrated with the Systems Modeling Language (SysML) versions 1.1, 1.2, 1.3, 1.4 and 1.5. For details, see the *Systems Modeling Language (SysML)* Help topic. Enterprise Architect provides a number of engineering model templates from which models of engineering structures and concepts can be developed. This is an image of a SysML 1.5 Block Definition diagram. It is part of the HSUV Model that can be found in the 'Systems Engineering' section of Enterprise Architect's Example Model.

	
Conformance to Standards	As well as applying the standards defined by the OMG for UML and SysML, the Enterprise Architect Model Based Systems Engineering platform also complies with these international standards: • International Council of Systems Engineering (INCOSE) 2012 • Ontology Definition Metamodel (ODM) (OMG document *ptc/2013-12-03*, pub. February 2014) • Systems Modeling Language (SysML) (OMG document *formal/2017-05-01*) • Unified Profile for United States Department of Defense Architecture Framework (DoDAF) and United Kingdom Ministry of Defense Architecture Framework (MODAF) (UPDM) (OMG document *formal/2013-01-01*)

Executable Code Generation	You can quickly generate executable software code from your model elements, using Executable StateMachines. The code generated for an Executable StateMachine is based on its language property. This might be Java, C, C++, C# or JavaScript. Whichever language it is, Enterprise Architect generates the appropriate code, which is immediately ready to build and run. There are no manual interventions necessary before you run it. For more information, see the *Code Generation for Executable StateMachines* Help topic.
Model to Code Transformations for HDLs	You can not only generate executable software code, but you can generate Hardware Description Languages and Ada from your model elements, for the chips and circuits in system hardware components. For more information, see the *StateMachine Modeling for HDLs* Help topic.
Parametric Model Simulation	Enterprise Architect provides facilities to create Parametric diagrams using the Parametric Diagram Modeling Assistant, and to perform Parametric Model Simulation through OpenModelica. Being

	able to simulate a system through the model is a huge advantage where live-testing would be dangerous (defense systems) or prohibitively expensive (space probes). This image shows an Internal Block diagram used in a Parametric Model Simulation. The diagram is part of the 'Two Tanks' example that can be found in the 'Systems Engineering > Modelica Examples' section of Enterprise Architect's Example Model.  For further information, see the *Parametric Diagrams*, *Parametric Diagram Modeling Assistant* and *Parametric Simulation Using OpenModelica* Help topics.
System-of-Systems Modeling	In addition to developing system models, you can also design 'system-of-system' models, or system architectures, using the Unified Profile for DoDAF and MODAF (UPDM) or the Unified Architecture

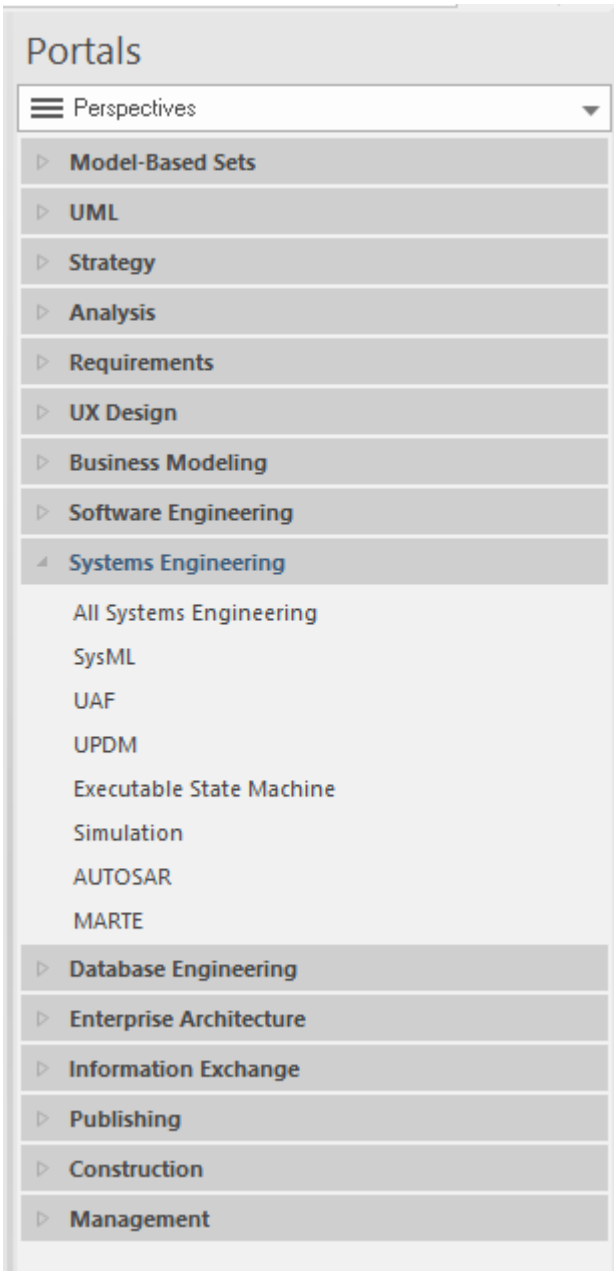
	Framework (UAF); these are both accessible through the Systems Engineering Perspective with SysML.
Requirements Management	Enterprise Architect has an extensive suite of Requirements Management tools that can be applied to System Engineering, dove-tailed to the SysML Requirements modeling facility. See the *SysML Requirements Model* and *Requirement Models* Help topics. This image shows an example of a SysML Requirements diagram. 
Project Management	Enterprise Architect has extensive Project Management and team support facilities to help you organize, support and manage both the Systems Engineering model content and the staff working on the project. Amongst other things, you can apply user security, organize and monitor resources, schedule tasks, apply Version

	Control and enable a range of discussions from simple messaging through informal topic discussion threads to formal reviews. For more information, see the *Project Management* and *The Modeling Team* Help sections.

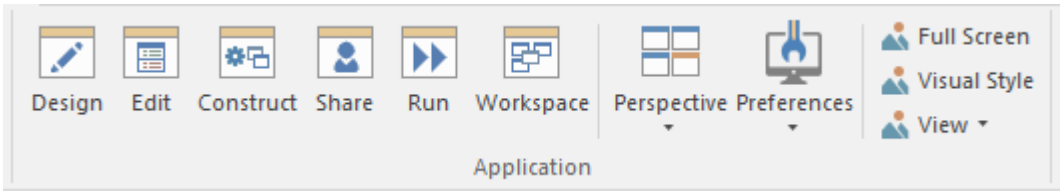
Getting Started

Getting started with a new tool is can be quite daunting even for experienced engineers, but Enterprise Architect makes this easy by providing a number of facilities to assist the newcomer to the tool. Enterprise Architect is a large and multi-featured application and the breadth of its coverage might appear overwhelming to a person new to the program, but fortunately a solution to this has been built into the design. One of the main tool features is perspectives.

You can use Perspectives to limit the functionality to a specific domain or language, such as System Engineering, making it easy for a System Engineer or Manager to get started. A user still can utilize other functionality that might be useful, such as Strategic Modeling, Mind Mapping, Code Engineering, and more, simply by changing Perspectives, all without having to open a different tool. It is worth noting that Perspectives exist for a wide range of modeling disciplines that Enterprise Architect supports.

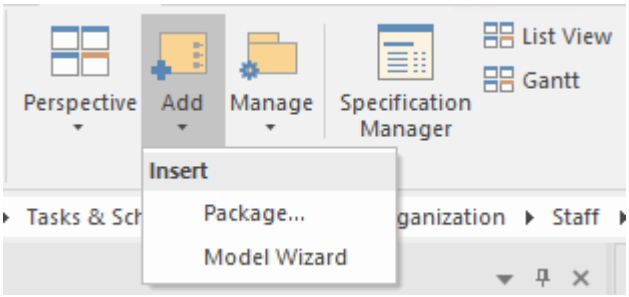


A user also has tremendous flexibility to tailor their environment and the user interface by setting preferences and selecting workspaces and visual styles.

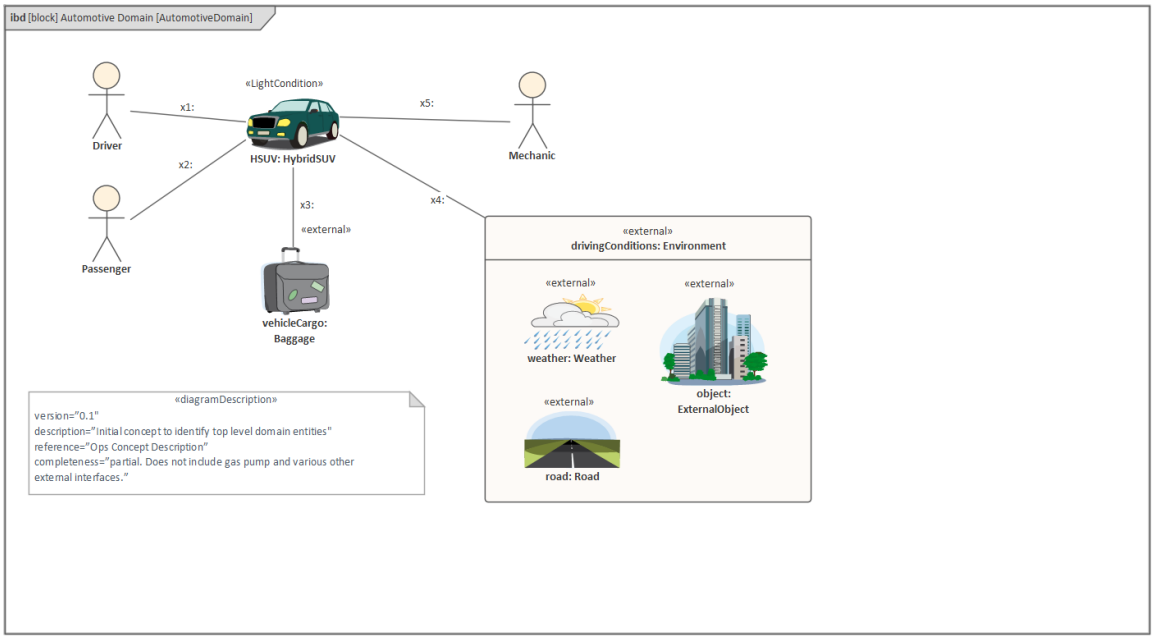


Setting up a new project is straightforward with the use of the Model Wizard Patterns (with accompanying

documentation) that can be utilized to automatically create an MBSE project structure to get you started. You can use the Model Wizard (Start Page 'Create from Pattern' tab) to create any number of SysML diagrams as you develop the model and the problem and solution spaces are fleshed-out.

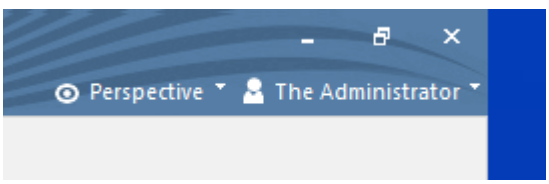


These and other facilities make it easy for a newcomer to get started, allowing them to become productive members of a team and start contributing to models quickly and without any delay. A novice engineer will be surprised how productive they can be when compared to working in text-based or other more rudimentary modeling tools. There will be challenges along the way as you push yourself and the tool to new limits but a detailed Help system, a large community of users, comprehensive forums, a community site and first-class support services will make the journey easy and informative. You will be able to create expressive diagrams like the following one from the automotive domain, and communicate with engineering colleagues, managers, consultants and customers.



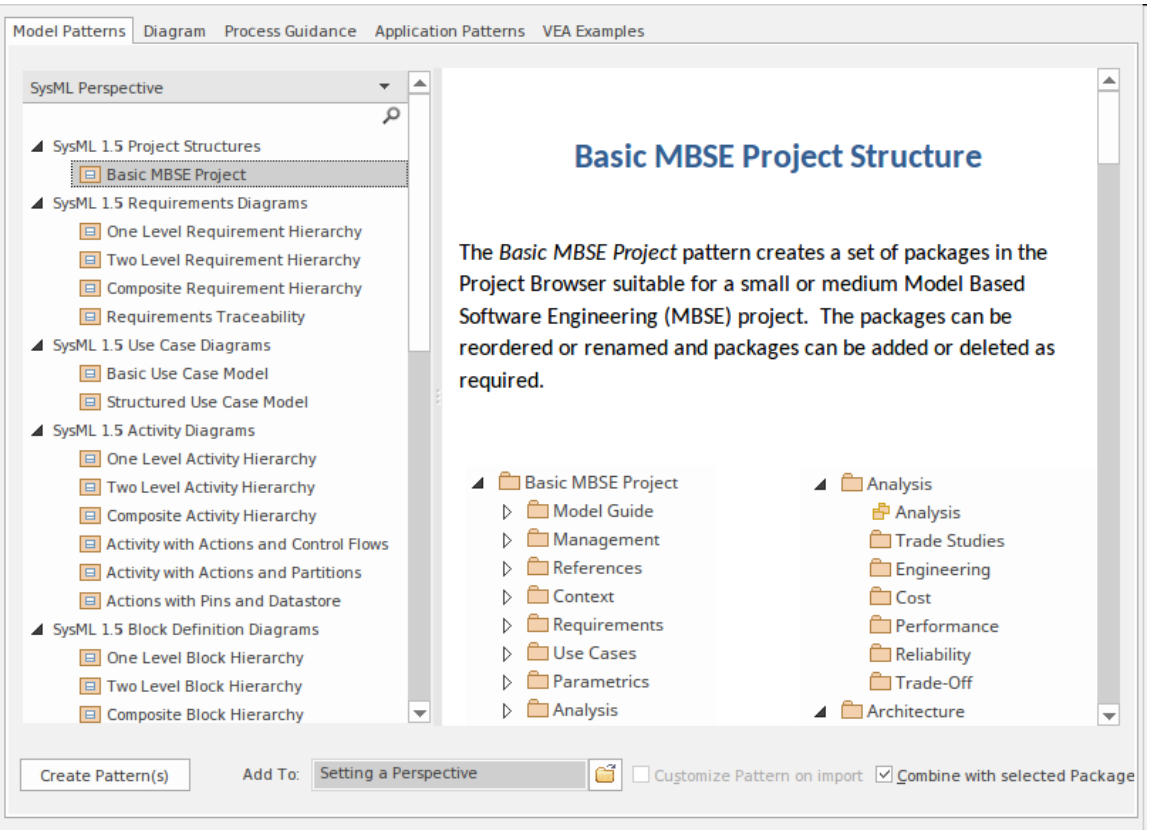
Setting a Perspective

Enterprise Architect is a tool packed with features for a wide range of disciplines, methods, languages and frameworks. Perspectives provide a way for a user to select a facet of the tool that allows them to focus on a particular subset of the tools features and facilities. The Systems Engineering group of Perspectives provides a natural starting point for Systems Engineers, but at any point if you decide to use other facilities in the tool you can simply change Perspectives and the tool will change to provide a focus on the selected area.



Selecting one of the Systems Engineering Perspectives will

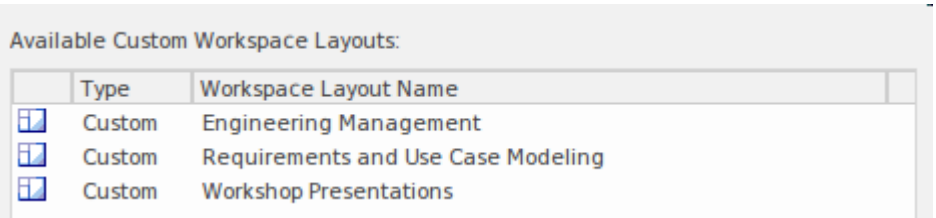
change the tools to focus on the selected aspect of Systems Engineering. For example, choosing the SysML Perspective will display a series of model patterns giving a user a jump start by being able to load a pattern for a standard model fragment or diagram. The 'New Diagram' dialog will also just display SysML diagram types.



Selecting a Workspace

Enterprise Architect has a helpful way of quickly changing the User Interface layout to facilitate particular engineering or management tasks or ways of working. This is achieved by simply selecting a workspace that will change the visible windows and tools to provide the most efficient working

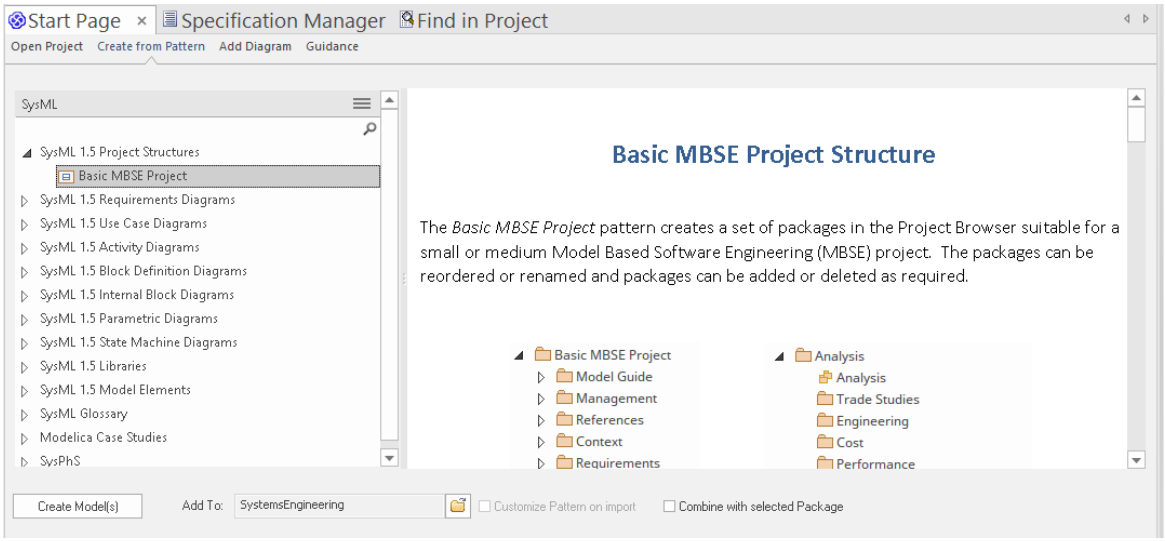
method to suit the task. For example, there is a workspace defined for Systems Engineering Simulations, one for Use Case Modeling, and another for Testing. You can also specify any number of your own workspace layouts that you find helpful by opening windows and tools and positioning them in an arrangement that facilitates working on a particular task or set of tasks and saving them. In this example, a modeler has defined three custom workspace layouts.



Setting Up a Model Structure

Enterprise Architect has been designed as a productivity tool from the ground up. One of the first tasks in a modeling project is setting up a model structure which can sometimes be daunting for the beginner and tedious for the experienced user. Enterprise Architect makes this task simple by using the Model Wizard (Start Page 'Create from Pattern' tab).

You can create the structure for a new initiative (project) using the Model Wizard, which will produce an entire project structure that can be tailored on import, providing all the Packages ready to start the project.



The repository structure is a subject that is explored in a later topic because it is critical to the success of a model-based engineering approach to Systems Engineering. We will learn later that Packages are essential units in the organization and maintenance of a model repository. There is an entire topic dedicated to using Packages to structure the repository. For more information, see the Model Wizard Help topic.

Example Models

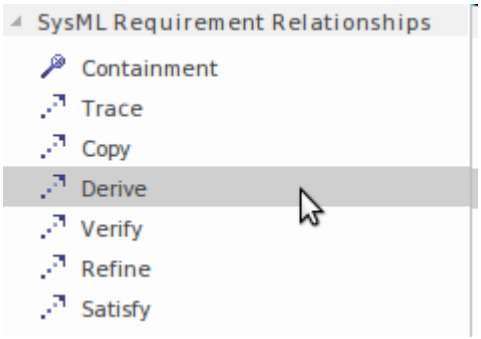
Transitioning from a document centric approach to a model based systems engineering method can present some hurdles for the newcomer. Fortunately Enterprise Architect has rich and supportive help available and a series of in-tool facilities such as patterns that will get you started with the tool and project models. In this topic we present some straight-forward examples of requirement, structural and behavioral models and the diagrams that you or your colleagues would create in a typical project.

Requirements Models

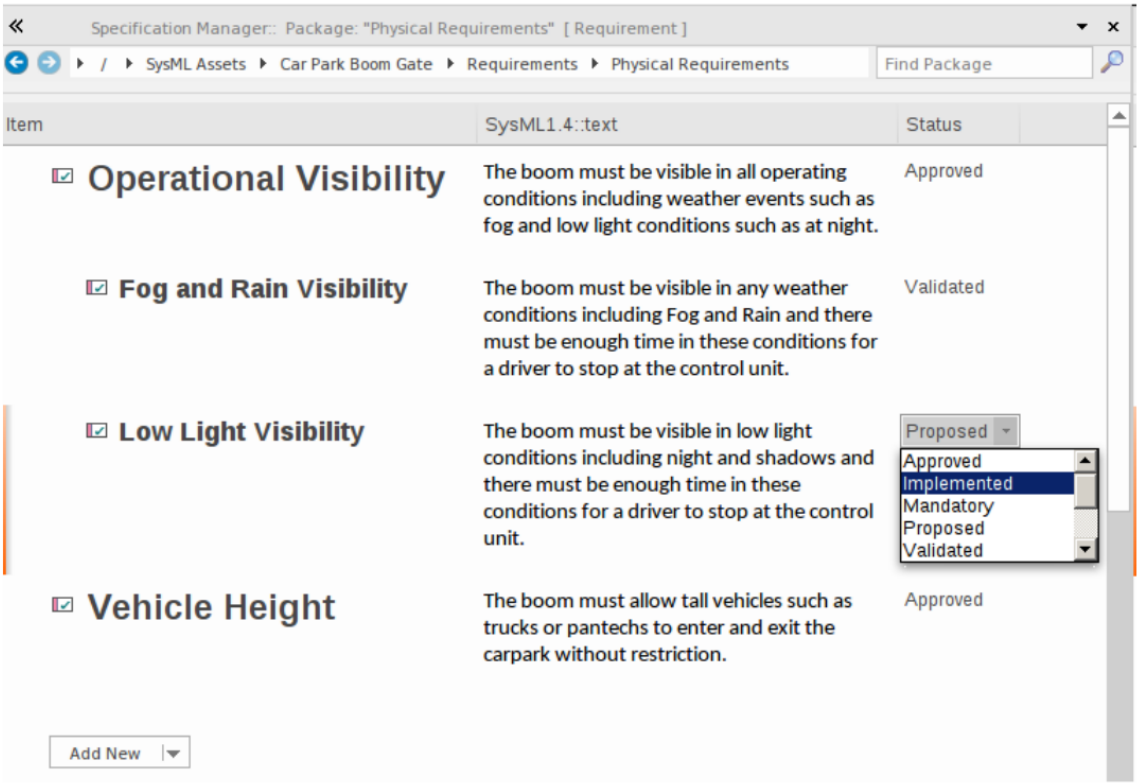
Requirements models are fundamental to any Systems Engineering project whether be a greenfield project or a change to an existing system. Requirements are typically derived from a number of sources including meetings and workshops with stakeholders, documents or formal and formal requests from sponsors and other project stakeholders. There are a three main types of requirements including high level strategic or business requirements, user requirements and system requirements often referred to as quality attributes of the system.

Requirements models will typically evolve over the lifetime of the project and adaptive and iterative methods encourage requirement changes as stakeholders view the partially

completed product at critical project milestones. These changes can be represented in the tool in a variety of ways including using the change management functionality available from the construct ribbon. Alternatively these changes can be represented as derivation relationships and visualized in a requirements diagram.



Enterprise Architect provides a rich suite of tools for requirement elicitation, development and management and enforces good requirement engineering practices. One of the key tools for working with requirements is the Specification Manager which allows requirements engineers more familiar with tools like word processors or spreadsheets to work in these familiar paradigms inside Enterprise Architect.



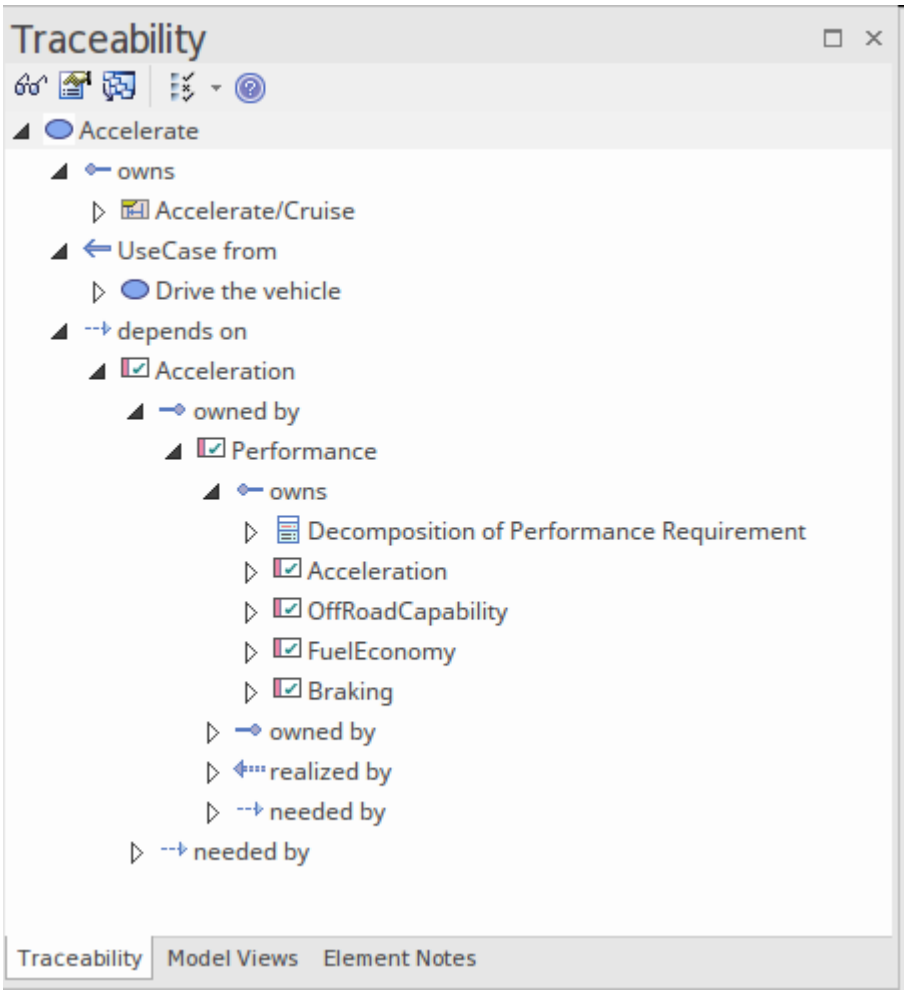
Behavioral Models

A systems engineer can describe how the structural elements of a model behave using a series of diagrams collectively known as the Behavioral model. Structural elements exhibit behavior in a running system, and a number of the structural elements themselves have behavioral features such as operations. The system modeling language specification classifies the following diagrams as behavior. They are all used to represent different aspects of a system's behavior, from the Use Case that describes the behavior that is

valuable to a user to a sequence diagram that articulates how elements interact.

Use Case Diagram

Use cases and actors are high-level representations of a system's behavior from the user's point of view. An engineer models the value that a user performing a role with respect to the system derives from the system behavior. Use Case will typically be traced to other elements such as requirements and structural elements such as blocks.

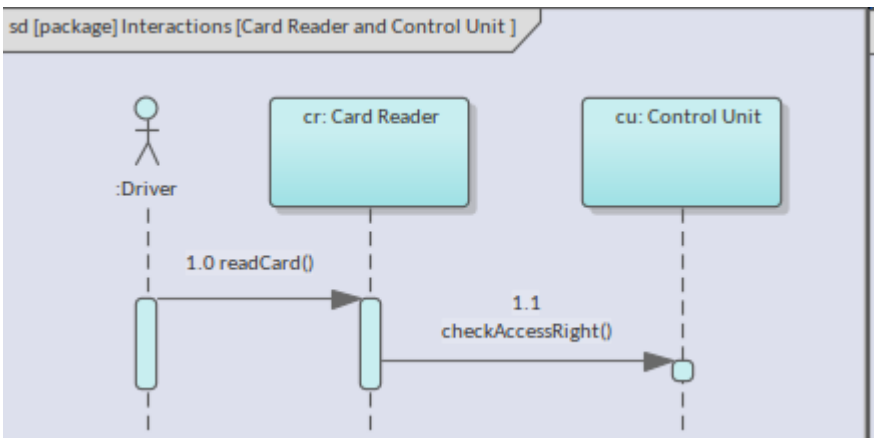


Activity Diagram

Activity diagrams are flow-based models that describe a system's behavior by articulating the flow of items, including information and physical items that act as inputs and outputs to activities and actions as the system performs work.

Sequence Diagram

A modeler uses the sequence diagram to describe the way messages flow between parts and properties of blocks. The messages are sequenced and are typically implemented by behaviors such as operations owned by the block.



State Machine Diagram

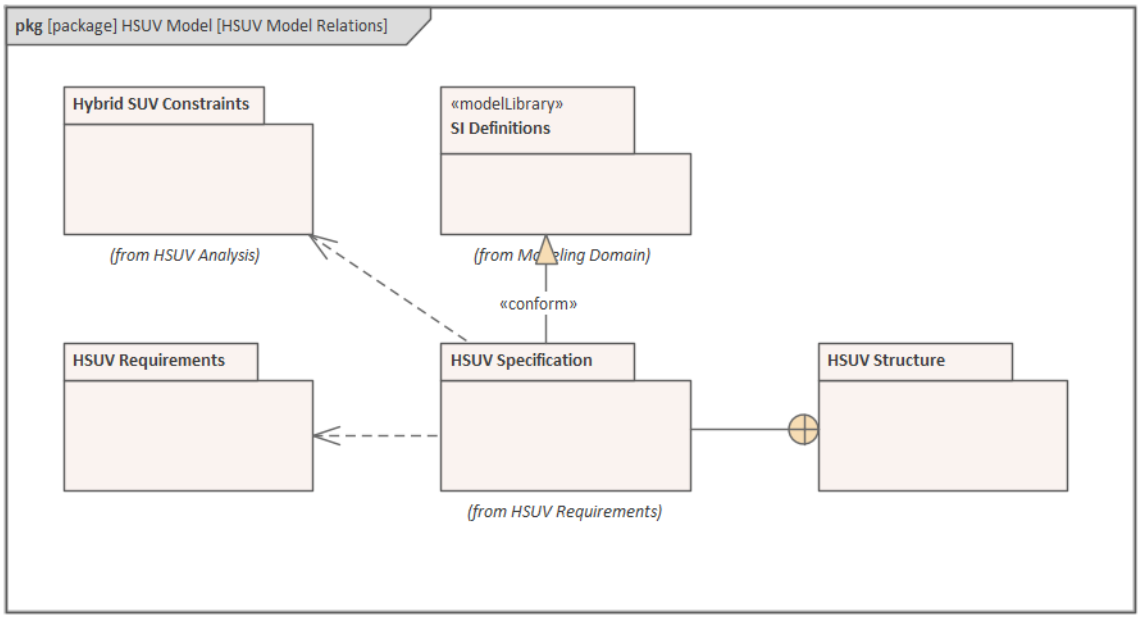
A system engineer uses state machine diagrams to describe how structural elements such as blocks behave in response to events that fire and how the states that a block exhibits a transition from one state to another.

Structural Models

A systems engineer can describe the structure of a system using a series of diagrams that together form the structural model. These diagrams and the elements and relationships they contain, define the components of the system that are bought to life by the behavioral models. The system modeling language specification classifies the following diagrams as structural. They are all used to represent different aspects of a system's structure at a logical and physical level, from the packages that organize the model to parametric diagrams that define equations and their input and output parameters.

Package Diagram

A complex system needs to be organized to ensure both humans and other systems can understand, digest, and locate items of interest in the model. Packages that appear in the Browser window can also be placed onto diagrams and are the primary element used to structure the model.



Block Definition Diagram

Blocks are the fundamental structural element in a system's representation; they contain features, exhibit behavior, change states, and interact with other blocks to produce the behavioral contracts of the system.

Internal Block Diagram

The usage of blocks and their parts are described on an Internal Block diagram using parts, ports, interfaces, and relationships, including flows that describe items that pass between interconnected blocks.

Parametric Diagram

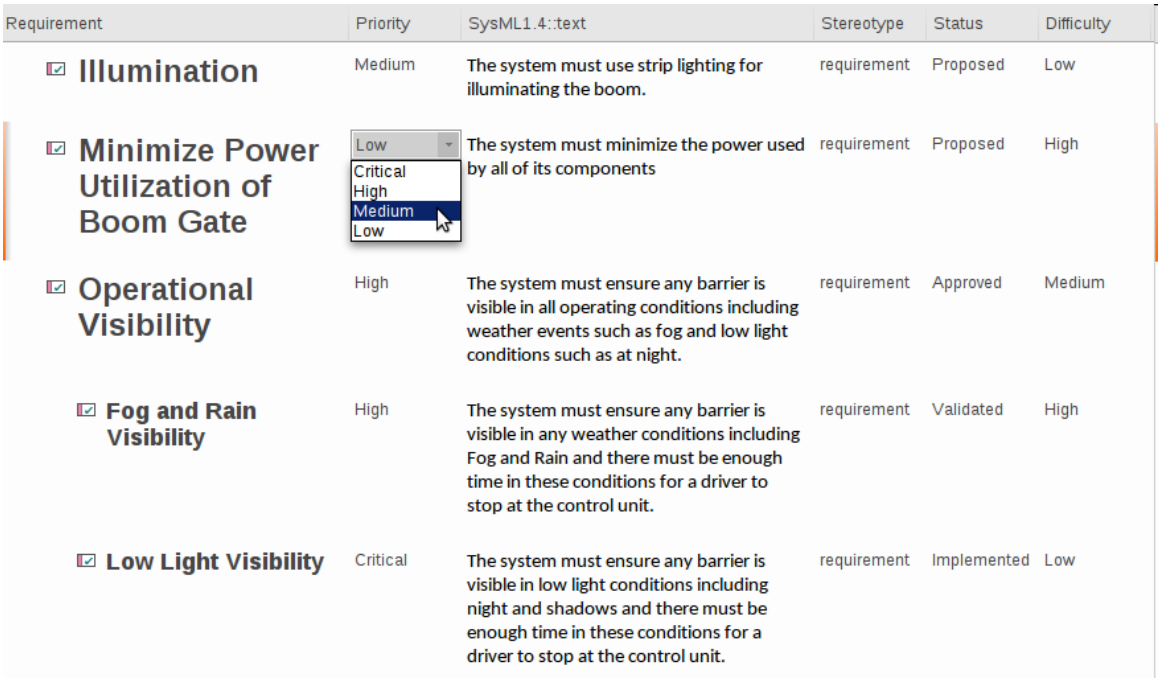
A property's constraints are defined using parametric diagrams that model engineering and mathematical equations and their input and output paramters.

Defense and Commercial Architecture Models

A number of frameworks have been used to model large systems or systems-of-systems in defense organizations and large commercial or industrial organizations. These frameworks are based on modeling languages such as the Unified Modeling Language (UML), the Systems Modeling Language (SysML), and Service-Oriented Architecture standards. The frameworks have evolved over several decades as defense, and commercial systems and projects have become larger and more complex. For example, DoDAF and MODAF have been combined to form the basis for the Unified Profile for DoDAF/MODAF (UPDM), and this, in turn, has evolved into Unified Architecture Framework (UAF). Enterprise Architect has rich support for both UPDM and UAF, and systems engineers can create robust, expressive, and compliant defense and commercial models that provide views of complex systems or systems of systems.

Requirements Models

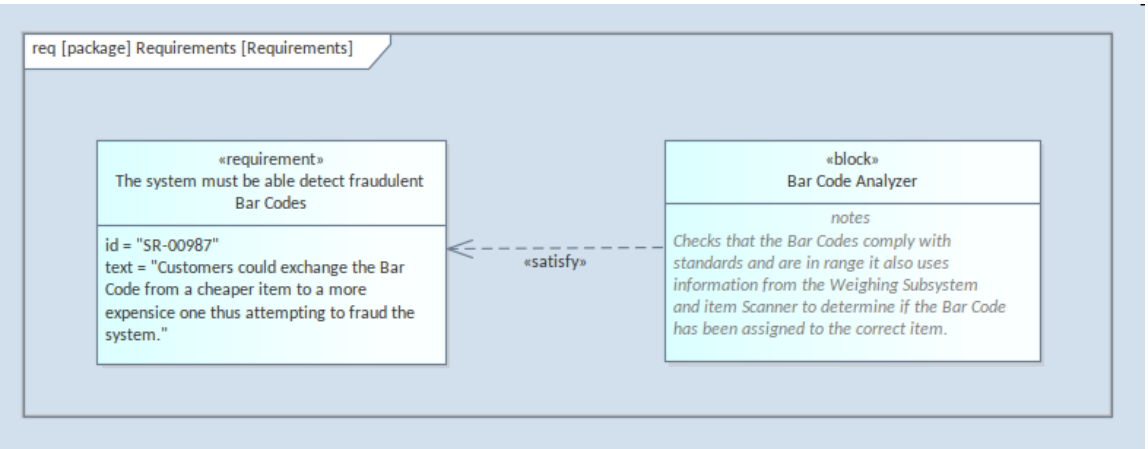
Requirement Engineering is one of the most important disciplines in the system lifecycle. When done well, it will set the foundation for a successful project or program of work, ultimately ensuring that engineering teams deliver great value to the users and other stakeholders. Enterprise Architect is a sophisticated and intuitive platform for developing and managing Requirements gleaned from modeling stakeholder statements, business cases, business drivers, and capabilities to define detailed Functional and Non-functional Requirements. The engineer can prioritize, trace, and track requirements and record changes, baseline, version, and record audits of changes. Engineers, managers, consultants, and customers can work together in a collaborative platform with role-based Security, Discussions, a team Library, Model Mail, and a range of other tools to encourage best practice and productivity, either directly on the local system or through Pro Cloud Services.



Requirement	Priority	SysML1.4::text	Stereotype	Status	Difficulty
Illumination	Medium	The system must use strip lighting for illuminating the boom.	requirement	Proposed	Low
Minimize Power Utilization of Boom Gate	Low (Critical, High, Medium, Low)	The system must minimize the power used by all of its components	requirement	Proposed	High
Operational Visibility	High	The system must ensure any barrier is visible in all operating conditions including weather events such as fog and low light conditions such as at night.	requirement	Approved	Medium
Fog and Rain Visibility	High	The system must ensure any barrier is visible in any weather conditions including Fog and Rain and there must be enough time in these conditions for a driver to stop at the control unit.	requirement	Validated	High
Low Light Visibility	Critical	The system must ensure any barrier is visible in low light conditions including night and shadows and there must be enough time in these conditions for a driver to stop at the control unit.	requirement	Implemented	Low

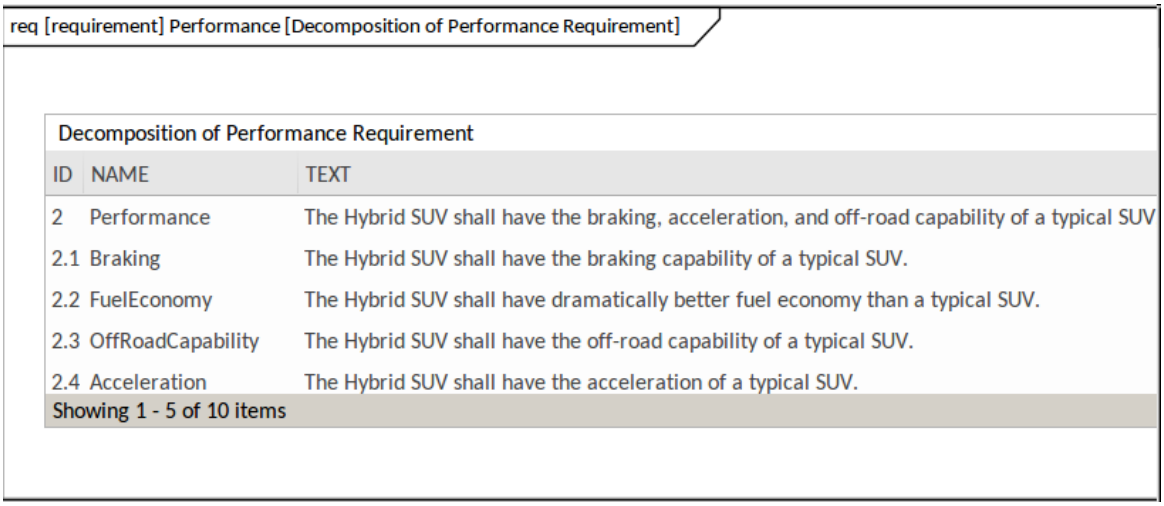
Requirements Diagram

A systems engineer uses a requirements diagram to create and view Requirements and their relationships to other elements, including other Requirements. You can specify requirements at any level, from strategic enterprise or business requirements through stakeholder requirements down to low-level engineering and even software and transition requirements. Requirements properties, including their id and text, can be displayed or suppressed on a diagram; the choice depends on the diagram's purpose and its intended audience.



Requirements Discipline

Requirement development includes all the activities and tasks associated with discovering, evaluating, recording, documenting, and validating the requirements for an engineering project or program of work. Requirements are discovered, analyzed, specified, and verified, and Enterprise Architect has a wide range of tools and features to assist the requirement engineer as they develop requirements. The Specification Manager is the centerpiece for requirement development, allowing the Analyst to enter, view, and manage requirements in textual form in a spreadsheet or document-like view.


req [requirement] Performance [Decomposition of Performance Requirement]

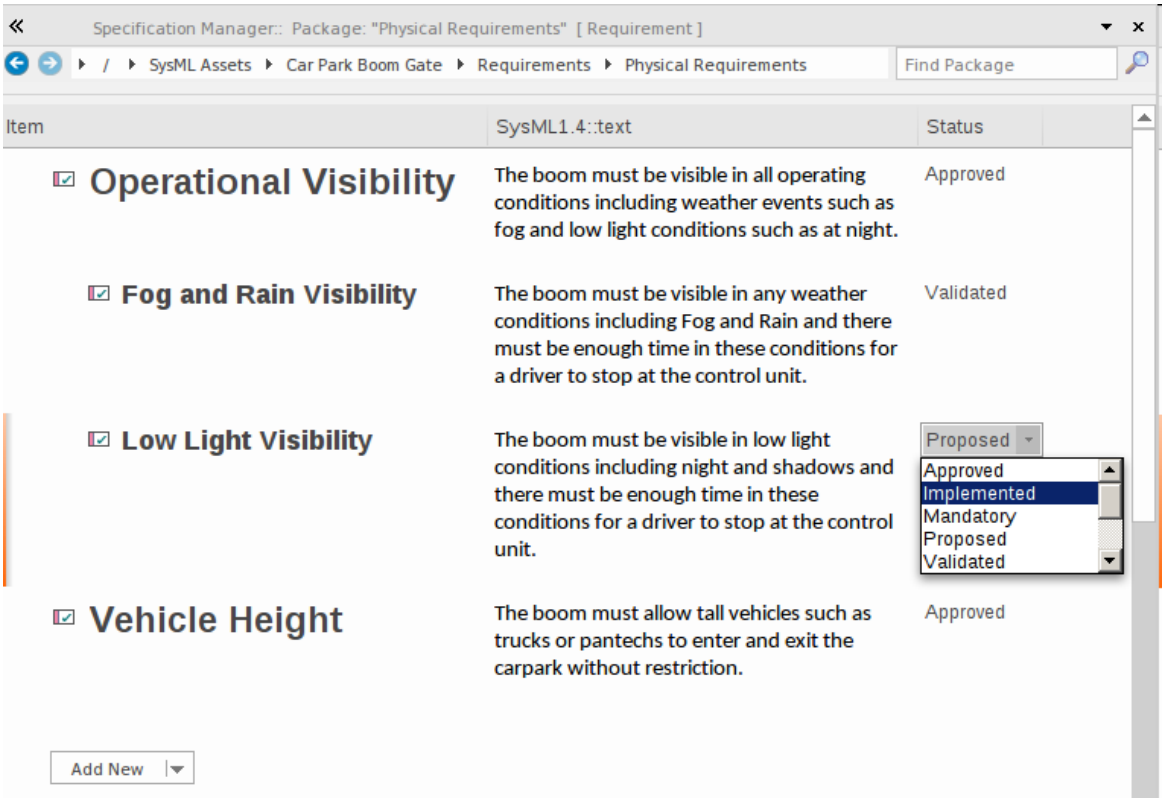
Decomposition of Performance Requirement

ID	NAME	TEXT
2	Performance	The Hybrid SUV shall have the braking, acceleration, and off-road capability of a typical SUV
2.1	Braking	The Hybrid SUV shall have the braking capability of a typical SUV.
2.2	FuelEconomy	The Hybrid SUV shall have dramatically better fuel economy than a typical SUV.
2.3	OffRoadCapability	The Hybrid SUV shall have the off-road capability of a typical SUV.
2.4	Acceleration	The Hybrid SUV shall have the acceleration of a typical SUV.

Showing 1 - 5 of 10 items

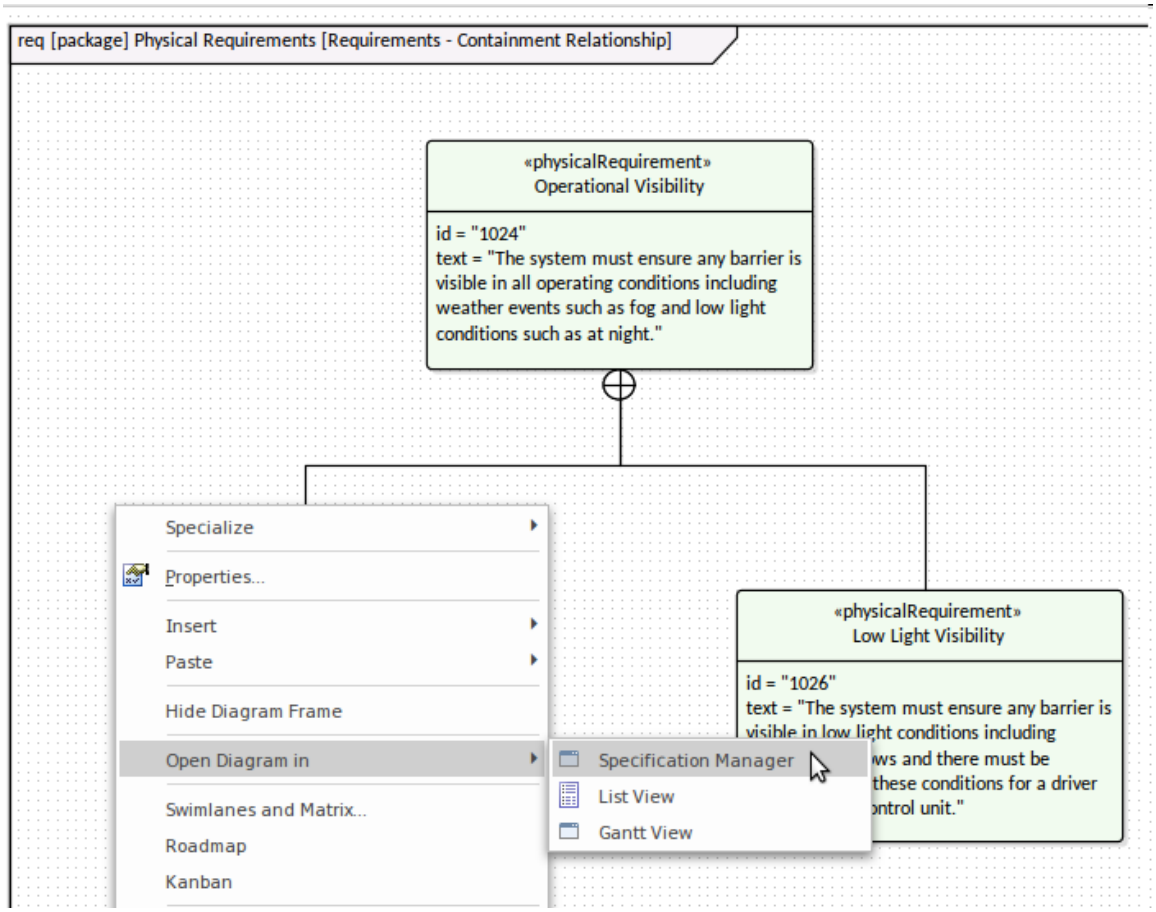
Requirements Tools

The Specification Manager is an easy-to-use tool providing a spreadsheet or word processor view that you can use to manage requirements or any other model element. It is particularly beneficial when working with requirements that have descriptive text to describe the requirement in detail. An engineer can create new requirements with names and detailed descriptions, and properties such as Status and Priority can be added or changed from drop-down lists. You can conveniently view and manage existing requirements using other diagrams and windows - and changing them in the Specification Manager will change them in all other places in the repository.



The Specification Manager is the perfect tool for those analysts who are more comfortable working with text rather than diagrams and who are accustomed to working in a Word Processor or Spreadsheet. It has the added advantage that the Requirements are part of a model, and an engineer can trace them to other elements, including Business Drivers, Stakeholders, and blocks. This image shows that you can specify and manage the Requirement status and other element properties using drop-down lists.

An engineer can open diagrams and packages containing requirements in many views, including the specification manager.

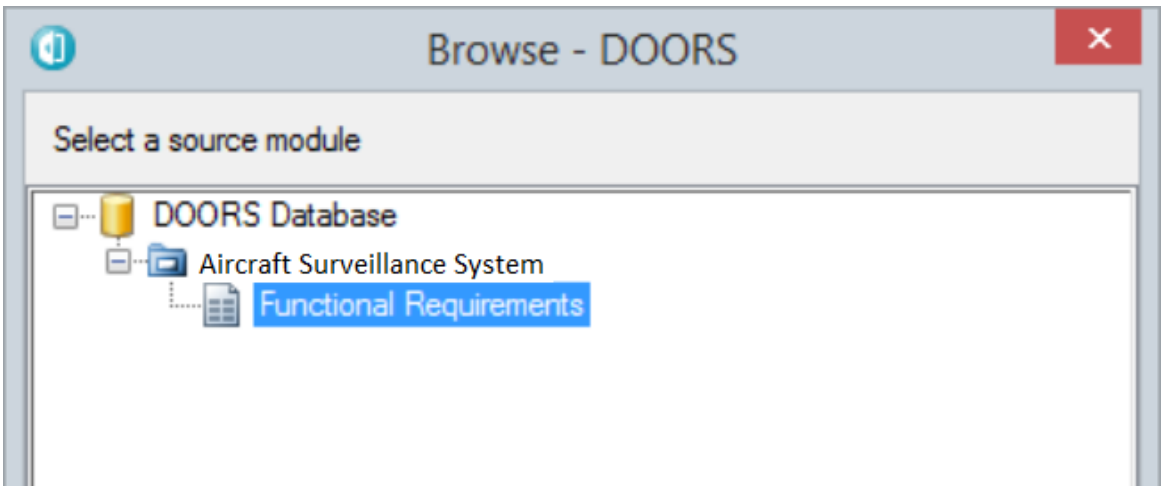


DOORS Integration

The designers of Enterprise Architect understand that customers may have existing or incumbent requirement management tools that they use as part of a corporate or engineering policy. To allow the full scope of modeling and traceability, Enterprise Architect integrates with both the legacy DOORS application and the newer DOOR Next Generation tool, allowing you to view the DOORS requirements inside the and trace to both up-process and down-process elements thes requirements.

DOORS MDG Link (Legacy method)

In the MDG Link for DOORS, you can create a link between Sparx Systems Enterprise Architect and an existing IBM® Rational® DOORS® module, which enables you to exchange requirements data between DOORS and Enterprise Architect. You can also redirect the link to a different module. You can import data from DOORS to Enterprise Architect and export data from Enterprise Architect to DOORS through this link. You can both import requirements from DOORS or export requirements located in the Enterprise Architect repository to DOORS.



DOORS NG Integration

Using the Pro Cloud Server Jazz Plug-in integration, you can manage various Rational tools, including the DOORS Next Generation Requirement Management tool. This allows you to push and pull requirements from any configured DOORs project. The tools include:

- IBM Rational DOORS Next Generation's Requirement management tool

- Rational Rhapsody Design Management (DM)
- Rational Team Concert Change and Configuration Management (CCM)
- Rational Quality Manager (QM).

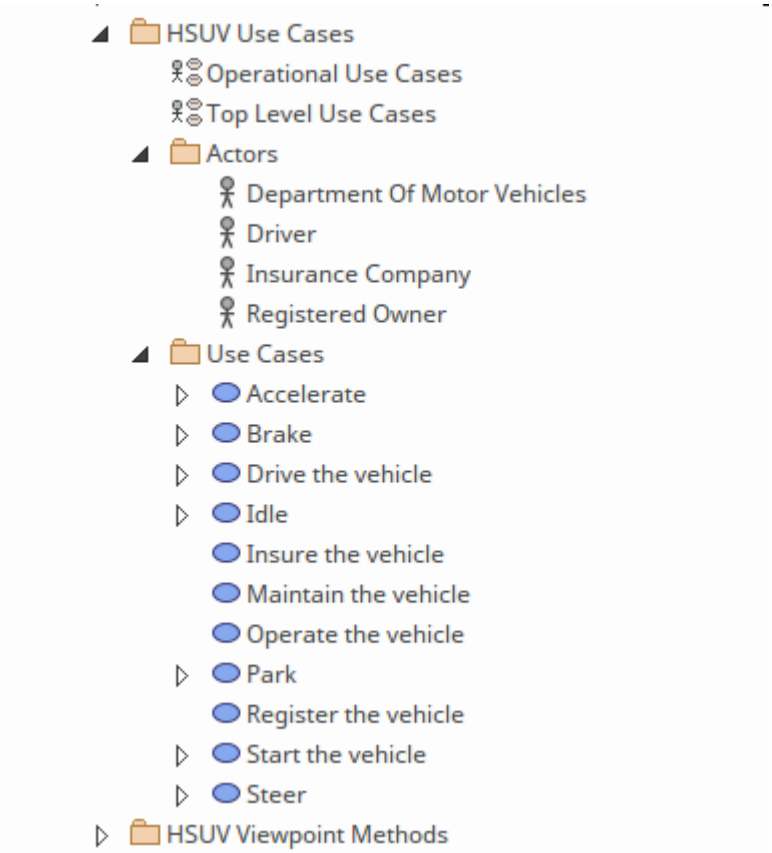
Structural Models

The structural models contain the 'nouns' of the system and define the structures or components of the system. Packages are the primary element for structuring a model or repository and act as containers or namespaces for other elements and their features, including other packages. The fundamental element of structure is the Block which can contain both structural and behavior features and can be used to model any logical or physical aspect of a system. Blocks are typically created and viewed on Block Definition Diagrams and also appear on Internal Block diagrams that you use to describe the usage of the Block in a particular context showing the parts that make up the Block. Parametric diagrams are a specialized type of Internal Block diagram used for modeling mathematics and physics equations.

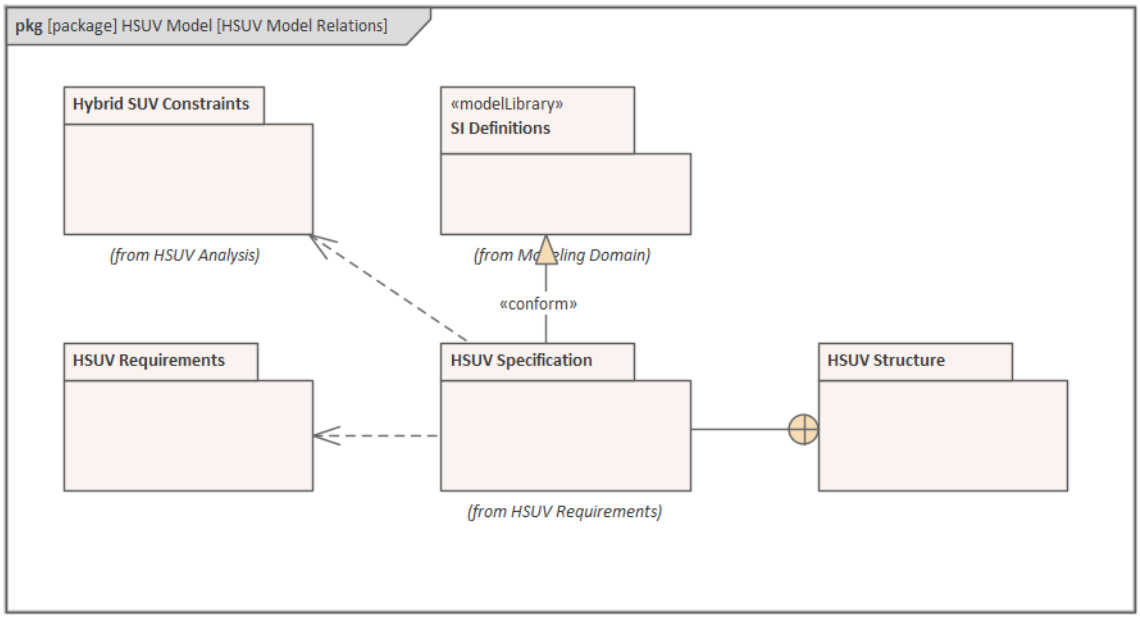
Structure with Packages

The organization of a model is critical to the success of a project or whole of engineering level endeavor. The Package is one of the primary and important elements in the SysML for defining structure. It functions as a container and viewed simply, it is resembles a folder in your favorite file explorer software for your computer. So, in this way, it is firstly a container that groups together other elements, including other Packages but it also has other important

functions in Enterprise Architect including for version control, baselining, publications and more.



Packages diagrams can also be used to visualize the structure of a repository and have the advantage that they can be included in publications or web views of the repository.

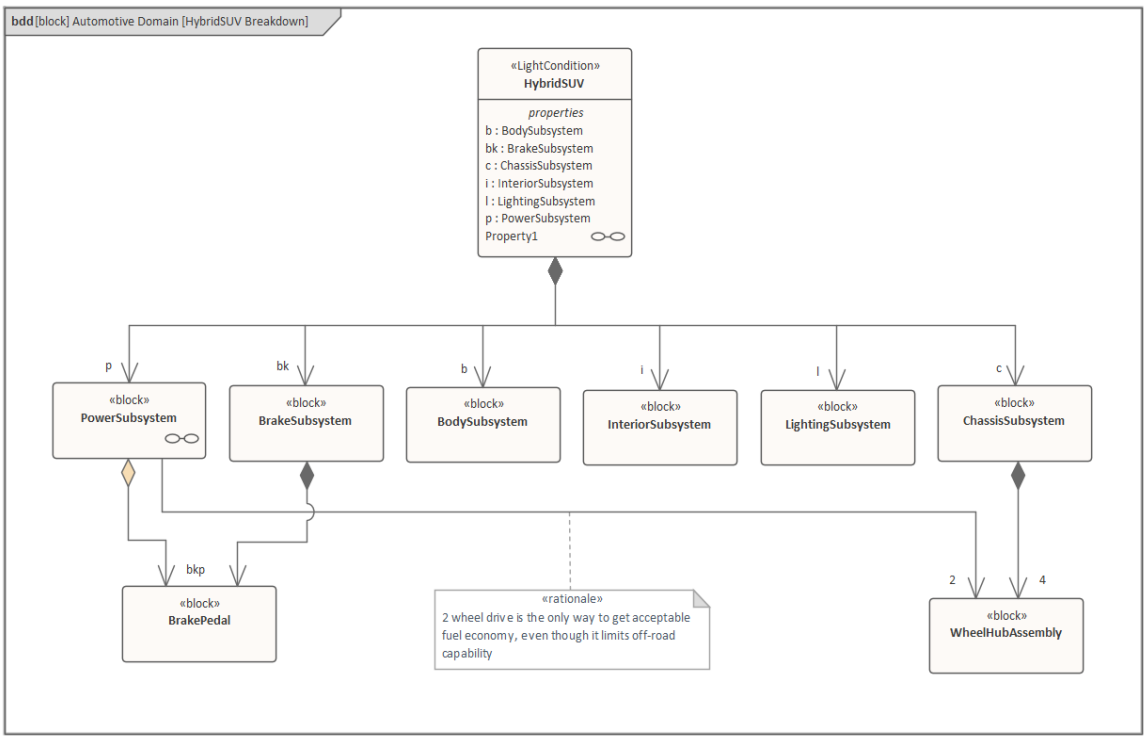


Blocks and Constraints

The SysML has similar grammatical categories found in natural languages, with elements that describe structure and other elements that describe behavior. The SysML describes structural things (nouns) using a Block. When engineers create diagrams, they will often use a mixture of behavior or structure elements, describing a particular aspect of a system - bringing to light some aspect of the modeled system.

The Block is the fundamental unit of system structure and is used to describe an entire system, a subsystem, a component, an item that flows through a system, a constraint, or entities that reside outside a system. Similar to our natural languages, a Block can represent something

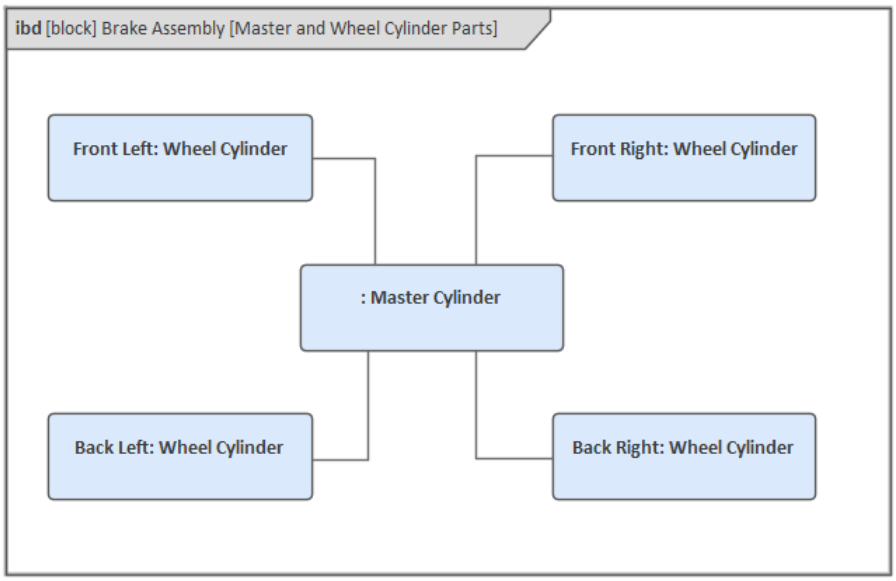
abstract, logical, or physical. This is an important concept, and the SysML writers and readers must be clear about the intention of the representation. For example, in a logical architecture, there are typically Blocks representing conceptual ideas or designs that physical and tangible components might realize at the time of detailed design and construction. A systems architect might define a Block called Collision Detection Subsystem that is an expression of a logical system component that could at the detailed design phase, be in part, realized by a set of radar and laser transmitters, detectors and cameras.



Parts and Block Usage

Blocks are classifiers and describe the characteristics of a set of elements that represent how the Block is used in a context. When the Block has attributes (value properties) defined, these are given specific values in the Block instances. Effectively, each Block instance has an identity and typically would have different values assigned that define the Block's state. Enterprise Architect allows these values to be specified using a Set Run State option available from the context menu. Block instances are properties or parts. Thus an engineer working in an automotive domain could define aspects of a vehicle's braking system showing blocks representing a master cylinder's relationship to a wheel cylinder defining a multiplicity of 3..4. The engineer would place instances of these blocks on Internal Block diagrams to express how the parts work together to carry out the behavioral contracts of the system.

The engineer has named each of the wheel cylinder parts (Front Left, Front Right, Back Left, Back Right) as these need to be identified with respect to their location in the vehicle, but has decided not to name the master cylinder as no further qualification is required.

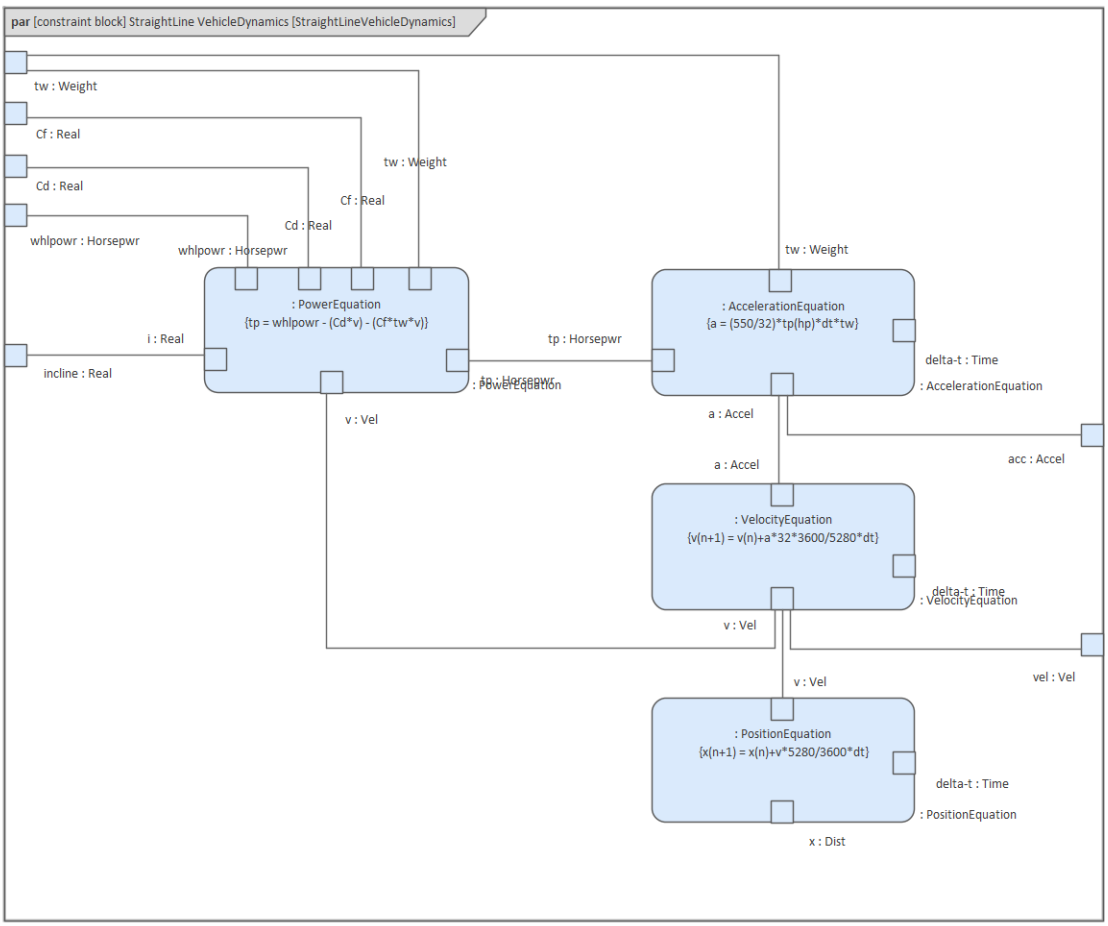


Parametrics and Equations

Systems engineering models created in Enterprise Architect provide a valuable tool for analysis, design, architecture, testing, and visualization. Systems Engineers are charged with finding solutions to problems and opportunities and using models to visualize the system's simplifications under consideration and the system's operation context or environment. This includes predicting how a system will behave in a given context, balancing competing requirements and design considerations in the form of stakeholder negotiations and trade-off analysis. Parametric diagrams are a powerful tool that can assist the engineer in addressing these concerns in a model and pre-emptively to represent how a system is likely to behave.

Constraints can be modeled on a block definition diagram and then Parametric diagrams are used to show how theses

ConstraintBlocks are used in a particular context, being represented on the diagram as ConstraintProperties. We can visualize how the total power parameter is calculated, connecting the Power Equation and the equivalent parameter on the Acceleration Equation. Connections can be seen between the Position Equation and the Velocity Equation, ultimately connected back to the Acceleration Equation.



Behavioral Models

The behavioral models contain the 'verbs' of the system and define how the system behaves from several different viewpoints.

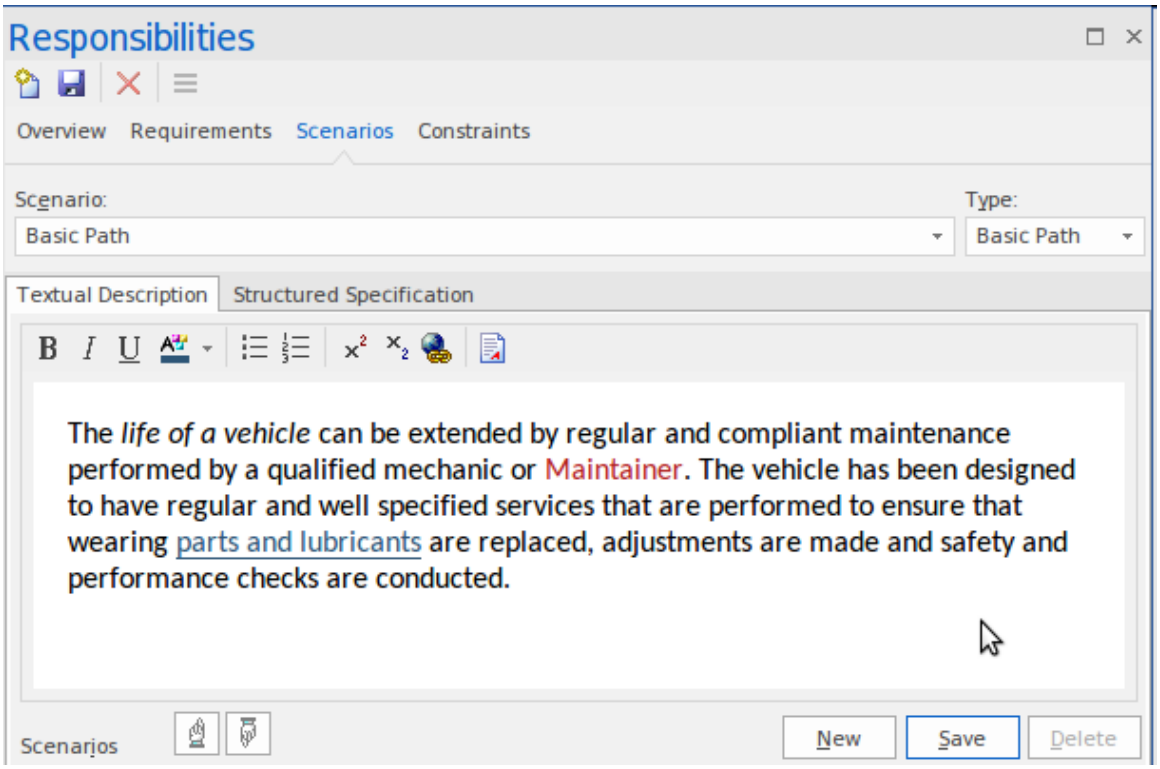
The Behavior diagrams communicate the behavior of the system and demonstrate how the parts of a system work together to satisfy behavioral requirements. Behavioral models have a range of purposes. The engineer must understand what part of the system's behavior they are modeling and then choose the appropriate tool feature and language construct to model this behavior. Systems Engineers use SysML diagrams to model these behavioral characteristics.:

- Use Case diagrams - used to narrow a system's scope and express the users' goals as a value proposition.
- Activity diagrams - used to define the ordered set of actions that carry the work of the system.
- Sequence diagram - used to show how system components or parts interact to produce an outcome.
- State Machine diagram - used to define the discrete states of a system or its parts during its lifetimes.

Enterprise Architect has a range of productivity tools that the systems engineer can use while working with behavioral models, including the Scenario Builder, State Machine Tables, Simulation engine, and many more.

Use Cases and User Goals

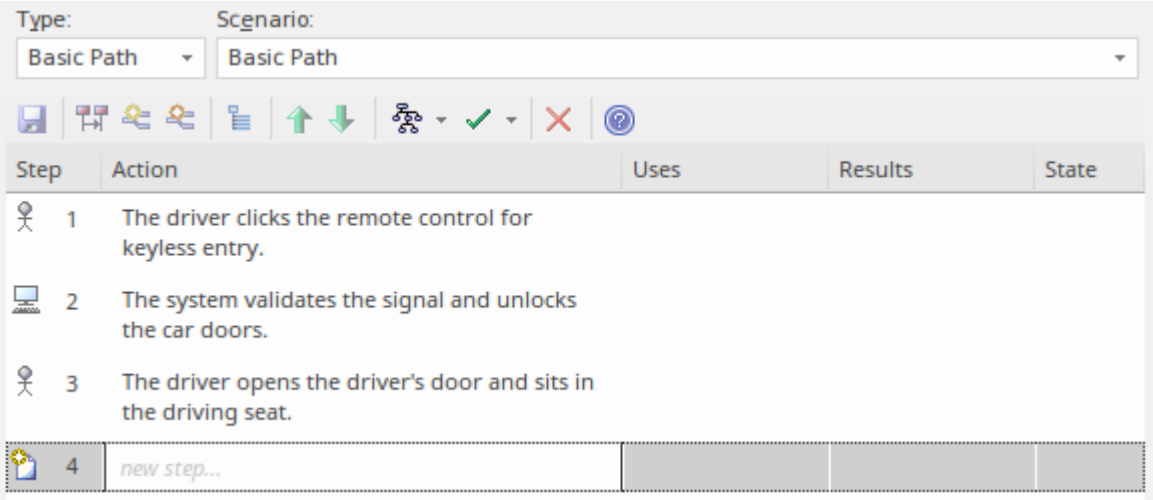
The use case model describes the value or goals that users (human and system) derive from interacting with the system. A brief description summarizes this value for each scenario, including the all-important basic (sunny day) scenario.



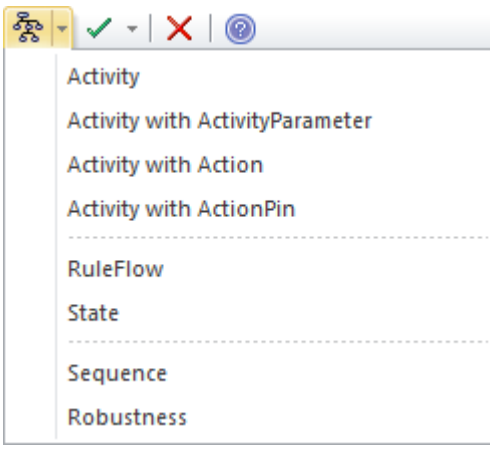
The Use Case technique is fundamentally straightforward and was devised to ensure that functional requirements were written from the User's perspective. This standpoint helped to ensure that deployed systems would be fit for purpose and be accepted by the diverse community of users. There is, however, a vast amount of conflicting literature and an equally large number of styles for defining Use Cases. This situation has led to confusion and uncertainty and has tended to attenuate the value that can be derived from this

simple technique.

Enterprise Architect provides a solution to this by including a purpose-built tool called the scenario editor that the engineer uses to create detailed descriptions of use cases, including alternate and exception paths listing the steps performed by the User and the system.



The tool provides a helpful way to generate behavior diagrams such as activity, sequence and state machine diagrams directly from the scenarios and their steps. These can be synchronized as changes are made to the sequence of steps or to the branch and merge points for alternate and exception scenarios.



Activities and Behavioral Flows

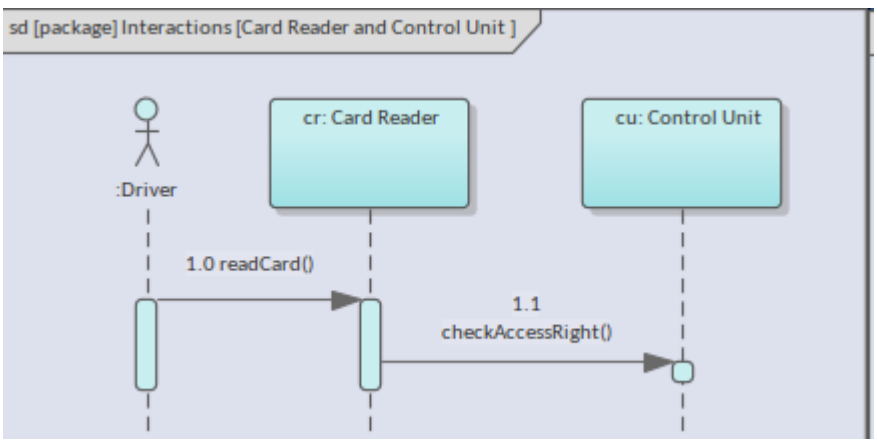
The Activity diagram is an expressive diagram that systems engineers use to show the sequence of actions that describe the behavior of a Block or other structural element. The Actions are sequenced using control flows and can contain input and output Pins that act as buffers for items that flow from one Action to another (or from Control or buffer Nodes). The work carried out by the Actions either consumes or produces these items. The items can be either material, energy, or information, depending on the system and the Activity being described.

Actions are the behavioral atoms that are connected to describe the behavior of an Activity, Sub-system, system, or one of its parts. Effectively an Activity is made up of a set of actions that work together to convert items (tokens) that are input into the Activity to items (tokens) that are output by the Activity. The first Action in a sequence will receive inputs from one of the owning Activity's Input Parameter Nodes. The last Action in the sequence will place the output onto one of the Activity's Output Parameter Nodes. The Actions themselves have input and output devices called Pins - an Action will receive tokens on its Input Pins, perform its work and place the resulting tokens on its Output Pins.

Sequences and Object Interactions

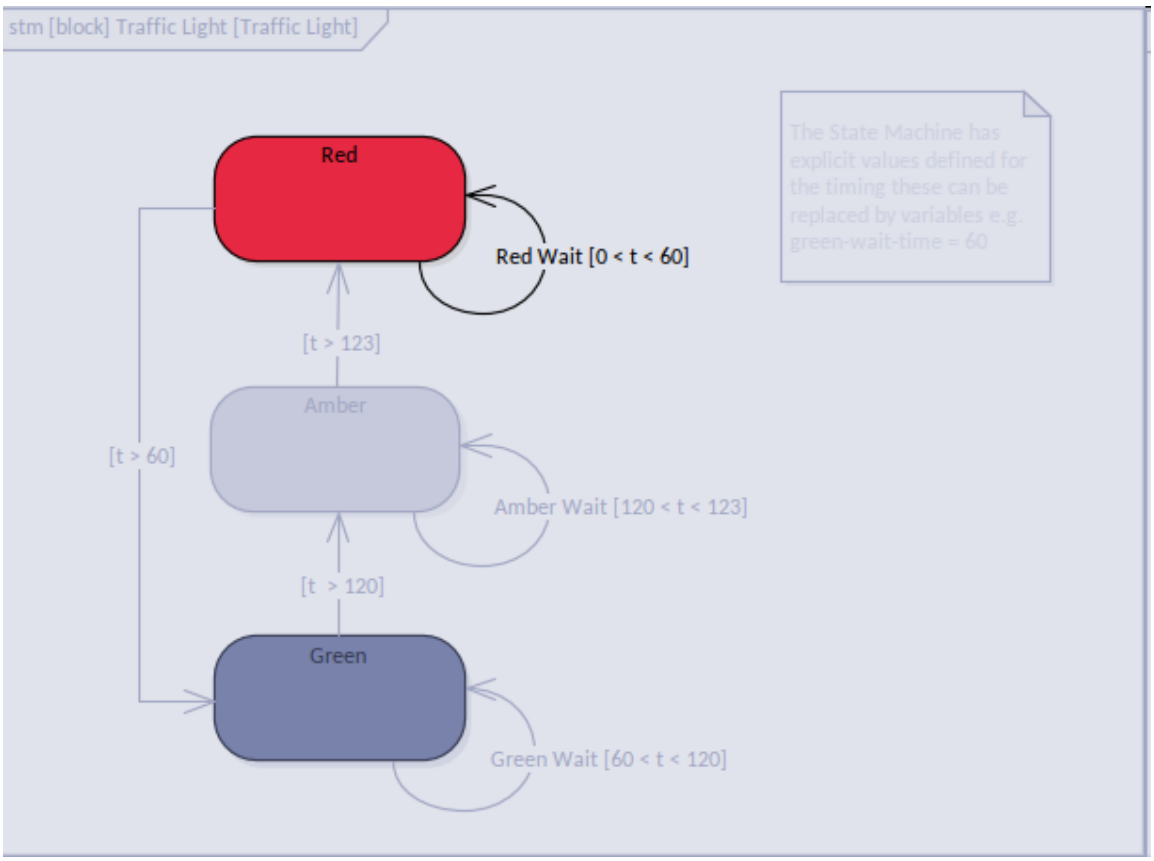
A system is enacted by its part, working collaboratively to carry out the behavior specified in the behavioral models. Instances of structural elements interact by exchanging messages. These interactions can be specified and visualized using sequence diagrams that provide a time-ordered set of messages exchanged between participating instances.

In a Sequence diagram, the Blocks that participate in the interaction have a lifetime that is represented by a dashed line, emanating from the base of the element and continuing vertically for the life of the element. Elements can be created or destroyed at any time during the period represented by the Sequence diagram, and the lifeline therefore represents their existence. Elements that are present at the top of the diagram are created at the beginning of the interaction. A message exchange between a sender and a receiver will originate in one lifeline (the sender) and end in another (the receiver).

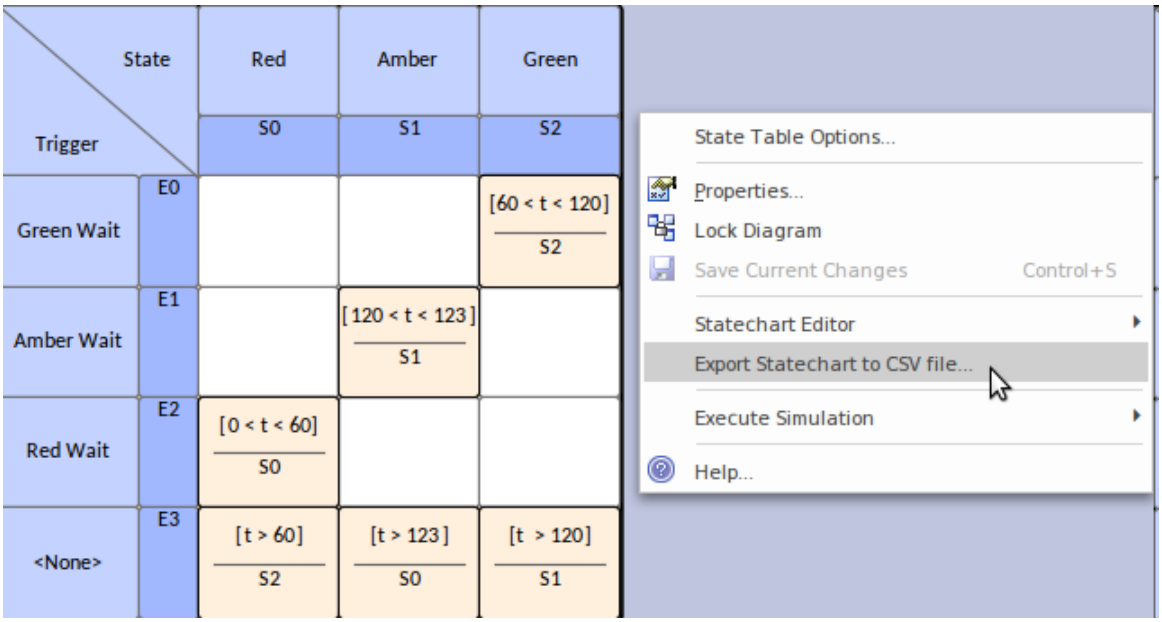


States and Block Lifetimes

The SysML StateMachine is used to describe how structure, modeled with blocks, changes its state in a time-boxed life cycle. Here the engineers' concern is not with the structure of the Block Instance but its behavior, which can, in turn, impact its structure. We are not interested in every single state a 'thing' can exhibit, but rather the significant states. So the critical states for water molecules, for example, could be a solid, liquid, or gas, but we are not ordinarily interested in liquid water at a temperature of 67 degrees Centigrade. If we were looking at a movie reel of an object's lifetime, a StateMachine would pick out the significant frames where significant and relevant changes occurred.



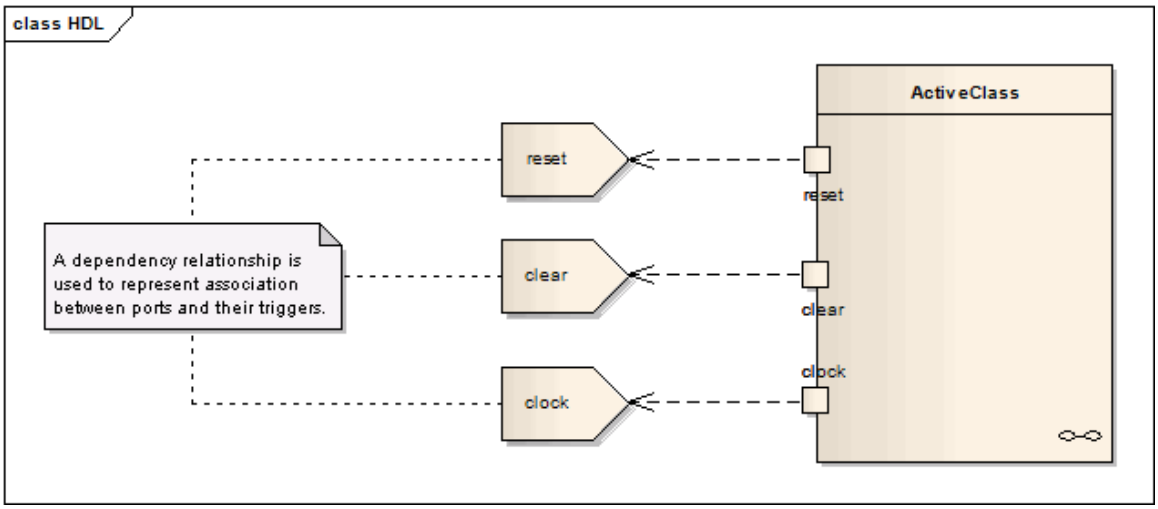
Enterprise Architect also allows an engineer to toggle between diagram and table views of a state machine providing alternative views for engineers and managers who prefer a tabular representation of the state changes. You can also export the tables to a spreadsheet in CSV format for further analysis.



A systems engineer can use state machines to generate executable software code from the model using Executable StateMachines. The code generated is based on its language property. The programming language might be Java, C, C++, C#, or JavaScript; regardless of the language, Enterprise Architect generates the appropriate code immediately ready to build and run.

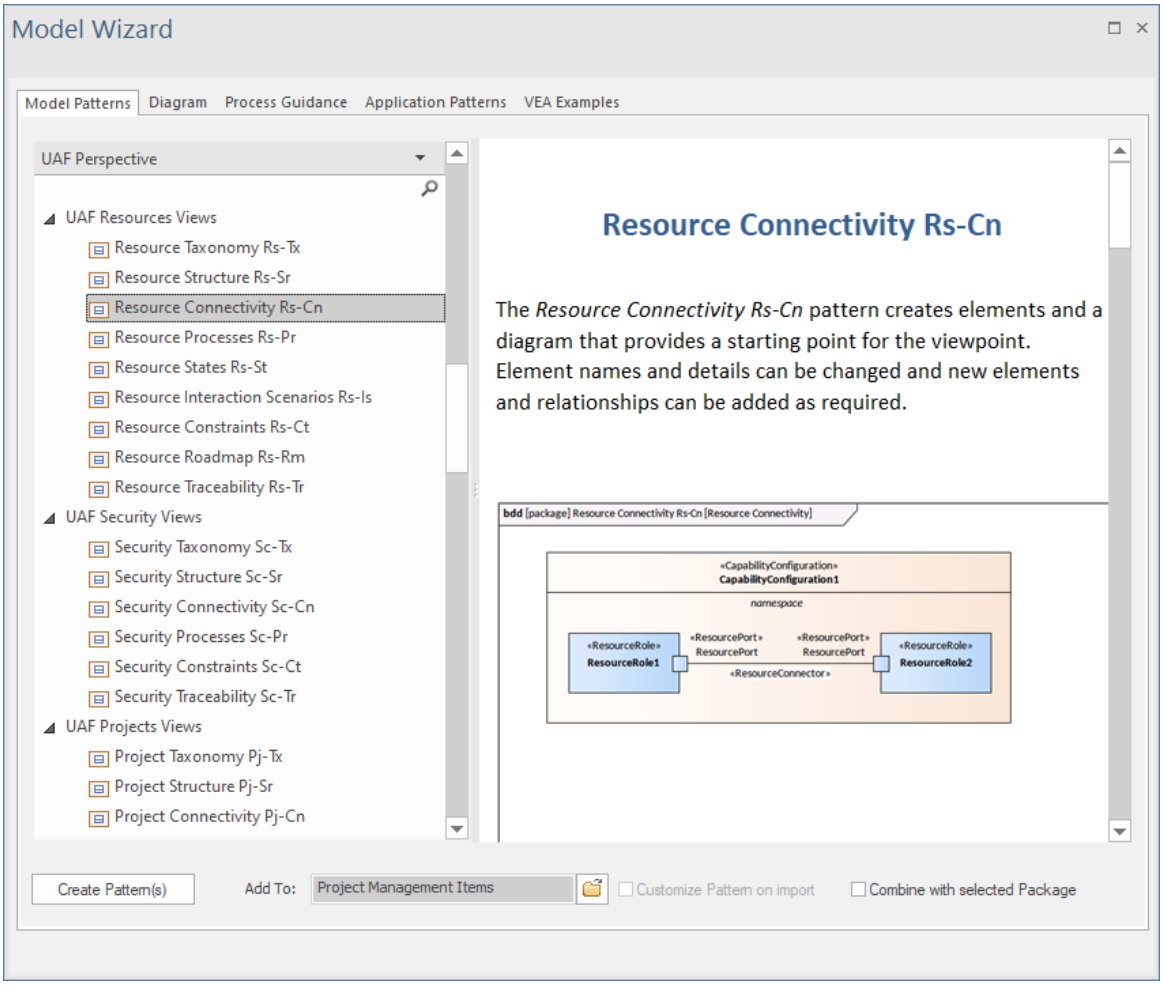
You can not only generate executable software code, but

you can generate Hardware Description Languages and Ada from your model elements, for the chips and circuits in system hardware components.



Defense and Commercial Architecture Models

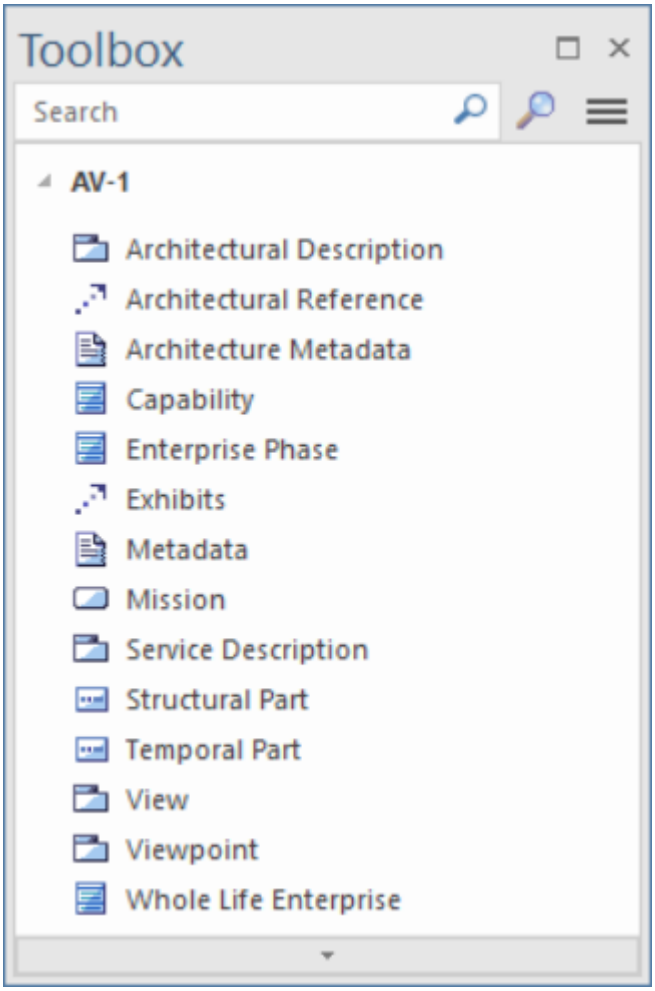
Aerospace and military systems have led the way for model-based system engineering and the use of languages such as the Systems Modeling Language (SysML) and frameworks such as the USA Department of Defense Architecture Framework (DoDAF) and the UK Ministry of Defence Architecture Framework (MODAF) and more recently the Unified Profile for DoDAF/MODAF (UPDM) and the Unified Architecture Framework (UAF).
Large-scale commercial organizations have subsequently adopted these languages and frameworks to develop systems-of-system projects. Enterprise Architect has provided tools for creating language-compliant models and has rich support for the frameworks, including pre-built model patterns for both the language models and frameworks.



Unified Profile for DoDAF/MODAF (UPDM)

UPDM is an acronym for the Unified Profile for DoDAF/MODAF (UPDM), which is a unified framework that supports both the USA Department of Defense Architecture Framework (DoDAF) and the UK Ministry of Defence Architecture Framework (MODAF). Historically these were independent frameworks and had heterogeneous architectures and metamodels; UPDM binds them into a single framework. Enterprise Architect has deep support for

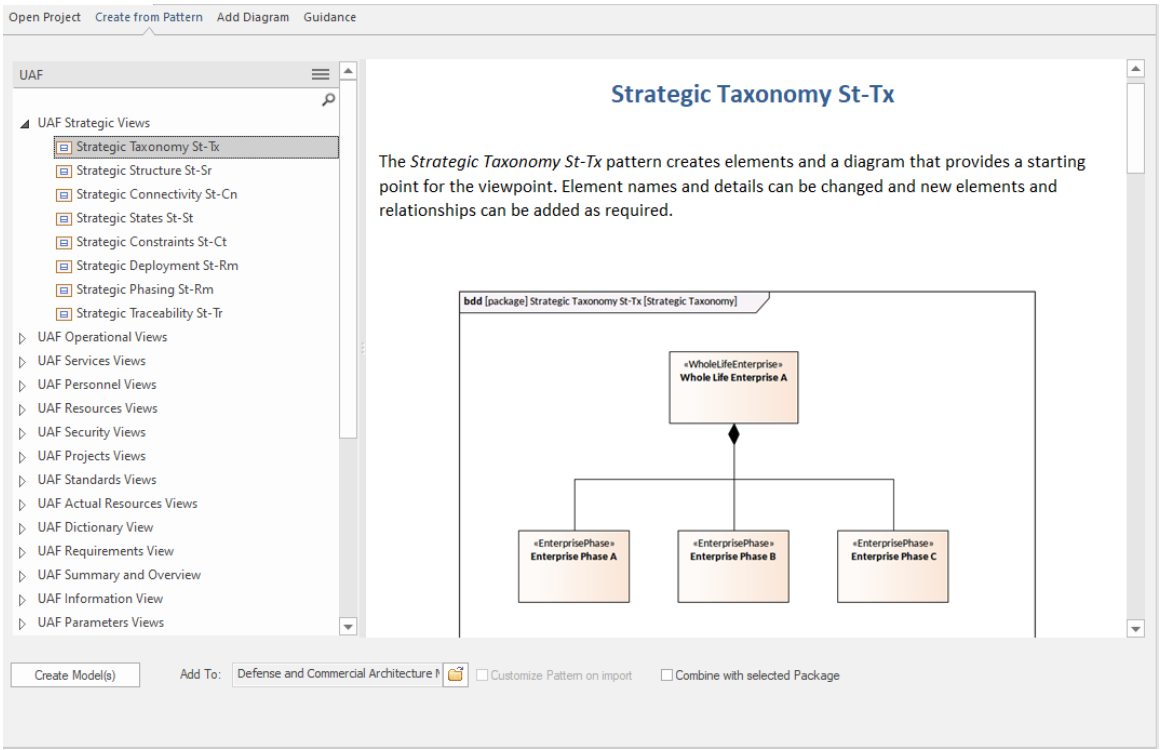
UPDM. The defense systems engineer can create compliant models using a range of toolboxes to create any set of specified viewpoints and views that support the specific needs of stakeholders.



Unified Architecture Framework (UAF)

UAF is an acronym for the Unified Architecture Framework, and the framework is based on the Unified Profile for DoDAF and MODAF (UPDM). UAF defines ways of defining and expressing an enterprise architecture that enables stakeholders to focus on specific areas of

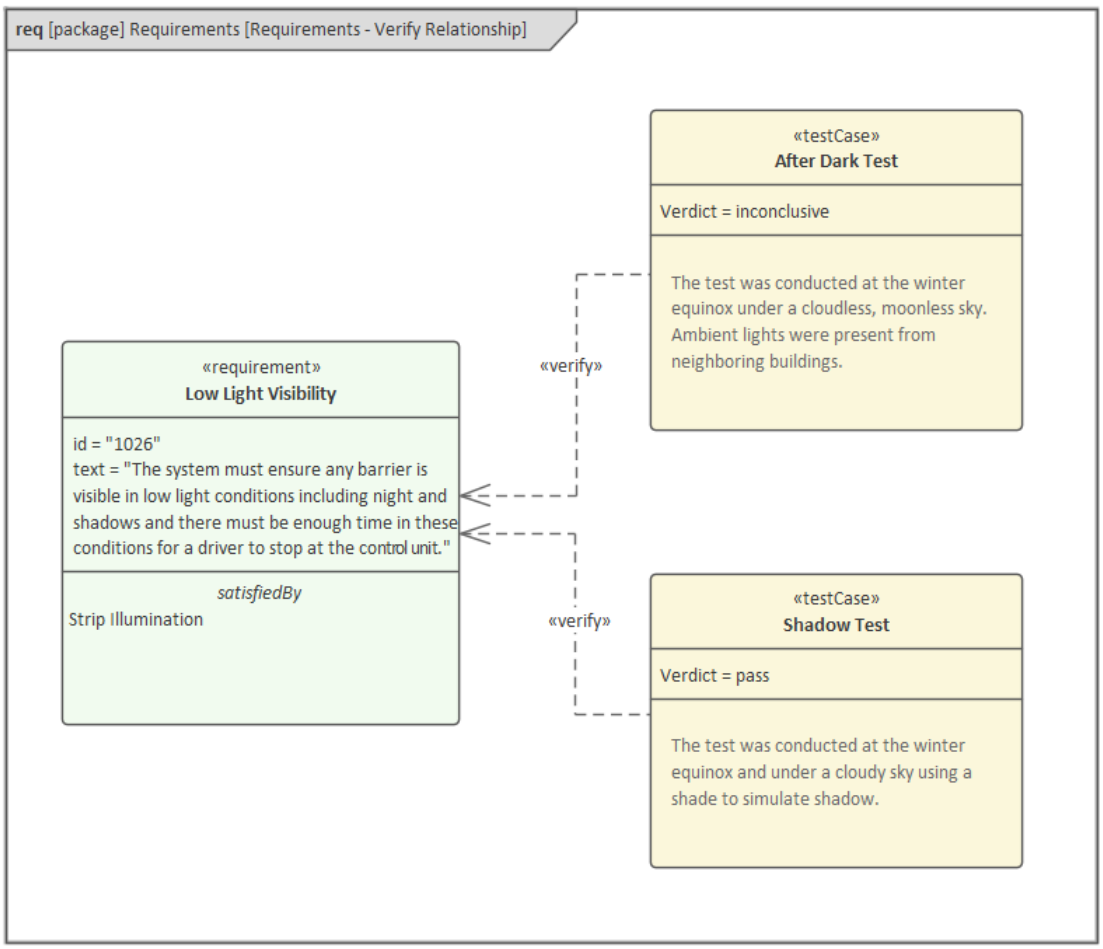
interest while keeping a high-level view of the entire system. UAF was designed to meet the needs of all stakeholders and allows the creation of specific business, operational, and systems-of-systems integration models of commercial and industrial enterprises as well as the U.S. Department of Defense (DoD), the UK Ministry of Defence (MOD), the North Atlantic Treaty Organization (NATO), and other defense organizations models. Enterprise Architect has a full suite of pre-built patterns with accompanying documentation that can be automatically injected into your models.



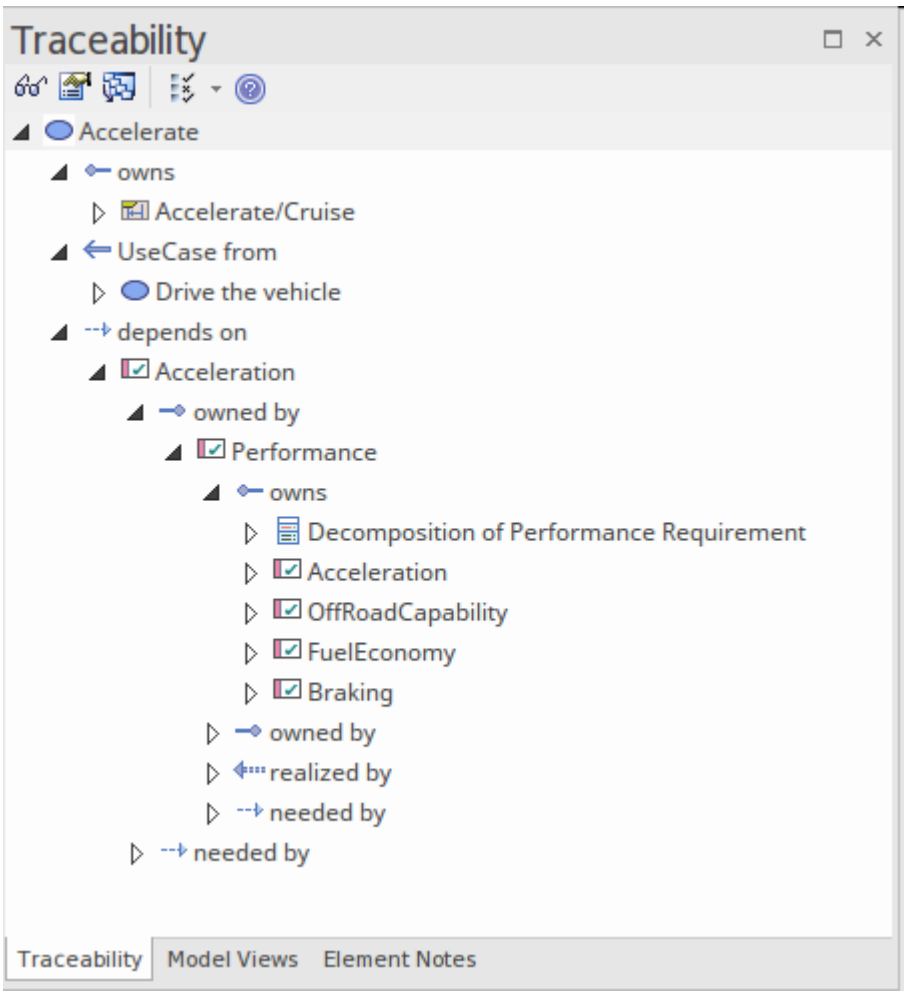
Verification and Validation

Enterprise Architect has a full feature set for verification and validation. Broadly verification is best described as 'has the correct system been built' and validation as 'has the system been built correctly'. The models that the systems engineer creates in Enterprise Architect can be used to verify that the correct system is being built using the traceability features including diagrams the traceability window and the element matrix.

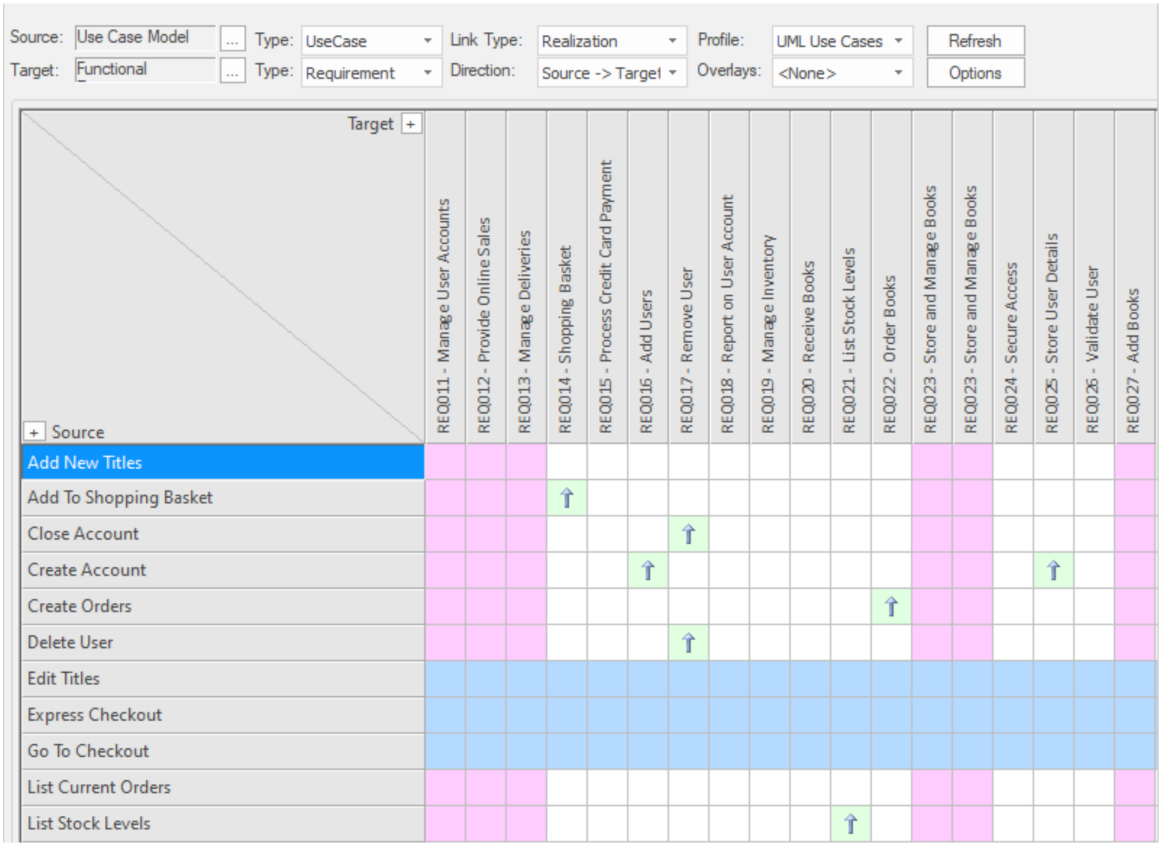
An engineer can use a requirements diagram to express the relationship between Requirements and test cases effectively verifying that the requirements a met by the system.



Modelers can use the traceability window to find and visualize connections between elements. The window is particularly useful for understanding analyzing and verifying that the system's requirements have been implemented by some part of the built system.



A system engineer or project manager can use the matrix window with its tabular view to quickly understand and verify if the system's requirements have been linked to the parts of the system. Colored bands indicate



Enterprise Architect also has a large feature set for specifying tests and recording the results. Enterprise Architect is not only a Systems and Software Modeling environment, it is also a complete Test Management environment. Using Enterprise Architect you can create and manage test scripts for model elements, developing unit, integration, scenario, system, acceptance and inspection tests; these can include test cases generated from xUnit testing and Testpoint Management. Validation and testing teams can also record the test results using the test windows.

You can also import or move tests from other elements, generate them from scenarios, and generate test documentation and reports; you can indicate the presence of tests on an element by displaying test information in a compartment of the element in a diagram, which in turn can be included in documentation and browser based views of

the elements.

Simulation and Visualization

One of the great benefits of creating system models is the ability to see the system in motion by running simulations - effectively you to visualize and analyze a system. Enterprise Architect provides a suite of tools to simulate the execution and behavior of the processes that your models define. The tools provide facilities to dynamically simulate a range of types of model including SImulation of Activity and State Machine diagrams, Parametric diagrams, Decision Models and more. Executable State Machines provide rich support for programming language-specific implementations. OpenModelica and MATLAB Simulink can be used to support rapid and robust evaluation of how a SysML model will behave in different circumstances.

Dynamic Simulation

Model Simulation brings your behavioral models to life with instant, real-time behavioral model execution. Coupled with tools to manage triggers, events, guards, effects, breakpoints and Simulation variables, plus the ability to visually track execution at run-time, the Simulator is a multi-featured means of 'watching the wheels turn' and verifying the correctness of your behavioral models. With Simulation you can explore and test the dynamic behavior of models. In the Corporate, Unified and Ultimate Editions, you can also use JavaScript as a run-time execution language for evaluating

guards, effects and other scriptable items of behavior.

Extensive support for triggers, trigger sets, nested states, concurrency, dynamic effects and other advanced Simulation capabilities, provides a remarkable environment in which to build interactive and working models that help explore, test and visually trace complex business, software and system behavior. With JavaScript enabled, it is also possible to create embedded COM objects that will do the work of evaluating guards and executing effects - allowing the Simulation to be tied into a much larger set of dependent processes. For example, a COM object evaluating a guard condition on a State Transition might query a locally running process, read and use a set of test data, or even connect to an SOA web service to obtain some current information.

As Enterprise Architect uses a dynamic, script driven Simulation mechanism that analyzes and works with UML constructs directly, there is no need to generate intermediary code or compile simulation 'executables' before running a Simulation. This results in a very rapid and dynamic Simulation environment in which changes can be made and tested quickly. It is even possible to update Simulation variables in real time using the Simulation Console window. This is useful for testing alternative branches and conditions 'on the fly', either at a set Simulation break point or when the Simulation reaches a point of stability (for example, when the Simulation is 'blocked').

In the Professional Edition of Enterprise Architect, you can manually walk through Simulations - although no JavaScript

will execute - so all choices are manual decisions. This is useful for testing the flow of a behavioral model and highlighting possible choices and processing paths. In the Corporate, Unified and Ultimate Editions it is possible to:

- Dynamically execute your behavioral models
- Assess guards and effects written in standard JavaScript
- Define and fire triggers into running Simulations
- Define and use sets of triggers to simulate different event sequences
- Auto-fire trigger sets to simulate complex event histories without user intervention
- Update Simulation variables 'on the fly' to change how Simulations proceed
- Create and call COM objects during a Simulation to extend the Simulation's reach and input/output possibilities
- Inspect Simulation variables at run time
- Set a script 'prologue' for defining variables, constants and functions prior to execution
- Use multiple Analyzer Scripts with differing 'prologues' for running the Simulation under a wide range of conditions

In the Unified and Ultimate Editions it is also possible to simulate BPMN models.

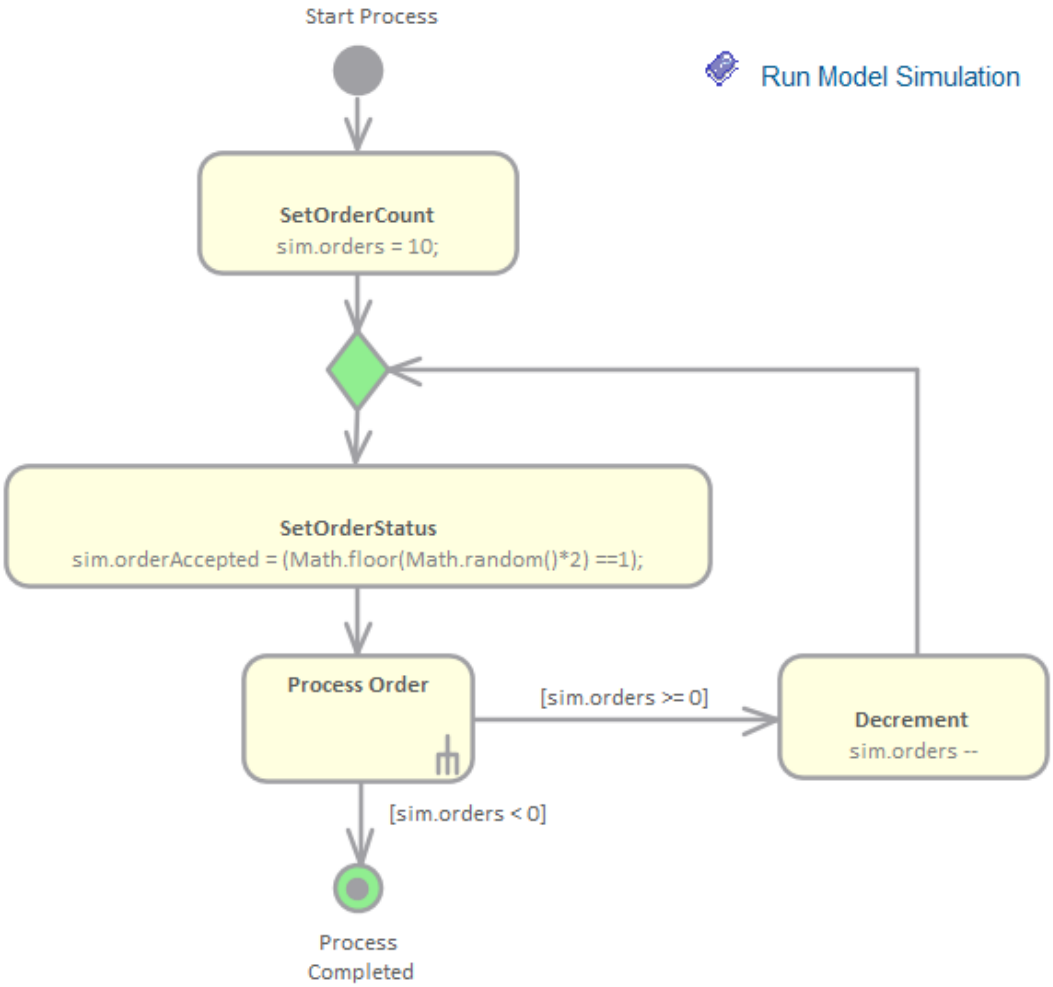
Using the Model Simulator, you can simulate the execution of conceptual model designs containing behavior. When you start a Simulation, the current model Package is analyzed

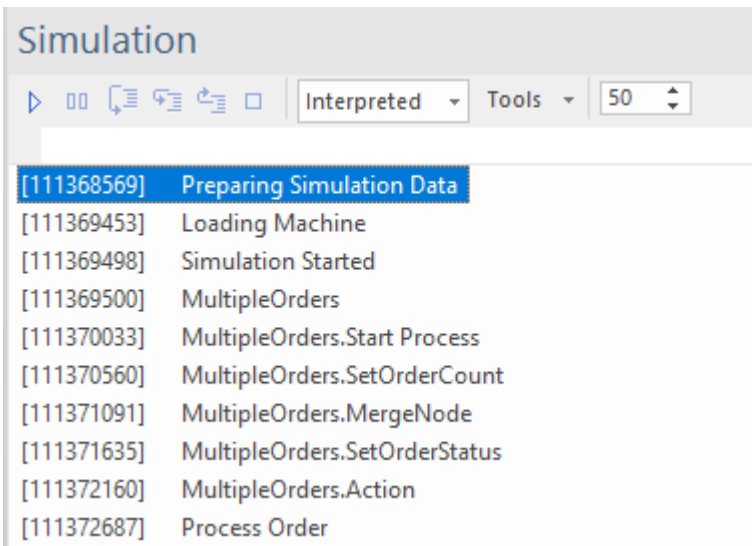
and a dynamic Simulation process is triggered to execute the model.

To get up and running with Simulation, the only steps required are:

- Build a behavioral diagram (State or Activity for manual or dynamic execution, Sequence for manual interaction only)
- Optional: load the 'Simulation Workspace' layout - a fast way of bringing up all the frequently used Simulation windows
- Click on the Simulator Play button

If the diagram contains any external elements (those not in the same Package as the diagram) you will have to create an Import connector from the diagram's Package to the Package containing the external elements. To do this, drag both Packages from the Browser window onto a diagram and then use the Quick Linker arrow to create the connector between them.


Start Process
Run Model Simulation
SetOrderCount
sim.orders = 10;
SetOrderStatus
sim.orderAccepted = (Math.floor(Math.random()*2) ==1);
Process Order
[sim.orders >= 0]
Decrement
sim.orders --
[sim.orders < 0]
Process
Completed


Simulation
Interpreted
Tools
50
[111368569] Preparing Simulation Data
[111369453] Loading Machine
[111369498] Simulation Started
[111369500] MultipleOrders
[111370033] MultipleOrders.Start Process
[111370560] MultipleOrders.SetOrderCount
[111371091] MultipleOrders.MergeNode
[111371635] MultipleOrders.SetOrderStatus
[111372160] MultipleOrders.Action
[111372687] Process Order

Mathematical Simulations

Enterprise Architect provides a wide range of options for introducing advanced mathematical tools and capabilities into your simulations.

You can bring the power of integrated external tools such as MATLAB into your models through the use of Solver Classes, and can also export your models for execution in other external tools such as MATLAB Simulink, Stateflow and Simscape, or OpenModelica.

Enterprise Architect includes an extensive library of mathematical functions within the JavaScript engine, providing the benefits of a significantly expanded Simulation capability.

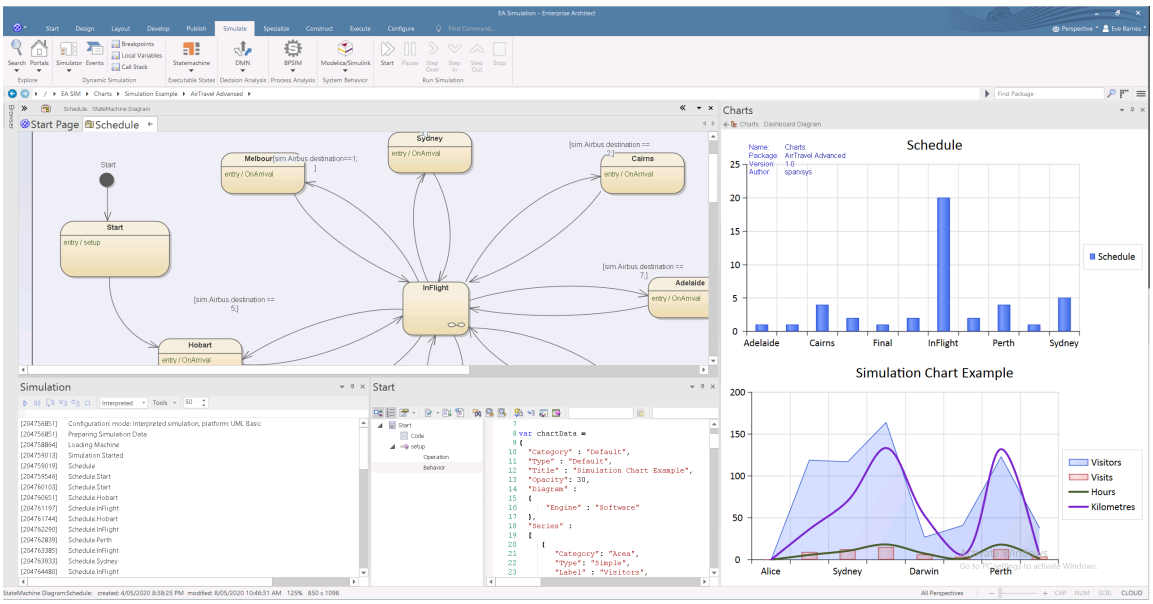
Enterprise Architect also provides a wide range of Dynamic Charts; without the need for external tools, you can configure these Charts to extract and plot information from Simulations that have been directly executed inside Enterprise Architect.

Explore the:

- Solver classes in Enterprise Architect that call MATLAB or Octave to incorporate complex mathematics into your model-based simulations
- Extensive internal Math Library based on the popular Cephes function library
- Integration with the OMG SysPhS standard, enabling you to configure your model for export to common tools
- Support for exporting models to MATLAB Simulink,

Simscape and Stateflow; you can create your model in Enterprise Architect and execute it in MATLAB

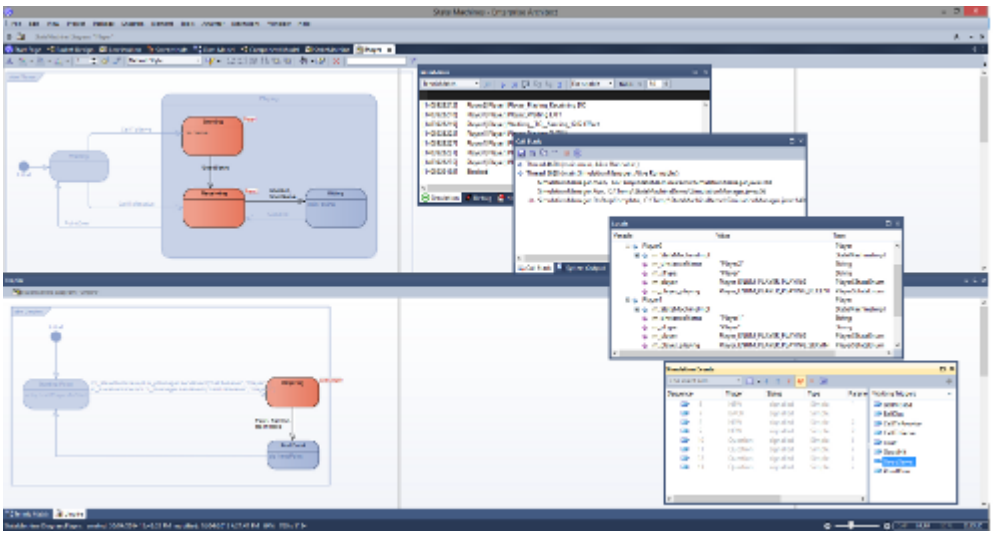
- Extensive support for Modelica; you can create and configure your model in Enterprise Architect and execute it in Modelica
- Presentation of the results of your modeling and simulation in Chart formats, either within a dedicated graphics presentation tool or through the Dynamic Charting facilities of Enterprise Architect



Executable State Machines

Executable StateMachines provide a means of rapidly generating, executing and simulating complex state models. In contrast to dynamic simulation of State Charts using Enterprise Architect's Simulation engine, Executable StateMachines provide a complete language-specific implementation that can form the behavioral 'engine' for

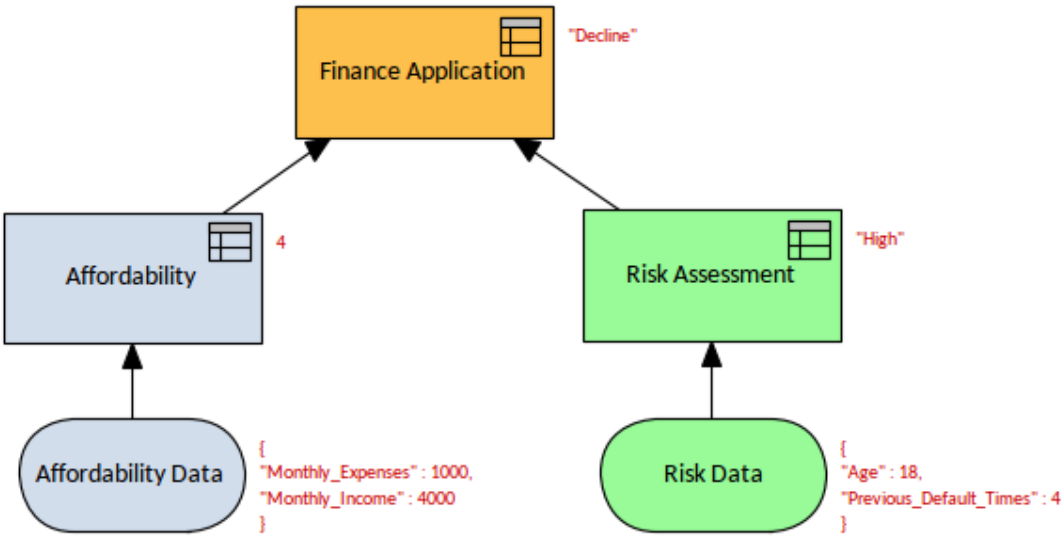
multiple software products on multiple platforms. Visualization of the execution is based on a seamless integration with the Simulation capability. Evolution of the model now presents fewer coding challenges. The code generation, compilation and execution is taken care of by Enterprise Architect. For those having particular requirements, each language is provided with a set of code templates. Templates can be customized by you to tailor the generated code in any ways you see fit.



Decision Model Simulation

Organizations face increasingly difficult operating environments with fierce and often unpredictable competition from existing and new market players, changes in government and industry regulations, and upheavals in the social fabric of their customer base. An organization's decisions in this context are critical to its success and its ability to steer a safe path through these uncharted corporate

waters. Using Enterprise Architects Decision Model and Notation (DMN) features, you can not only model your organization's decisions, but you can also run simulations from these models to predict outcomes based on example data sets. The power of the language is that business people can readily understand and work with expressive but straightforward Decision Requirements diagrams that detail the decisions, including their inputs and the expected outputs. A modeler can document the rules in several ways, including easy to define decision tables. Once completed, these diagrams with accompanying input data examples, can be simulated to show the results of the decisions.



Publications and Documentation

A systems engineer, manager, or other stakeholders can use the documentation features to automatically generate a wide range of documentation directly from the models. These can be document based such as PDF and Docx format or HTML-based. You can use flexible templates to completely tailor the generated documents, including company logos, tables of content, tables of element information, and diagrams. The success of a systems engineering project and, ultimately the entire engineering practice will depend on how well you communicate with stakeholders. Many stakeholders will be content to view systems models, including lists, diagrams, and matrices, directly in the repository. Still, others will want or require by contract, electronic, or printed documentation delivered to them. You can use the documentation generator to create high-quality corporate publications automatically from the repository. This includes a wide range of standard publications such as the Operation Concept document, Functional Architecture, Requirements Specification, and more. Modelers can also create Ad-hoc reports from tools such as the Glossary and the Search Window.


Your
LOGO
Concept of Operation
Smart Boom Gate Project
Version 1.0
Sponsored by: Innovation and Technology Hub

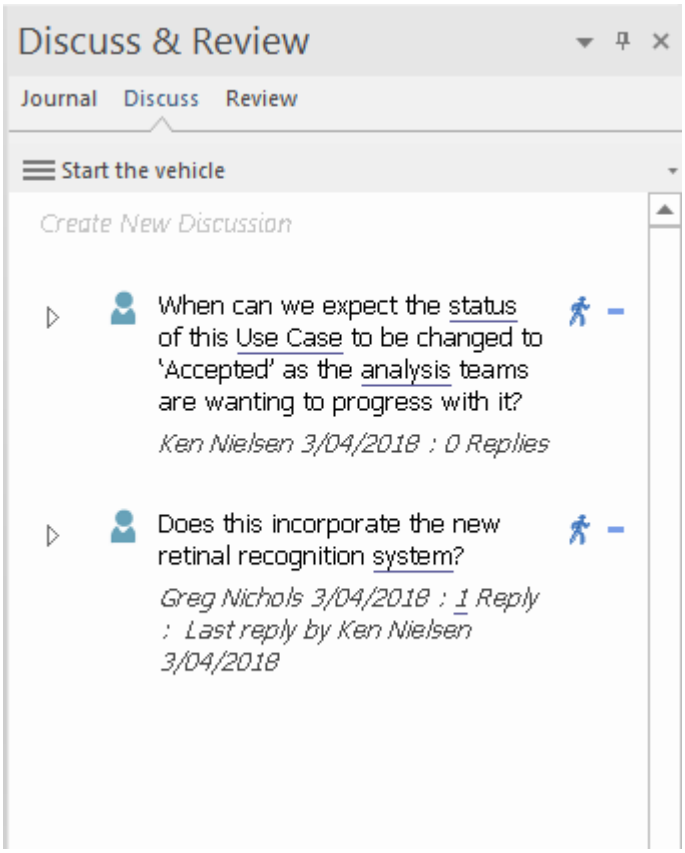
Collaboration and Teams

An engineering team is multidisciplinary and consists of strategists, managers, system engineers, software engineers, testers and others. The commercial pressures to release a product or provide a solution means that teams have to work more cleverly and cohesively to ensure engineering outcomes. Enterprise Architect has been built from the ground up as a collaborative platform, not just for engineers but for all disciplines. It facilitates individuals and teams working together and sharing information, models, designs, and solutions with a full range of tools from discussions, reviews, a team library, and chat to Version Control and Baselines.

Discussions and Chat

Central to the notion of collaboration is a modeler's ability to discuss and chat with colleagues or industry and standards specialists about a problem or solution. Enterprise Architect allows engineers, managers, and others to discuss elements, diagrams, and connectors. Any modeler can create a post to start a thread or conversation that other modelers can enter into by replying. The discussions are kept separately from element and diagram meta-information, allowing you to make rich and constructive comments without affecting documentation or reports generated from the models. The discussions and chat are two options

available, discussions from the Discuss & Review window and chats from the Chat & Mail window.



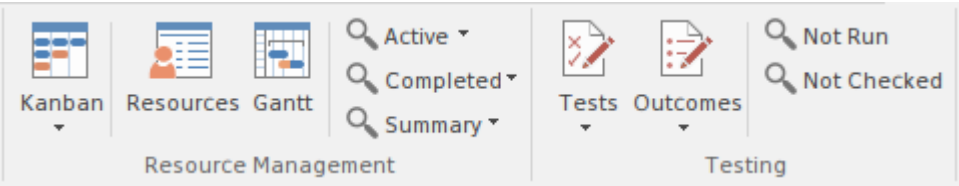
Chat is useful for quick and responsive communication with colleagues or experts that have been defined as part of a security defined group of users. Chats are not related to model elements in the way that discussions are but rather are global and when the Chat & Mail window is opened and a group is selected, the items are listed in date-time order. For more information see the Collaboration Panel Help topic.

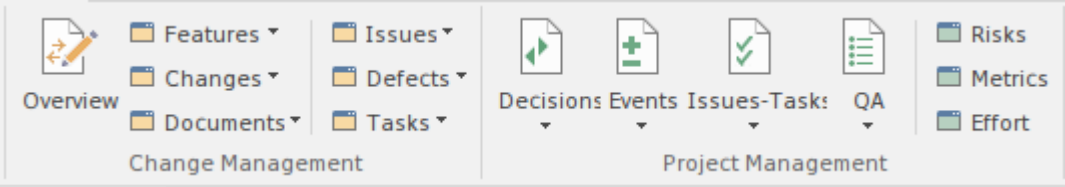
Project Management

Enterprise Architect has been built from the ground up with the Project Manager in mind. Systems repositories are valuable engineering assets and should be managed and maintained accordingly. Any team can assign themselves as a resource applied to an item performing particular roles, and you can conveniently view these in a built-in Gantt Chart. Risk can be modeled and managed in various locations, and engineers can determine project effort with built-in support for Metrics and Estimation. An Audit function allows changes to be tracked at a fine-grained level, and a Library facility and Element Reviews and Discussions enable users to work collaboratively on models.

The construct ribbon provides a number of panels that contain project, resource and test management features including element maintenance items. The engineer can enter project management items at two levels:

- Project level items - these apply at the level of the project.
- Element level items - these apply to specific elements.

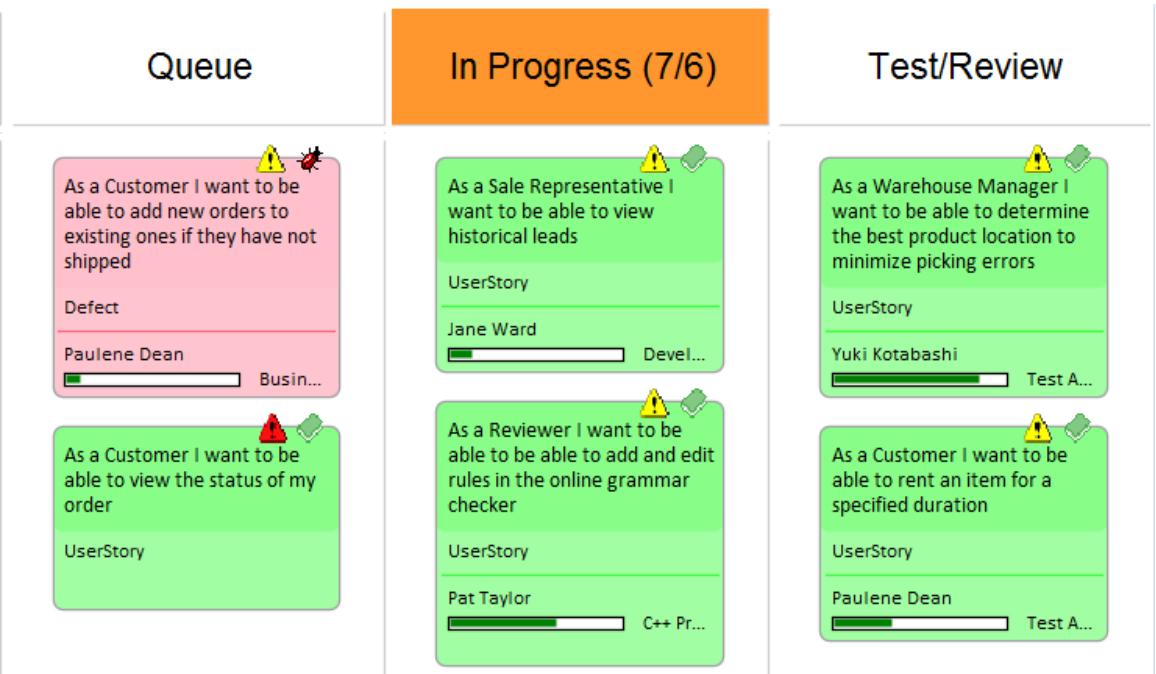




The tool has a range of Project Management features for agile and iterative development as well as more traditional methods including the following.

Kanban Diagrams

Engineers can utilize Kanban diagrams to manage parts or all of an engineering project more agile and flexible. You can assign resources to various elements from use cases to deployment packages, and progress bars show each resource's percentage completion for each item.

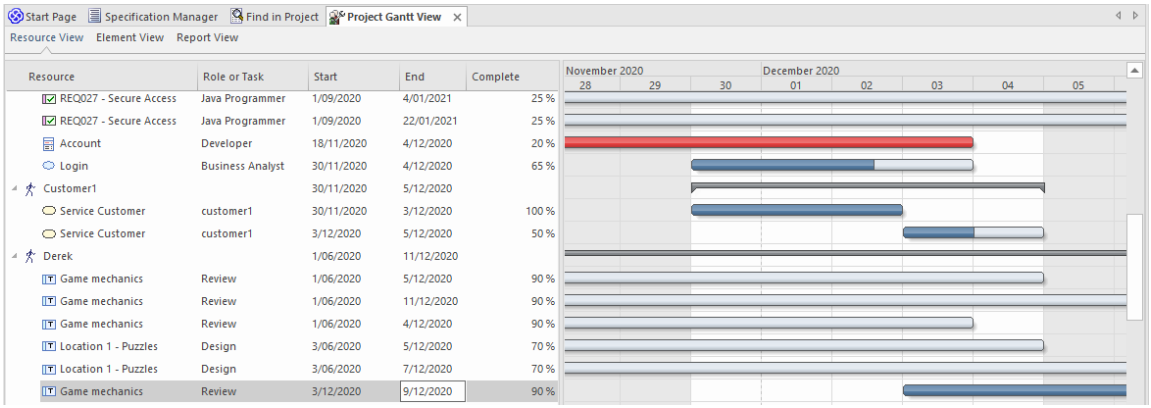


Resource Allocation and Gantt Charts

An engineering team comprises many different human resources, typically performing one or more roles in the engineering project. In developing a model, these resources create or maintain model structures and elements that describe or prescribe how the project will be run and the system built. As a Project Manager or engineer, you can assign resources to tasks on elements (including Packages) in the model, planning and monitoring their work within the timeframe you have allocated for that work to be completed. The allocation of resources can be carried out using the Project Gantt View, Kanban or Construct facilities.

The Gantt View is a tool that allows the engineer or project manager to visualize the elements in a project, Package, or diagram and the resources that have been allocated to them. There are several different Gantt charts available:

- The Project Gantt View used to view elements across the entire repository.
- Diagram Gantt View, which you use to display the allocation of resources to the elements in a given diagram.
- Package Gantt View, which you use to display the allocation of resources to the elements contained in a selected Package.



The tool will empower the traditional or agile project manager to ensure that a project's resources are allocated to repository content and help ensure high-value outcomes are achieved right from within the repository.

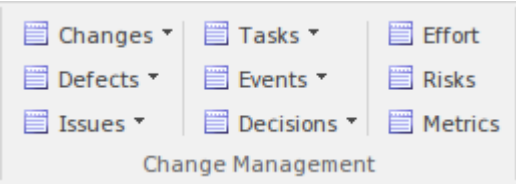
The Gantt View's primary use is to display the allocation of resources to elements in the repository and to manage the work breakdown structure. Youy can apply a wide range of views and filters to tailor the view or make it more relevant to a particular audience. Allocations can be made to any elements in the repository, from high-level packages to an individual element such as a Class, Activity, or Change. It is a powerful tool for project managers to visualize how a team utilization and ultimately deliver high value and high priority outcomes. Modelers working on a project can view their own work and update their progress on assigned tasks.

While an engineer can make broad changes using the visual duration bars in the Gantt View, it is common practice to use the tool in conjunction with the Resource Allocation window where fine details can be entered and adjusted.

Change Management

Change is inevitable, and the best-run engineering projects predict that change will occur and plan for it. As an engineer, project, or product manager, you can use Enterprise Architect to register a range of maintenance items against any element in the project, including requirements, blocks, activities, use cases, and packages. The

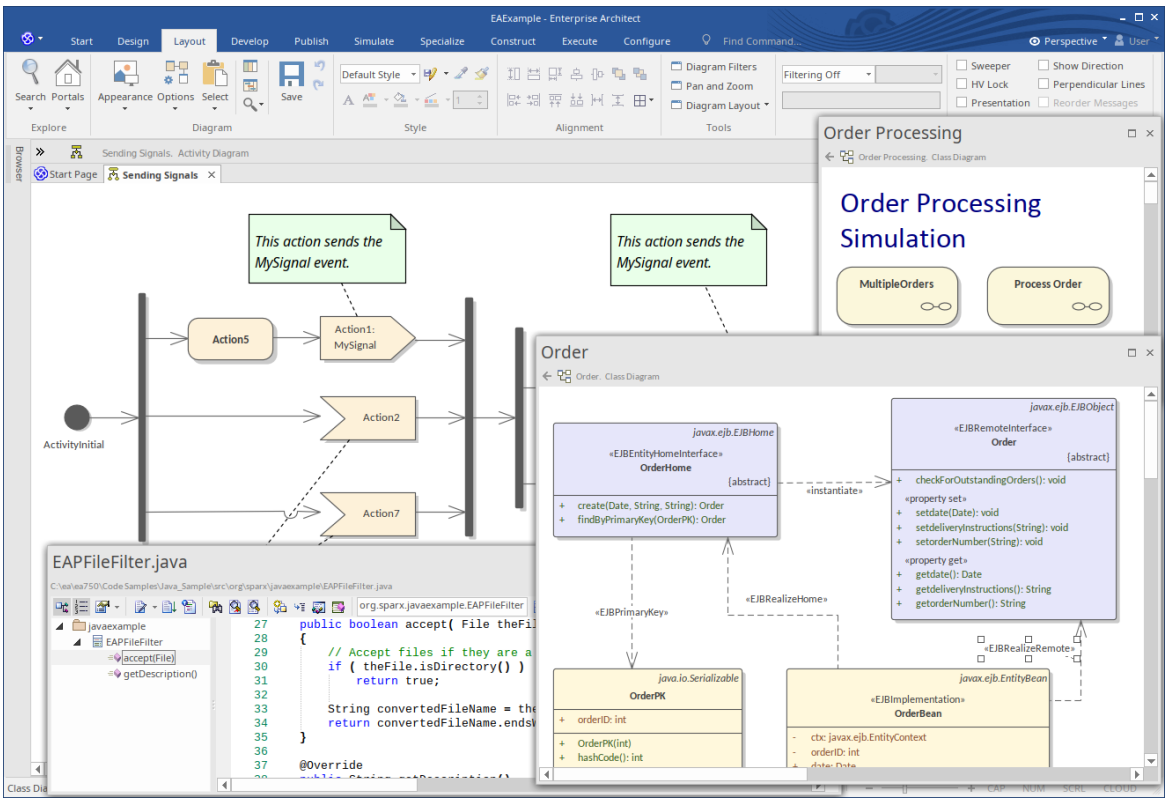
Maintenance items are defects, changes, issues, tasks, features, and documents that apply at the model element level. They are properties of individual model elements that the engineer can use to record and capture problems, changes, issues, and tasks as they arise and document the solution and associated details.



Software Engineering

Create and Manage Effective and Productive Structural and Behavioral Models of Software

Software engineering is the discipline of designing, implementing and maintaining software. The process of software engineering starts with requirements and constraints as inputs, and results in programming code and schemas that are deployed to a variety of platforms, creating running systems.



Enterprise Architect has a rich set of tools and features that assist Software Engineers to perform their work efficiently and reduce the number of errors in implemented solutions.

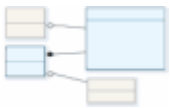
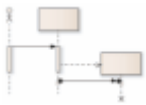
The features include design tools to create models of software, automated code generation, reverse engineering of source code, binaries and schemas, and tools to synchronize source code with the design models. The programming code can be viewed and edited directly in the integrated Code Editors within Enterprise Architect, which provide Intelli-sense and other features to aid in coding.

Another compelling aspect of the environment is the ability to trace the implementation Classes back to design elements and architecture, and then back to the requirements and constraints and other specifications, and ultimately back to stakeholders and their goals and visions.

Enterprise Architect supports a wide range of programming languages and platforms and provides a lightweight and seamless integration with the two most prevalent Integrated Development Environments: Visual Studio and Eclipse. In addition there is a fully featured Execution Analyzer that allows the Software Engineer to design, build debug and test software modules right inside Enterprise Architect.

Facilities

Facility	Description
Development Tools 	Discover the tightly Integrated Development Environment with outstanding tools and functionality.

Code, Build and Debug 	Model, develop, debug, profile and manage an application from within the modeling environment.
Visual Analysis of Executing Code 	Understand your code base by visually analyzing running code. Use Test Points, profiling and automated diagram generation.
Generate Source Code 	Explore some of the ways to generate source code for a single Class, a selection of Classes, or a whole Package. Generate from structural or behavioral models.
Importing Source Code 	Examine existing systems by importing source code into Enterprise Architect. View and modify dialog definitions. Synchronize the model with the latest updates to source code.

Getting Started

Configuration Settings

Selecting the Perspective

Enterprise Architect partitions the tool's extensive features into Perspectives, which ensures that you can focus on a specific task and work with the tools you need without the distraction of other features. To work with Software Model features you first need to select one of these Perspectives:

The Software Engineering Set:


<perspective name> > Software Engineering > Code Engineering


<perspective name> > Software Engineering > GoF Patterns


<perspective name> > Software Engineering > ICONIX

The UX Design Set:


<perspective name> > UX Design > Win 32 UI Models

Setting the Perspective ensures that the Case Management Model and Notation diagrams, their tool boxes and other

features of the Perspective will be available by defaulᵗ.

Example Diagram

An example diagram provides a visual introduction to the topic and allows you to see some of the important elements and connectors that you use to specify or describe classes for the visualization of software and the forward and reverse engineering to and from a wide range of programming languages.

Integrated Development

In this topic you will learn how to use the fully featured integrated development environment. You will learn how to create structural and behavioral models of software artifacts in a rich code editor, generate and reverse engineer code, customize the way code is generated, run analyzer scripts to optimize code, use the debugger and set units test and much more.

Behavioral Models

Behavioral Models

In this topic you will learn how to generate code for software, system and hardware description languages directly from behavioral diagrams including: StateMachine, Sequence and Activity Diagrams. This will add new dimensions and precisions to the way you work with software and engineering systems.

Gang of Four (GoF) Patterns

This topic introduces the renowned twenty-three design patterns collected together as the Gang of Four (GoF) patterns which refers to their four authors. You will have at hand the solutions to common problems facing software engineers and be able to inject these patterns into your own models adding to the quality and rigor to your software systems.

Win32 User Interface Dialogs

In this topic you will learn how to work with Enterprise Architect's User Interface modeling capability that allows you to model user interface screens using Win32® controls. The models can be forward or reverse engineered and can

also provide an interface for StateMachine and Activity diagram simulation, allowing them to receive and process user input.

Code Template Framework

In this topic you will learn how to work with the Code Template Framework which governs how models and converted to code. There are a standard set of templates but you can extended these to create your own templates and to generate code to suit your needs. There are also templates that control transformations and the generation of Database Definition Language (DDL.

Grammar Framework

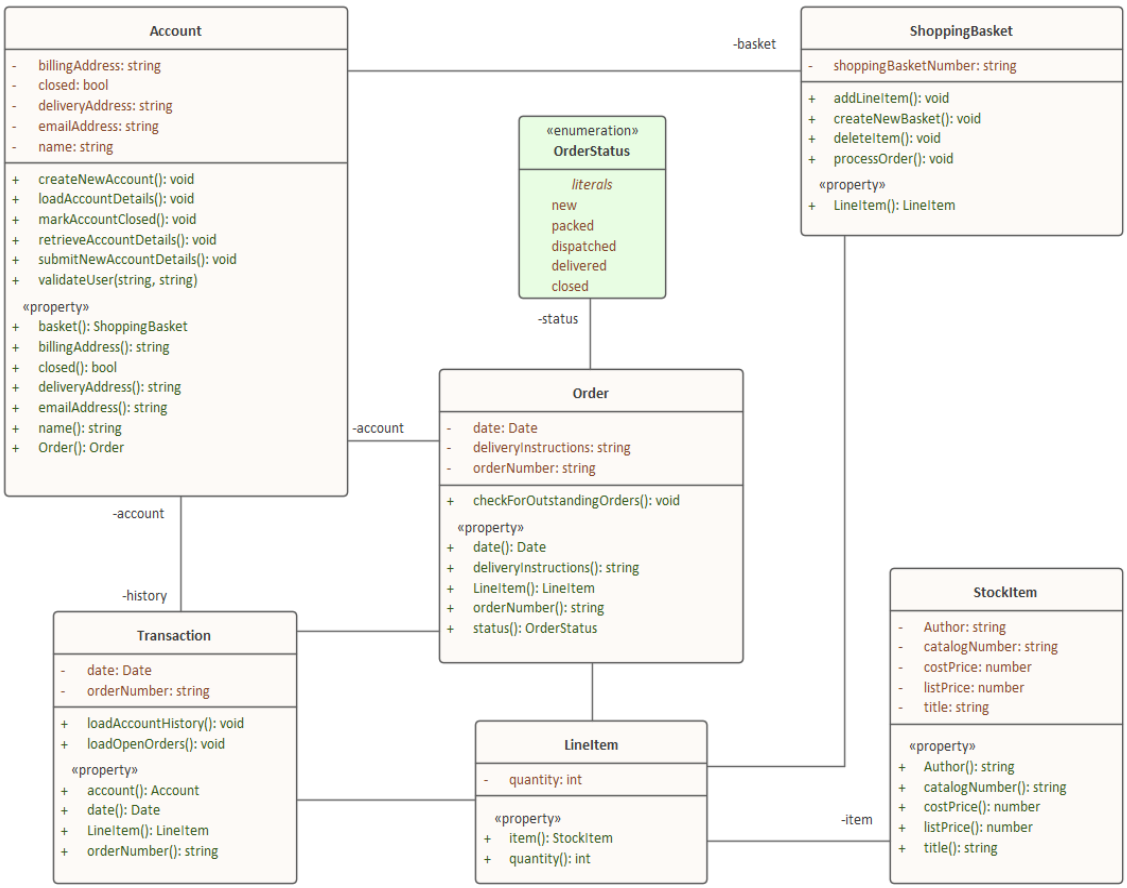
In this topic you will learn how to create a grammar to convert an unsupported programming language into a UML model. Enterprise Architect has built in support for a wide range of programming languages but if you need to work with an unsupported language you can use the Grammar Framework to write your own parser. The grammar is used to reverse engineer programming code in the form of text and is the direct compliment of the Code Template Framework which you would you to specify how a UML model for an unsupported language is converted to code.

Code Miner Framework

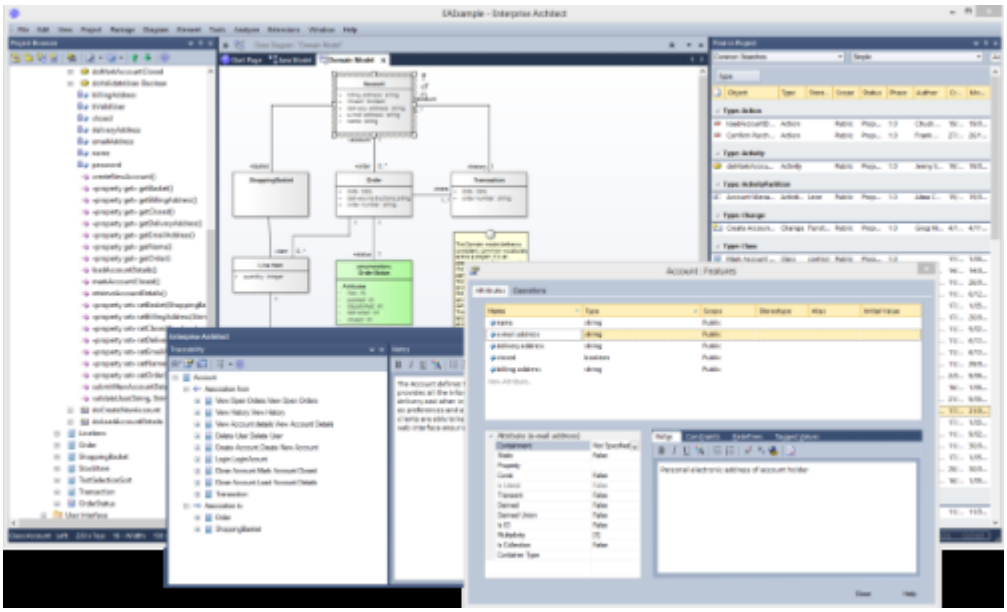
In this topic you will learn how to work with a database of source code which provides access to the data hidden within source code in a timely and effective manner. Source code is parsed creating a tree structure which can be used to analyze program structure, calculate metrics, trace relationships and even perform refactoring.

Example Diagram

Software diagrams allow you to model the structure and behavior of software including User Interfaces. Enterprise Architect has at its core fundamental support for modelling software and the tool supports a wide range of programming languages and paradigms. In the following diagram we see classes used to model an online shop including Classes that contain compartments for Attributes, Operations and Properties. An Enumeration has also been used to model Order Status.



Integrated Development



Enterprise Architect provides an unmatched set of tools and features for the Software Engineer, to assist in the process of creating robust and error free software systems. The engineer can start by defining the architecture and ensuring that it traces back to the requirements and specification. Technology neutral models can be transformed to target a comprehensive range of programming languages. The Model Driven Development Environment fits the bill for various technologies.

Features

Development Tools	• Model driven development with best-in-class UML tools • Generate and reverse engineer code

- Customize code generation with templates
- Analyzer Scripts to manage your applications
- Code editors to author the code base
- Debuggers to investigate behavior
- Profilers to visualize behavior
- Analyzers to record behavior
- Testpoints for validation of programming contracts
- Integration with jUnit and nUnit
- Eclipse or Visual Studio Integration where required

Traceability At a glance traceability of Generalizations, Realizations, Associations, Dependencies and more. Customize relationship views. Easily navigate related elements in the model.

Usage Quickly browse element usage across all diagrams. Perform effective element searches using sophisticated queries.

Popular Languages

- C/ C++
- Java
- Microsoft .NET family
- ADA

- Python
- Perl
- PHP

Toolboxes Toolboxes are provided for a vast array of modeling technologies and programming languages.

Application Patterns Enterprise Architect provides complete starter projects, including model information, code and build scripts, for several basic application types.

Feature Overview

Code Engineering with Enterprise Architect broadly encompasses various processes for the design, generation and transformation of code from your UML model.

Features

Model Driven Code Engineering	• Source code generation and reverse engineering for many popular languages, including C++, C#, Java, Delphi, VB.Net, Visual Basic, ActionScript, Python and PHP • A built in 'syntax highlighting' source code editor • Code generation templates, which enable you to customize the generated source code to your company specifications
Transformations for Rapid Development	• Advanced Model Driven Architecture (MDA) transformations using transformation templates • Built-in transformations for DDL, C#, Java, EJB and XSD • One Platform Independent Model can

	be used to generate and synchronize multiple Platform Specific Models, providing a significant productivity boost • XSL Transform diagram, toolbox, editor and debugger.
Visual Execution Analysis / Debugging, Verification and Visualization	• Execute build, test, debug, run and deploy scripts • Integrate UML development and modeling with source development and compilation • Generate NUnit and JUnit test Classes from source Classes using MDA Transformations • Integrate the test process directly into the Enterprise Architect IDE • Debug .NET, Mono, Java and Microsoft Native (C, C++ and Visual Basic) applications • Design and execute Test suites based on Programming by Contract principles • XSL Stylesheet debugging
Database Modeling	Enterprise Architect enables you to: • Reverse engineer from many popular DBMSs, including SQL Server, My SQL, Access, PostgreSQL and Oracle

- Model database tables, columns, keys, foreign keys and complex relationships using UML and an inbuilt data modeling profile
- Forward generate DDL scripts to create target database structures

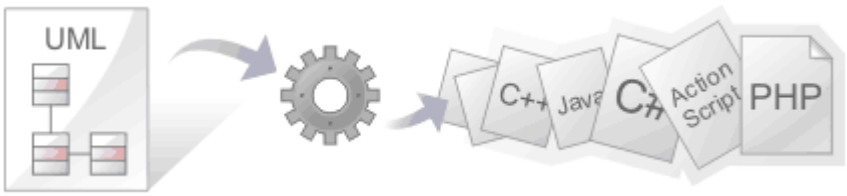
XML Technology Engineering

Enterprise Architect enables you to rapidly model, forward engineer and reverse engineer two key W3C XML technologies:

- XML Schema (XSD)
- Web Service Definition Language (WSDL)

XSD and WSDL support is critical for the development of a complete Service Oriented Architecture (SOA), and the coupling of UML 2.5 and XML provides the natural mechanism for implementing XML-based SOA artifacts within an organization.

Generate Source Code



Source code generation is the process of creating programming code from a UML model. There are great benefits in taking this approach as the source code Packages, Classes and Interfaces are automatically created and elaborated with variables and methods.

Enterprise Architect can also generate code from a number of behavioral models, including StateMachine, Sequence and Activity diagrams. There is a highly flexible template mechanism that allows the engineer to completely tailor the way that source code is generated, including the comment headers in methods and the Collection Classes that are used.

From an engineering and quality perspective, the most compelling advantage of this approach is that the UML models and therefore the architecture and design are synchronized with the programming code. An unbroken traceable path can be created from the goals, business drivers and the stakeholder's requirements right through to methods in the programming code.

Facilities

Facility	Description

Languages	Enterprise Architect supports code generation in each of these software languages: • Action Script • Ada • ArcGIS • C • C# (for .NET 1.1, .NET 2.0 and .NET 4.0) • C++ (standard, plus .NET managed C++ extensions) • Delphi • Java (including Java 1.5, Aspects and Generics) • JavaScript • MFQL • MySql • PHP • Python • Teradata SQL • Visual Basic • Visual Basic .NET • WorkFlowScript You can also generate Hardware Definition Language code in these languages:

	• VHDL • Verilog • SystemC
Elements	Code is generated from Class or Interface model elements, so you must create the required Class and Interface elements to generate from. All other types of element to contribute to the code (such as StateMachines or Activities) must be child elements of a Class. Add attributes (which become variables) and operations (which become methods). Constraints and Receptions are also supported in the code.
Settings	Before you generate code, you should ensure the default settings for code generation match your requirements; set up the defaults to match your required language and preferences. Preferences that you can define include default constructors and destructors, methods for interfaces and the Unicode options for created languages. Languages such as Java support 'namespaces' and can be configured to specify a namespace root. In addition to the default settings for

	generating code, Enterprise Architect facilitates setting specific generation options for each of the supported languages.
Code Template Framework	The Code Template Framework (CTF) enables you to customize the way Enterprise Architect generates source code and also enables generation of languages that are not specifically supported by Enterprise Architect.
Local Paths	Local path names enable you to substitute tags for directory names.
Behavioral Code	You can also generate software code from three UML behavioral modeling paradigms: • Interaction (Sequence) diagrams • Activity diagrams • StateMachine diagrams (using Legacy StateMachine Templates in the code generation operations under 'Tasks') • StateMachine diagrams (using an Executable StateMachine Artifact)
Live Code Generation	On the 'Develop > Source Code > Options' drop-down menu, you have the option to update your source code

	instantly as you make changes to your model.
Tasks	When you generate code, you perform one or more of these tasks: • Generate a Single Class • Generate a Group of Classes • Generate a Package • Update Package Contents

Notes

- Most of the tools provided by Enterprise Architect for code engineering and debugging are available in the Professional and higher editions of Enterprise Architect; Behavioral Code Generation is available in the Unified and Ultimate Editions
- When security is enabled you require the access permissions 'Generate Source Code and DDL' and 'Reverse Engineer from DDL and Source Code'

Generate a Single Class

Before you generate code for a single Class, you:

- Complete the design of the model element (Class or Interface)
- Create Inheritance connectors to parents and Associations to other Classes that are used
- Create Inheritance connectors to Interfaces that your Class implements; the system provides an option to generate function stubs for all interface methods that a Class implements

Generate code for a single Class

Step	Action
1	Open the diagram containing the Class or Interface for which to generate code.
2	Click on the required Class or Interface and select the 'Develop > Source Code > Generate > Generate Single Element' ribbon option, or press F11. The 'Generate Code' dialog displays, through which you can control how and where your source code is generated.

3	In the 'Path' field, click on the ... button and select a path name for your source code to be generated to.
4	In the 'Target Language' field, click on the drop-down arrow and select the language to generate; this becomes the permanent option for that Class, so change it back if you are only doing one pass in another language.
5	Click on the Advanced button. The 'Object Options' dialog displays, providing subsets of the 'Source Code Engineering' and code language options pages on the 'Preferences' dialog.
6	Set any custom options (for this Class alone), then click on the Close button to return to the 'Generate Code' dialog.
7	In the 'Import(s) / Header(s)' fields, type any import statements, #includes or other header information. Note that in the case of Visual Basic this information is ignored; in the case of Java the two import text boxes are merged; and in the case of C++ the first import text area is placed in the header file and the second in the body (.cpp) file.
8	Click on the Generate button to create the source code.

9	When complete, click on the View button to see what has been generated. Note that you should set up your default viewer/editor for each language type first; you can also set up the default editor on the 'Code Editors' page of the Preferences window ('Start > Application > Preferences > Preferences > Source Code Engineering > Code Editors').

Generate a Group of Classes

In addition to being able to generate code for an individual Class, you can also select a group of Classes for batch code generation. When you do this, you accept all the default code generation options for each Class in the set.

Generate Class Group

Step	Detail
1	Select a group of Classes and/or interfaces in a diagram.
2	Click on an element in the group and select the 'Develop > Source Code > Generate > Generate Selected Element(s)' ribbon option (or press Shift+F11). If no code exists for the selected elements, the 'Save As' dialog displays on which you specify the file path and name for each code file; enter this information and click on the Save button.
3	The 'Batch Generation' dialog displays, showing the status of the process as it executes (the process might be too fast to see this dialog).

	If code already exists for the selected Class elements, and changes have been made to the Class name or structure, the 'Synchronize Element <package name>.<element name>' dialog might also display; this dialog helps synchronize the model and code.

Notes

- If any of the elements selected are not Classes or interfaces the option to generate code is not available

Generate a Package

In addition to generating source code from single Classes and groups of Classes, you can generate code from a Package. This feature provides options to recursively generate code from child Packages and automatically generate directory structures based on the Package hierarchy. This helps you to generate code for a whole branch of your project model in one step.

Access

Ribbon	Develop > Source Code > Generate > Generate All
Keyboard Shortcuts	Ctrl+Alt+K

Generate code from a Package, on the Generate Package Source Code dialog

Step	Action

1	In the 'Synchronize' field, click on the drop-down arrow and select the appropriate synchronize option: • 'Synchronize model and code': Code for Classes with existing files is forward synchronized with that file; code for Classes with no existing file is generated to the displayed target file • 'Overwrite code': All selected target files are overwritten (forward generated) • 'Do not generate': Generate code for only those selected Classes that do not have an existing file; all other Classes are ignored
2	Highlight the Classes for which to generate code; leave unselected any to not generate code for. If you want to display more of the information within the layout, you can resize the dialog and its columns.
3	To make Enterprise Architect automatically generate directories and filenames based on the Package hierarchy, select the 'Auto Generate Files' checkbox; this enables the 'Root Directory' field, in which you select a root directory under which the source directories are to be generated. By default, the 'Auto Generate Files' feature ignores any file paths that are already associated with a Class; you can change this behavior by also selecting the 'Retain Existing File Paths' checkbox.

4	To include code for all sub-Packages in the output, select the 'Include Child Packages' checkbox.
5	Click on the Generate button to start generating code. As code generation proceeds, Enterprise Architect displays progress messages. If a Class requires an output filename the system prompts you to enter one at the appropriate time (assuming Auto Generate Files is not selected). For example, if the selected Classes include partial Classes, a prompt displays to enter the filename into which to generate code for the second partial Class.

Further information on the dialog options

Option	Action
Root Package	Check the name of the Package for which code is to be generated.
Synchronize	Select options that specify how existing files should be regenerated.
Auto	Specify whether Enterprise Architect

Generate Files	should automatically generate file names and directories, based on the Package hierarchy.
Root Directory	If Auto Generate Files is selected, display the path under which the generated directory structures are created.
Retain Existing File Paths	If Auto Generate Files is selected, specify whether to use existing file paths associated with Classes. If Auto Generate Files is unselected, Enterprise Architect generates Class code to automatically determined paths, regardless of whether source files are already associated with the Classes.
Include all Child Packages	Also generate code for all Classes in all sub-Packages of the target Package in the list. This option facilitates recursive generation of code for a given Package and its sub-Packages.
Select Objects to Generate	List all Classes that are available for code generation under the target Packages; only code for selected (highlighted) Classes is generated. Classes are listed with their target source

	file.
Select All	Mark all Classes in the list as selected.
Select None	Mark all Classes in the list as unselected.
Generate	Start the generation of code for all selected Classes.
Cancel	Exit the 'Generate Package Source Code' dialog; no Class code is generated.

Update Package Contents

In addition to generating and importing code, Enterprise Architect provides the option to synchronize the model and source code, creating a model that represents the latest changes in the source code and vice versa. You can use either the model as the source, or the code as the source.

The behavior and actions of synchronization depend on the settings you have selected on the 'Attributes and Operations' page of the 'Preferences' dialog. Working with these settings, you can either protect or automatically discard information in the model that is not present in the code, and prompt for a decision on code features that are not in the model. In these two examples, the appropriate checkboxes have been selected for maximum protection of data:

- You generated some source code, but made subsequent changes to the model; when you generate code again, Enterprise Architect adds any new attributes or methods to the existing source code, leaving intact what already exists, which means developers can work on the source code and then generate additional methods as required from the model, without having their code overwritten or destroyed
- You might have made changes to a source code file, but the model has detailed notes and characteristics you do not want to lose; by synchronizing from the source code into the model, you import additional attributes and methods but do not change other model elements

Using the synchronization methods, it is simple to keep source code and model elements up to date and synchronized.

Access

Ribbon	Develop > Source Code > Synchronize > Synchronize Package

Synchronize Package contents against source code

Field/Button	Action
Update Type	Select the radio button to either Forward Engineer or Reverse Engineer the Package Classes, as appropriate.
Include child packages in generation	Select the checkbox to include child Packages in the synchronization.
OK	Click on the button to start synchronization.

	Enterprise Architect uses the directory names specified when the project source was first imported/generated and updates either the model or the source code depending on the option chosen. If: • Performing forward synchronization AND • There are differences between the model and code AND • The 'On forward synch, prompt to delete code features not in model' checkbox is selected in the 'Options - Attributes and Operations' dialog THEN the 'Synchronize Element <package name>.<element name>' dialog displays. Otherwise, no further action is required.

Notes

- Code synchronization does not change method bodies; behavioral code cannot be synchronized, and code generation only works when generating the entire file
- In the Corporate, Unified and Ultimate Editions of Enterprise Architect, if security is enabled you must have 'Generate Source Code and DDL' permission to

synchronize source code with model elements

Synchronize Model and Code

You might either:

- Synchronize the code for a Package of Classes against the model in the Browser window, or
- Regenerate code from a batch of Classes in the model

In such processes, there might be items in the code that are not present in the model.

If you want to trap those items and resolve them manually, select the 'On forward synch, prompt to delete code features not in model' checkbox in the 'Options - Attributes and Operations' dialog, so that the 'Synchronize Element <package name>.<element name>' dialog displays, providing options to respond to each item.

Synchronize Items

Button	Detail
Select All	Highlight and select all items in the Feature column.
Clear All	Deselect and remove highlighting from all items in the Feature column.
Delete	Mark the selected code features to be

	removed from the code (the value in the Action column changes to Delete).
Reassign	Mark the selected code features to be reassigned to elements in the model. This is only possible when an appropriate model element is present that is not already defined in the code. The Select the Corresponding Class Feature dialog displays, from which you select the Class to reassign the feature to. Click on the OK button to mark the feature for reassignment.
Ignore	Mark the selected code elements not present in the model to be ignored completely (the default; the value in the Action column remains as or changes to <none>).
Reset to Default	Reset the selected items to Ignore (the value in the Action column changes to <none>).
OK	Make the assigned changes to the items, and close the dialog.

Namespaces

Languages such as Java support Package structures or namespaces. In Enterprise Architect you can specify a Package as a namespace root, which denotes where the namespace structure for your Class model starts; all subordinate Packages below a namespace root will form the namespace hierarchy for contained Classes and Interfaces.

To define a Package as a namespace root, click on the Package in the Browser window and select the 'Develop > Source Code > Options > Set as Namespace Root' ribbon option. The Package icon in the Browser window changes to show a colored corner indicating this Package is a namespace root.

Generated Java source code, for example, will automatically add a Package declaration at the beginning of the generated file, indicating the location of the Class in the Package hierarchy below the namespace root.

To clear an existing namespace root, click on the namespace root Package in the Browser window and deselect the 'Develop > Source Code > Options > Set as Namespace Root' ribbon option

To view a list of namespaces, select the 'Settings > Reference Data > Settings > Namespace Roots' ribbon option; the 'Namespaces' dialog displays. If you double-click on a namespace in the list, the Package is highlighted in the Browser window; alternatively,

right-click on the namespace and select the 'Locate Package in Browser' option.

You can also clear the selected namespace root by selecting the 'Clear Namespace Attribute' option.

To omit a subordinate Package from a namespace definition, select the 'Develop > Source Code > Options > Suppress Namespace' ribbon option; to include the Package in the namespace again, deselect the ribbon option.

Notes

- When performing code generation, any Package name that contains whitespace characters is automatically treated as a namespace root

Importing Source Code

The ability to view programming code and the models it is derived from at the same time brings clarity to the design of a system. One of Enterprise Architect's convenient code engineering features is the ability to Reverse Engineer source code into a UML model. A wide range of programming languages are supported and there are options that govern how the models are generated. Once the code is in the model it is possible to keep it synchronized with the model regardless of whether the changes were made directly in the code or the model itself. The code structures are mapped into their UML representations; for example, a Java class is mapped into a UML Class element, variables are defined as attributes, methods modeled as operations, and interactions between the Java classes represented by the appropriate connectors.

The representation of the programming code as model constructs helps you to gain a better understanding of the structure of the code and how it implements the design, architecture and the requirements, and ultimately how it delivers the business value.

It is important to note that if a system is not well designed, simply importing the source into Enterprise Architect does not turn it into an easily understandable UML model. When

working with a poorly designed system it is useful to assess the code in manageable units by examining the individual model Packages or elements generated from the code; for example, dragging a specific Class of interest onto a diagram and then using the 'Insert Related Elements' option at one level to determine the immediate relationships between that Class and other Classes. From this point it is possible to create Use Cases that identify the interaction between the source code Classes, providing an overview of the application's operation.

Several options guide how the code is reversed engineered, including whether comments are imported to notes and how they are formatted, how property methods are recognized and whether Dependency relationships are created for operation return and parameter types.

Copyright Ownership

Situations that typically lend themselves to reverse engineering tend to operate on source code that:

- You have already developed
- Is part of a third-party library that you have obtained permission to use
- Is part of a framework that your organization uses
- Is being developed on a daily basis by your developers

If you are examining code that you or your organization do not own or do not have specific permission to copy and edit, you must ensure that you understand and comply with the

copyright restrictions on that code before beginning the process of reverse engineering.

Supported languages for Reverse Engineering

Language
Action Script
Ada 2012 (Unified and Ultimate Editions)
C
C#
C++
CORBA IDL (MDG Technology)
Delphi
Java
PHP

Python
SystemC (Unified and Ultimate Editions)
Verilog (Unified and Ultimate Editions)
VHDL (Unified and Ultimate Editions)
Visual Basic
Visual Basic .NET

Notes

- Reverse Engineering is supported in the Professional, Corporate, Unified and Ultimate Editions of Enterprise Architect
- If security is enabled you must have 'Reverse Engineer From DDL And Source Code' permission to reverse engineer source code and synchronize model elements against code
- Using Enterprise Architect, you can also import certain types of binary file, such as Java .jar files and .NET PE files
- Reverse Engineering of other languages is currently available through the use of MDG Technologies listed on

the MDG Technology pages of the Sparx Systems website

Import Projects

Enterprise Architect provides support for importing software projects authored in Visual Studio, Mono, Eclipse and NetBeans. Importing and working on projects in Enterprise Architect has multiple benefits, not least the immediate access to Enterprise Architect's renowned modeling tools and management features, but also the access to development tools such as simulation, debugging and profiling.

Access

Ribbon	Develop > Source Code > Solutions > Import a <project type>

Import a Visual Studio Solution

This option allows you to import one or more projects from an existing Visual Studio Solution file or a running instance of Visual Studio. The wizard will generate a Class model for each of the projects and the appropriate Analyzer Scripts for each Visual Studio configuration.

Import a Mono Solution

This option allows you to import Mono projects from a solution file. The dialog that is presented is the same as the 'Visual Studio Import' dialog, but you can choose to target either Linux or Windows. The wizard will generate a Class model for each of the projects and configure them for debugging. The generated Analyzer Scripts reference msbuild to build the projects.

Import an Eclipse Project

The Eclipse 'Wizard' can reverse engineer a Java project described by its Eclipse .project file and ANT build. The feature will result in a UML Class model and Analyzer Scripts for each of the ANT targets you select. The process will also generate a script for each debug protocol you select through the 'Wizard'. You will be presented with the choice of JDWP (Java Debug Wire Protocol), good for servers, and JVMTI (Java Virtual Machine Tools Interface), which is suited to standalone Java applications. These scripts should be used for debugging the project in Enterprise Architect.

Import a NetBeans Project

The NetBeans 'Wizard' can reverse engineer a Java project described by a NetBeans XML project file and ANT build. The 'Wizard' will create a UML Class model of the project and Analyzer Scripts for each of the ANT targets you select. The process will also generate a script for each debug protocol you select through the 'Wizard'. These scripts should be used for debugging the project in Enterprise Architect. You will be presented with the choice of JDWP (Java Debug Wire Protocol), good for servers, and JVMTI (Java Virtual Machine Tools Interface), which is suited to standalone Java applications.

Import Options

When you select to import a Visual Studio or Mono Solution, the 'Visual Studio Solution Import' dialog displays. Complete the fields as directed in this table.

When you select to import an Eclipse or Netbeans solution, the appropriate Wizard start screen displays. Work through the screens as directed by the prompts on each screen.

Option	Description
<list of projects>	After you have selected the solution file, the projects in the solution are listed in the panel. Select the projects to be imported by the Wizard. You can use the All button to select all

	projects, and the None button to clear the selection of projects.
Select Solution File	Browse for and select the Solution file to import from. The Mono Solution files and Visual Studio Solution files have a .sln file extension.
Perform a Dry Run	Select this option to perform the import as a dry run, to check for any errors in the process or output before you repeat the import to change the model content. Click on the View Log button to check the log of the import.
Create Package per File	Select this option to perform the import with finer granularity, creating a separate Package for each file.
Import	Click on this button to start the import process.
Prompt for Missing Macro Definitions	Not applicable to Mono Solution imports. For C++ projects in Visual Studio, the parser might encounter unrecognized macros. If you select this option, you will be prompted when such an event occurs and will have the opportunity to define the macro. If you do not select this

	option, the resultant Class model could be missing certain items.
Create Diagram for Each Package	When selected, a Class diagram is created depicting the Class model for each Package. The result is a larger but more colorful model. Deselecting this option will cause diagram creation to be skipped and the import to run faster.
Generate Analyzer Scripts	For Visual Studio Solutions, selecting this option will generate Analyzer Scripts for each project configuration in addition to scripts for each Solution configuration. The scripts will allow for building and debugging the program(s) described by the solution immediately after the import completes. Select the 'Windows' checkbox; if you do not select this option, no Execution Analyzer features will be configured. For Mono Solutions, this option allows you to target either Linux or Windows. If you select Linux, it is assumed the machine on which Enterprise Architect is running is Linux, that the platform (Java or Mono) is installed there, and that the compiled programs run on Linux.

Startup Project	When this option is selected, the script for this Project will become the model default. The debugging tools, Execute ribbon and Toolbar buttons will automatically target this program.

Import Source Code

You can import source code into your Enterprise Architect model, to reverse-engineer a module. As the import proceeds, Enterprise Architect provides progress information. When all files are imported, Enterprise Architect makes a second pass to resolve associations and inheritance relationships between the imported Classes.

Procedure - Import source code

Step	Action
1	In the Browser window, select (or add) a diagram into which to import the Classes.
2	Click on the diagram background and either: • Select the 'Develop > Source Code > Files' ribbon option and click on the appropriate language, or • If the Code Generation toolbar is displayed, click on the 'Import' drop-down arrow and select the language to import The list of languages will include any customized languages you have created model structures for.
3	From the file browser that appears, locate and select

	one or more source code files to import.
4	Click on the Open button to start the import process.

Notes on Source Code Import

You can import code into your Enterprise Architect project, in a range of programming languages. Enterprise Architect supports most constructs and keywords for each coding language. You select the appropriate type of source file for the language, as the source code to import.

If there is a particular feature you require support for that you feel is missing, please contact Sparx Systems.

Notes

- When reverse engineering attributes with parameter substitutions (templated attributes):
  - If a Class with proper template parameter definitions is found, an Association connector is created and its parameter substitutions are configured
  - An Association connector is also created if a matching entry is defined as a Collection Class or in the 'Additional Collection Classes' option (for C#, C++ and Java); for an example, see *Example Use of Collection Classes*

Programming Language notes

Language	Notes
ActionScript	Appropriate type of source file: .as code file.
C	Appropriate type of source file: .h header files and/or .c files. When you select a header file, Enterprise Architect automatically searches for the corresponding .c implementation file to import, based on the options for extension and search path specified in the C options. Enterprise Architect does not expand macros that have been used, these must be added into the internal list of Language Macros.
C++	Appropriate type of source file: .h header file. Enterprise Architect automatically searches for the .cpp implementation file based on the extension and search path set in the C++ options; when it finds the implementation file, it can use it to resolve parameter names and method notes as necessary. When importing C++ source code, Enterprise Architect ignores function

	pointer declarations. To import them into your model you could create a typedef to define a function pointer type, then declare function pointers using that type; function pointers declared in this way are imported as attributes of the function pointer type. Enterprise Architect does not expand macros that have been used; these must be added into the internal list of Language Macros.
C#	Appropriate type of source file: .cs.
Delphi	Appropriate type of source file: .pas.
Java	Appropriate type of source file: .java. Enterprise Architect supports the AspectJ language extensions.  Aspects are modeled using Classes with the stereotype aspect; these aspects can

	then contain attributes and methods as for a normal Class. If an intertype attribute or operation is required, you can add a tag 'className' with the value being the name of the Class it belongs to. Pointcuts are defined as operations with the stereotype <<pointcut>>, and can occur in any Java Class, Interface or aspect; the details of the pointcut are included in the 'behavior' field of the method. Advice is defined as an operation with the stereotype <<advice>>; the pointcut this advice operates on is in the 'behavior' field and acts as part of the method's unique signature. afterAdvice can also have one of the Tagged Values returning or throwing.
PHP	Appropriate type of source file: .php, .php4, or .inc. Nested if condition syntax is enabled.
Python	Appropriate type of source file: .py.
Visual Basic	Appropriate type of source file: .cls Class file.

Visual Basic .NET	Appropriate type of source file: .vb Class file.

Import Resource Script

Enterprise Architect supports the import and export of Microsoft Windows Resource Scripts (as .rc files), which contain the Win32® dialog definitions (those with the stereotype «win32Dialog») for an application's graphical user interface. Dialog resources are imported and exported for a specific language, defaulting to the locale of the current computer system.

Access

Ribbon	Develop > Source Code > Files > Import Resource Script
Keyboard Shortcuts	F7 (synchronize element with code)

Import dialog resources from a .rc file

Option	Action
Resource File	Click on the ... button and locate the .rc

	file to import the screen elements(s) from.
Resource ID	Either: • Leave the default value 'All' to import all screen elements from the file, or • Click on the drop-down arrow and select the screen ID of a specific dialog to import
Language	Click on the drop-down arrow and select the language version (such as English - United States) of the dialog(s) to import.
Import	Click on this button to import the screens from the resource file. The progress of the import is reported in the field underneath the 'Language' field.

Export a dialog to a .rc file

Option	Action
Screen ID	Defaults from the Win32UI ID Tagged Value of the selected Screen element. (If the dialog does not have this ID, open

	the 'Win32UI' page of the element's 'Properties' dialog and provide a value for the ID tag.)
Resource File	Click on the ... button and locate the .rc file into which to export the screen element(s). If the element was previously imported, this field defaults to the source file.
Language	Click on the drop-down arrow and select the language version (such as English - United States) of the exported dialog.
Export	Click on this button to export the screens from the resource file. The progress of the export is reported in the field underneath the 'Language' field.

Notes

- New dialogs are exported to an existing .rc file
- In an export to an existing .rc file, no dialogs are ever deleted from the file, even when they are deleted from the model
- In an import, no dialogs are deleted from the model even

when omitted from the original .rc file

Import a Directory Structure

You can import from all source files in a complete directory structure, which enables you to import or synchronize multiple files in a directory tree in one pass.

Enterprise Architect creates the necessary Packages and diagrams during the import process.

Access

Ribbon	Develop > Source Code > Files > Import Source Directory
Keyboard Shortcuts	Ctrl+Shift+U

Import a directory structure, using the 'Import Source Directory' dialog

Field	Action
Root directory	Type in or browse for the name of the directory to import.

Source Type	Type in or select from the drop-down list the coding language of the files to import in the source directory.
File	Type in or select from the drop-down list, the file extensions to include in the import. Use a ';' to separate values.
Perform a Dry Run	If you want to perform the import as a dry run when you click on the OK button, select this check box. When processing is complete, click on the View Log button to check the predicted outcome of the process.
Recursively Process Subdirectories	If you want to include the contents of subdirectories in the import process, select this check box.
Import components from	If you want to import additional files (as described in the 'Import Component Types' dialog) select this checkbox. You then complete the prompt to specify where the components will come from.
Do not import	If you want to exclude private members from the model when importing libraries,

private members	select this checkbox.
Prompt for Missing Macro Definitions	During the import, the parser might encounter unrecognized macros. If you select this check box, you will be prompted when such an event occurs and will have the opportunity to define the macro. If you do not select this option, the resultant Package structure could be missing certain items.
Package Structure	Select the appropriate radio button to create a Package for every directory, every namespace or every file; this might be restricted depending on the source type selected.
Create Diagram for each Package	Select this checkbox to create a diagram in each Package created in the import. Click on the Options button to identify which element features to include on the diagrams.
Synchronization	Select the appropriate radio button to synchronize existing classes or overwrite existing classes. If a model Class is found that matches the one in code:

	• 'Synchronize' updates the model Class to include the details from the one in code, which preserves information not represented in code, such as the location of Classes in diagrams • 'Overwrite' deletes the model Class and generates a new one from code; any additional information is not preserved. If the option 'Use timestamps' is selected, then the representation with the latest time stamp (either model or code) will take precedence.
Remove Classes not found in code	Select the appropriate radio button to specify how to handle existing model classes that are not present in the imported code. • 'Never delete' retains all existing Classes in the model. • 'Prompt for action' enables you to review Classes individually • 'Always' delete' removes from the model any Class that is not present in the imported code.
OK	Click on this button to start the import.

Import Binary Module

Enterprise Architect enables you to reverse-engineer certain types of binary module.

Access

Ribbon	Develop > Source Code > Files > Import Binary Module

Use

Currently the permitted types are:

- Java Archive (.jar)
- .NET PE file (.exe, .dll) - Native Windows DLL and EXE files are not supported, only PE files containing .NET assembly data
- Intermediate Language file (.il)

Enterprise Architect creates the necessary Packages and diagrams during the import process; selecting the 'Do not import private members' checkbox excludes private members from libraries from being imported into the model.

When importing .NET files, you can import via reflection or

via disassembly, or let the system select the best method - this might result in both types being used.

The reflection-based importer relies on a .NET program, and requires the .NET runtime environment to be installed.

The disassembler-based importer relies on a native Windows program called Ildasm.exe, which is a tool provided with the MS .NET SDK; the SDK can be downloaded from the Microsoft website.

A choice of import methods is available because some files are not compatible with reflection (such as mscorlib.dll) and can only be opened using the disassembler; however, the reflection-based importer is generally much faster.

You can also configure:

- Whether to Synchronize or Overwrite existing Classes when found; if a model Class is found matching the one in the file:
  - Synchronize updates the model Class to include the details from the one in the file, which preserves information not represented in the file, such as the location of Classes in diagrams
  - Overwrite deletes the model Class and generates a new one from the file, which deletes and does not replace the additional information
- Whether to create a diagram for each Package
- What is shown on diagrams created by the import

Classes Not Found During Import

When reverse engineering from your code, there might be times when Classes are deliberately removed from your source code.

The 'Import Source Directory' functionality keeps track of the Classes it expects to synchronize with and, on the 'Import Directory Structure' dialog, provides options for how to handle the Classes that weren't found.

You can select the appropriate option to make Enterprise Architect, at the end of the import, ignore the missing Classes, automatically delete them or prompt you to manage them.

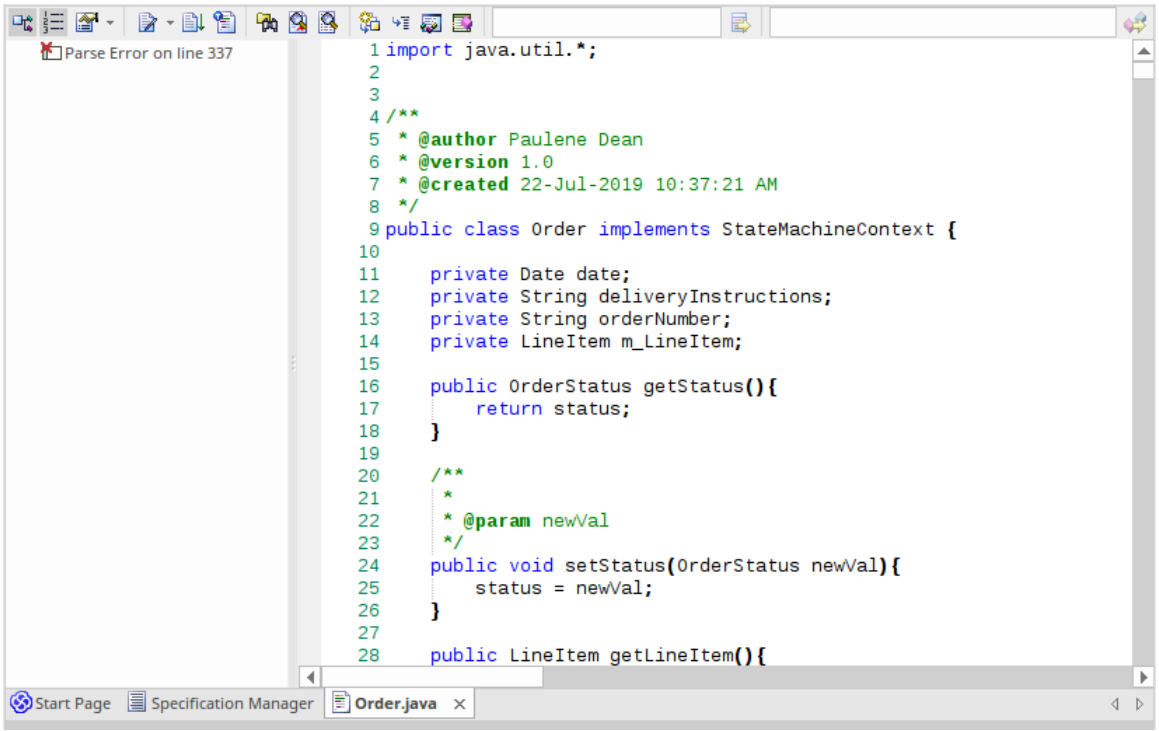
On the 'Import Directory Structure' dialog, if you select the 'Prompt For Action' radio button to manually review missing Classes, a dialog displays on which you specify the handling for each Class that was missing in the imported code.

By default, all Classes are marked for deletion; to keep one or more Classes, select them and click on the Ignore button.

Editing Source Code

Enterprise Architect contains a feature-rich source code editor that helps you to view, edit and maintain your source code directly inside the tool. Once source code has been generated for one or more Classes it can be viewed in this flexible editing environment. Seeing the code in the context of the UML models from which it is derived brings clarity to both the code and the models, and bridges the gap between design and implementation that has historically introduced errors into software systems.

The Source Code Editor is fully-featured, with a structure tree for easy navigation of attributes, properties and methods. Line numbers can be displayed and syntax highlight options can be configured. Many of the features that software engineers are familiar with in their favorite IDE, such as Intelli-sense and code completion are included in the editor. There are many additional features, such as macro recording that makes it easy to manage the source code inside Enterprise Architect. There are also many options for managing the code, available through the code editor context menu, toolbar and function keys.



For most programming languages a single file is created from a UML Class, but in the case of C++ both header and implementation classes are created and the source code editor displays these files in separate tabs.

A number of options change the way the source code editor works; they can be altered using the 'Preferences' dialog available from the Start ribbon:

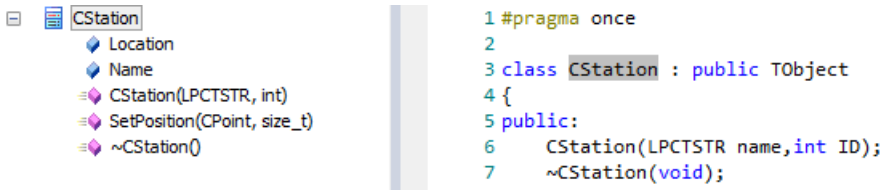
'Start > Appearance > Preferences > Preferences > Source Code Engineering > Code Editors'

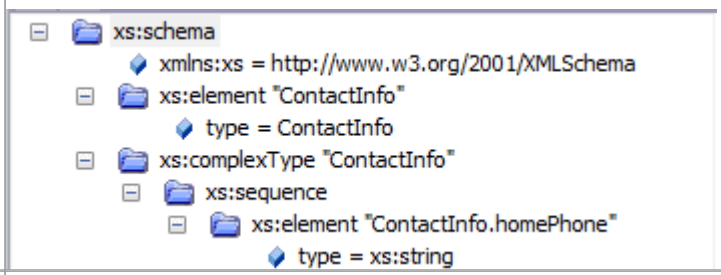
There are variants of the Source Code Editor, with different access methods. The variants are discussed in the *Compare Editors* topic.

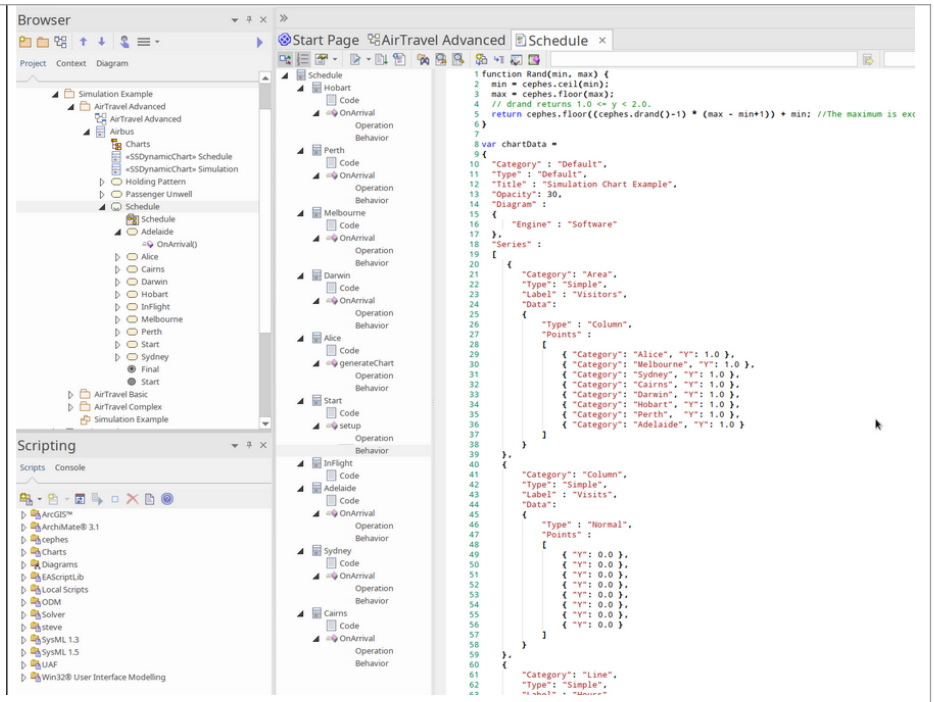
Access

Ribbon	Execute > Source > Edit > Open Source File (external file) or Execute > Source > Edit > Edit Element Source (for an existing source file) or Execute > Source > Edit > Edit New Source File or Design > Element > Behavior or Develop > Source Code > Behavior
Keyboard Shortcuts	F12 or Ctrl+E (for existing code for model elements) Ctrl+Alt+O (to locate external files)

Facilities

Facility	Description
Source Code editor	By default the Source Code editor is set to: • Parse all opened files, and show a tree of the results • Show line numbers 

	If you are editing an XML file, the structure tree mirrors the exact order and structure of the document. 
Structure Tree	The file structure tree is available for supported language files, such as C++, C#, Java and XML. The tree can be helpful to navigate content quickly in much the same way a table of contents would for other documents.
Simulation Behaviors	If you are editing the behaviors of the elements in a StateMachine or Activity diagram, the Code Editor allows you to list and edit the behaviors of all elements in the diagram together, using a structure tree.



In this illustration you can see a number of States within a StateMachine, each of which has operations and Behaviors, and all of which are listed together and can be selected without leaving or changing the editor window.

Notes

- For most selected elements you can use the keys F12 or Ctrl+E to view the source code.
- When you select an element to view source code, if the element does not have a generation file (that is, code has not been or cannot be generated, such as for a Use Case), Enterprise Architect checks whether the element has a

link to either an operation or an attribute of another element - if such a link exists, and that other element has source code, the code for that element displays

- You can also locate the directory containing a source file that has been created in or imported to Enterprise Architect, and edit it or its related files using an external editor such as Notepad or Visual Studio; click on the element in the Browser window and press Ctrl+Alt+Y

Languages Supported

The Source Code Editors can display code in a wide range of languages, as listed here. For each language, the editor highlights - in colored text - the standard code syntax.

- Ada (.ada, .ads, .adb)
- ActionScript (.as)
- BPEL Document (.bpel)
- C++ (.h, .hh, .hpp, .c, .cpp, .cxx)
- C# (.cs)
- DDL Structured Query Language (.sql)
- Delphi/Pascal (.pas)
- Diff/Patch Files (.diff, .patch)
- Document Type Definition (.dtd)
- DOS Batch Files (.bat)
- DOS Command Scripts (.cmd)
- HTML (.html)
- Interface Definition Language (.idl, .odl)
- Java (.java)
- JavaScript (.javascript)
- JScript (.js)
- Modified Backus-Naur Form Grammar (.mbnf)
- PHP (.php, .php4, .inc)
- Python (.py)

- Standard Generalized Markup Language (.sgml)
- SystemC (.sc)
- Visual Basic 6 (.bas)
- VB.NET (.vb)
- VBScript (.vbs)
- Verilog (.v)
- VHSIC Hardware Description Language (.vhdl)
- Visual Studio Resource Configuration (.rc)
- XML (eXtensible Markup Language) (.xml)
- XSD (XML Schema Definition)
- XSL (XML Stylesheet Language)

Configure File Associations

If you are a Windows® user, you can configure Enterprise Architect to be the default document handler for your language source files.

Access

Ribbon	Start > Appearance > Preferences > Preferences > Source Code Engineering > Code Editors : Configure Enterprise Architect File Associations ...

Actions

For each file type that you would prefer to open in Enterprise Architect, click on the checkbox to the left of the file type name. After selecting all of the document types you require, click on the Save button.

After this, clicking on any corresponding file in Windows® Explorer will open it in Enterprise Architect.

Notes

- You can change the default programs, or documents handled by them, directly through the 'Default Programs' option in Windows ® Control panel.

Compare Editors

Enterprise Architect provides four principal code editor variants, available through a number of access paths. The most direct access options are identified in these descriptions.

The first three code editor variants listed have the same display format, option toolbar, context menu options and internal function keys. They differ in their method of access and display mechanism.

Editor Variants

Variant	Details
Source Code View	F12 Ctrl+E Class context menu | 'View Source Code' Description: Displays the code on a tab of the Diagram View; the tab label shows the file name and extension (such as .java); again, for C++, there are two tabs for the Header and Implementation files. You can display the source code for other Classes on additional tabs, by reselecting the menu option/keys on the next Class.

Source Code window (Dockable)	Alt+7 'Execute > Source > Edit > Open Source File' Description: Displays the contents of the source file for a selected Class (except if the language is C++, when the window displays a tab for the Header file and a tab for the Implementation file). If you select a different Class, the window changes to show the code for the new Class (unless the first Class calls the second, in which case the window scrolls down to the second Class's code instead).
Internal Editor, External Source Code	Ctrl+Alt+O 'Execute > Source > Edit > Open Source File' ribbon option Description: Use this option if you intend to edit external code, XML or DDL files (that is, code not imported to or generated in Enterprise Architect). Displays an external browser, then opens the specific selected code file as a tab of the Diagram View (for C++, not two code files); otherwise this is identical to the F12 option.
External Editor,	Ctrl+Alt+Y

Internal or External Source Code	Class context menu | Open Source Directory Description: Displays an external file browser, open to the directory containing the selected Class's source files; you can open the files in Notepad, Visual Studio or other tools you might have on your system.

Code Editor Toolbar

When you are reviewing the code for a part of your model in the Source Code editor, you can access a wide range of display and editing functions from the editor toolbar.

Code Editor Toolbar

Toolbar Options

Structure Tree	Click on this icon to show or hide the element hierarchy panel (the left panel of the Source Code editor).
Line Numbers	Click on this icon to show or hide the line numbers against the lines of code.
Source Code Engineering Properties	Click on the drop-down arrow to display a menu of options to select individual 'Source Code Engineering' pages of the 'Preferences' dialog, from which you can configure display and behavior options for source code engineering:

- Language
- Syntax Highlighting Options
- Code Editor Options
- Code Engineering Options
- Code Editor Key Bindings

Editor Functions

Click on the drop-down arrow to display a menu providing access to a range of code editing functions:

- Open Corresponding File (Ctrl+Shift+O) - opens the header or implementation file associated with the currently-open file
- Go to Matching Brace (Ctrl+E) - for a selected opening or closing brace, highlights the corresponding closing or opening brace in the pair
- Go to Line (Ctrl+G) - displays a dialog on which you select the number of the line to highlight; click on the OK button to move the cursor to that line
- Cursor History Previous (Ctrl+-) - the Source Code viewer keeps a history of the previous 50 cursor positions, creating a record when the cursor is moved either more than 10 lines away from its previous position, or in a find-and-replace operation; the menu

option moves the cursor to the position in the immediately-previous cursor history record

- Cursor History Next (Ctrl+Shift+-) - if you have moved to an earlier cursor position, this option moves the cursor to the position in the immediately-following cursor history record
- Find (Ctrl+F) - displays a dialog in which you define a text string and search options to locate that text string in the code
- Replace (Ctrl+R) - displays a dialog in which you define a text string and search options to locate that text string in the code and replace it with another text string; the dialog has options to locate and replace each occurrence as you decide, or to replace all occurrences immediately
- Highlight Matching Words - (Ctrl+3) Enables or disables the highlighting of matching words during a find operation; by default this option is enabled
- Record Macro - records your next keystrokes to be saved as a macro
- Stop Recording and Save Macro - stops

recording the keystrokes and displays the 'Save Macro' dialog on which you specify a name for the macro

- Play Macro - displays the 'Open Macro' dialog from which you select and execute a saved macro, to repeat the saved keystrokes
- Toggle Line Comment (Ctrl+Shift+C) - comments out (//) or re-establishes the code for each full line in which text is highlighted
- Toggle Stream Comment (Ctrl+Shift+X) - inserts a stream comment (/* */) at the cursor position (comments out only the highlighted characters and lines), or re-establishes the commented text as code
- Toggle Whitespace Characters (Ctrl+Shift+W) - shows or hides the spacing characters: --> (tab space) and . (character space)
- Toggle EOL Characters (Ctrl+Shift+L) - shows or hides the end-of-line characters: CR (carriage return) and LF (line feed)
- Toggle Tree Synchronization - selects the tree item automatically as context changes within code editor
- Open Containing Folder - opens the file

	browser at the folder containing the code file; you can open other files in your default external editor for comparison and parallel work
Save Source and Resynchronize Class	Click on this icon to save the source code and resynchronize the code and the Class in the model.
Code Templates	Click on this icon to access the Code Templates Editor, to edit or create code templates for code generation.
Find in Project Browser	For a selected line of code, click on this icon to highlight the corresponding structure in the Browser window. If there is more than one possibility the 'Possible Matches' dialog displays, listing the occurrences of the structure from which you can select the required one.
Search in Files	Click on this icon to search for the selected object name in associated files, and display the results of the search in the File Search window. You can refine and refresh the search by specifying criteria on the Find in Files window toolbar.
Search in	Click on this icon to search for the

Model	selected text throughout the model, and display the results of the search in the Find in Project view.
Go to Declaration	Click on this icon to locate the declaration of a symbol in the source code.
Go to Definition	Click on this icon to locate the definition of a symbol in the source code (applicable to languages such as C++ and Delphi, where symbols are declared and defined in separate files).
Autocomplete List	Click on this icon to display the autocompletion list of possible values; double-click on a value to select it.
Parameter Information	When the cursor is between the parentheses of an operation's parameter list, click on this icon to display the operation's signature, highlighting the current parameter.
Find Current Class in Browser Window	Click on this icon to display the name of the currently-selected Class in the code, and highlight that name in the Browser window; if there is more than one possibility the 'Possible Matches' dialog

displays, listing the occurrences of the Class from which you can select the required one.

Find Member Click on this icon to display the name of the currently-selected attribute or method in the code, and highlight that name in the Browser window; if there is more than one possibility the 'Possible Matches' dialog displays, listing the occurrences of the feature from which you can select the required one.

Notes

- The 'Record Macro' option disables Intelli-sense while the macro is being recorded
- You can assign key strokes to execute the macro, instead of using the toolbar drop-down and 'Open Macro' dialog

Code Editor Context Menu

When working on a file with a code editor, you can perform a number of code search and editing operations to review the contents of the file. These options are available through the editor context menu, and can vary depending on which code editor you are using.

Access

Context Menu	Right-click on the code text string you are working on

Options

Go to Declaration Locate and highlight the declaration of a symbol in the source code.

Go to Definition Locate and highlight the definition of a symbol in the source code (applicable to languages such as C++ and Delphi, where symbols are declared and defined in separate places).

Open in Grammar Editor	Opens a view that lets you examine or validate the code using the appropriate grammar.
Synchronize Tree to Editor	Finds and displays the current element (method for example) in the structure tree.
Auto Synchronize Tree and Editor	When selected, the structure tree will automatically show the element being worked on in the editor.
XML Schema Validation	Allows an XML schema to validated.
Search for '<string>'	Display a submenu providing options to locate the selected text string in a range of locations. • 'Find in Project Browser' - Highlight the object containing the selected text in the Browser window • 'Search in Open Files' - Search for the selected text string in associated open files and display the results of the search in the Find in Files window; you can refine and refresh the search by specifying criteria on the Find in Files

window toolbar

- 'Search in Files' - Search for the selected text string in all associated files (closed or open), and display the results of the search in the Find in Files window; you can refine and refresh the search by specifying criteria on the Find in Files window toolbar (shortcut key: F12)
- 'Search in Model' - Perform an 'Element Name' search in the Model Search facility, and display the results on the Model Search tab
- 'Search in Scripts' - (Available while working in the Script Editor) Open the Find in Files window, set the 'Search Path' field to 'Search in Scripts' and the 'Search Text' field to the selected text, then search all scripts for the text string and display the results of the search; you can refine and refresh the search by specifying criteria on the Find in Files window toolbar
- 'EA User Guide' - Display the description of the code item in the *Enterprise Architect User Guide*
- 'Google' - Display the results of a Google search on the text
- 'MSDN' - Display the results of a

search on the text in the Microsoft Developer Network (MSDN)

- 'Sun Java SE' - Display the results of a search on the text in the Sun Microsystems 'Sun Search' facility
- 'Wikipedia' - Display any entry on the object on the Wikipedia web site
- 'Koders' - Display the results of a search for the text string on Koders.com

Search Intelli-sense

Perform a search on the specified string using the Code Miner service or library specified in the current Analyzer Script. The results are displayed in the 'Code Miner' tab of the Find in Files window.

Shortcut key: Shift+F12

Set Debugger to Line

(If the debugger is executing and has reached a breakpoint.) Move the execution point to the current line. Check that you do not skip over any code or declarations that affect the next section of code being debugged.

Display Variable

(If the debugger is executing.) Open the Locals window and highlight the local variable for the current point in the code.

Show in String Viewer	Display the full contents of a variable string in the String Viewer.
Create Use Case for '<string>'	Display the 'Create Use Case For Method' dialog, through which you create a Use Case for the method containing the text string.
Breakpoint	Display a submenu of options for creating a recording marker on the selected line of code. The recording markers you can add include: • Breakpoint • Start Recording Marker • End Recording Marker • Stack Auto Capture Marker • Method Auto Record Marker • Tracepoint
Testpoints	Display options to add a new Testpoint, show the Testpoints Manager (Testpoints window) or edit an existing Testpoint if one or more are already defined at the selected location. (The sub-options depend on the type of code file you are reviewing.)

XML Validation	Allows an XML document to be checked for compliance with its own schema references or using a user-specified schema; either a local schema file or a URL.
Open (Close) IME	Open (or close) the Input Method Editor, so that you can enter text in a selected foreign language script, such as Japanese. You set the keyboard language using the Windows Control Panel - Regional and Language Options facility.
Line Numbers	(Script Editor only.) Show or hide the code line numbers on the left hand side of the editor screen.
Undo **Cut** **Copy** **Paste** **Delete** **Select All**	These six options provide simple functions for editing the code.

Notes

- The options in the lower half of the 'Search for <string>'

submenu (after 'Search in Scripts') are configurable; you can add new search tools or remove existing ones by editing the searchProviders.xml file in the Sparx Systems > EA > Config folder - this file is in OpenSearch description document format

Create Use Case for Method

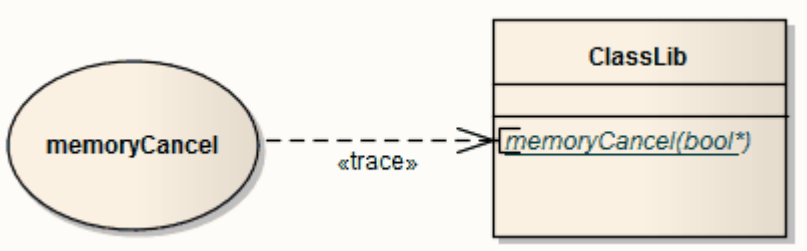
Using the code editor context menu, you can create a Use Case element for a method that you select from the code. You can also:

- Link the Use Case directly to the method
- Add the parent Class to a diagram (if it is not already in the selected diagram) and/or add the Use Case element to the diagram
- Block from display any attributes or methods that are not also the targets of feature links

Create a Use Case for a method, through the code editor

Step	Action
1	(If you want to depict the Use Case and its link to the method in a diagram) click on the diagram name in the Browser window.
2	In the code editor, right-click on either the method name or any part of the method body, and select the 'Create Method for <methodname>' option. The 'Create Use Case for Method' dialog displays.

3	The basic function of this dialog is to create a Use Case for the selected method: • If this is all that is required, click on the OK button; the Use Case element is created in the Browser window, in the same Package as the parent Class for the method, and with the same name as the method • If you intend to make the relationship tangible, continue with the procedure
4	To create a Trace connector linking the Use Case to the method, select the 'Link Use Case to Method' checkbox.
5	To add the method's parent Class to the diagram, if it is not already there, select the 'Add Class to Diagram' checkbox.
6	To add the newly-created Use Case to the diagram, select the 'Add Use Case to Diagram' checkbox; this would now show the Use Case, Class and Trace connector on the diagram.
7	To only show the features (attributes and methods) of the parent Class that are the targets of 'link to feature' relationships, select the 'Display only linked features in Class' checkbox. The Class might contain any number of attributes

	and methods, but those without a 'link to feature' relationship are hidden.
8	Click on the OK button to create and depict the Use Case and relationship; if you selected all options, the diagram now contains linked elements resembling this illustration: 

Code Editor Functions

The common Code Editor provides a variety of functions to assist with the code editing process, including:

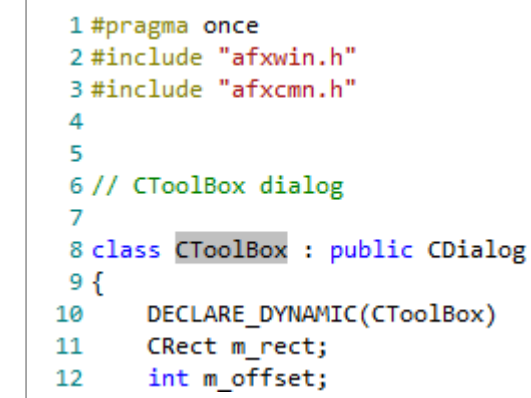
- Syntax Highlighting
- Bookmarks
- Cursor History
- Brace Matching
- Automatic Indentation
- Commenting Selections
- Scope Guides
- Zooming
- Line Selection
- Intelli-sense
- Find and Replace
- Find in Files

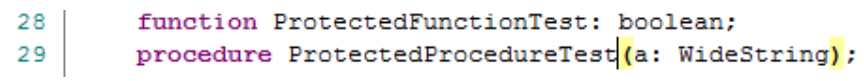
A range of these functions is available through keyboard key combinations and/or context menu options.

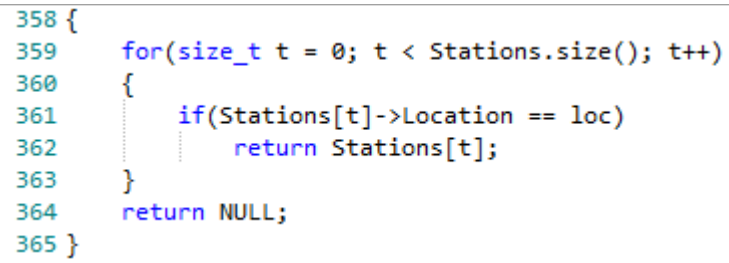
You can customize several of the Code Editor features by setting properties in the Code Editor configuration files; for example, by default the line containing the cursor is always highlighted, but you can turn the highlighting off.

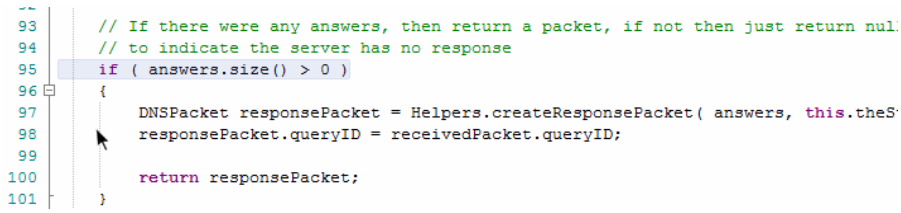
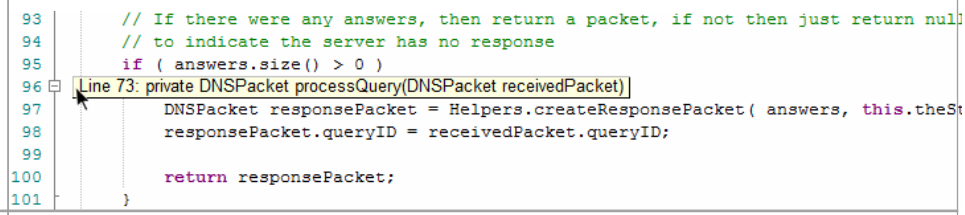
Function Details

Code Editor Functions

Function	Description
Syntax Highlighting	The Code Editor highlights - in colored text - the standard code syntax of all language file formats supported by Enterprise Architect  You can define how the Code Editor implements syntax highlighting for each language, through the 'Code Editors' page of the 'Preferences' dialog.
Bookmarks	Bookmarks denote a line of interest in the document; you can toggle them on and off for a particular line by pressing Ctrl+F2. Additionally, you can press F2 and Shift+F2 to navigate to the next or

	previous bookmark in the document. To clear all bookmarks in the code file, press Ctrl+Shift+F2.
Cursor History	The Code Editor Control keeps a history of the previous 50 cursor positions; an entry in the history list is created when: • The cursor is moved more than 10 lines from its previous position • The cursor is moved in a find/replace operation You can navigate to an earlier point in the cursor history by pressing Ctrl+-, and to a later point by pressing Ctrl+Shift+-.
Brace Matching	When you place the cursor over a brace or bracket, the Code Editor highlights its corresponding partner; you can then navigate to the matching brace by pressing Ctrl+E. 
Automatic Indentation	For each supported language, the Code Editor adjusts the indentation of a new line according to the presence of control statements or scope block tokens in the lines leading up to the cursor position.

	 The levels of indent are indicated by pale horizontal lines. You can also manually indent selected lines and blocks of code by pressing the Tab key; to un-indent the selected code, press Shift+Tab.
Commenting Selections	For languages that support comments, the Code Editor can comment entire selections of code. The Code Editor recognizes two types of commenting: • Line Commenting - entire lines are commented from the start (for example: // This is a comment) • Stream Commenting - sections of a line are commented from a specified start point to a specified end point (for example: /* This is a comment */) You can toggle comments on the current line or selection by pressing:

	• Ctrl+Shift+C for line comments, or • Ctrl+Shift+X for stream comments
Scope Guides	If the cursor is placed over an indentation marker, the Code Editor performs a 'look back' to find the line that started the scope at that indentation level; if the line is found and is currently on screen, it is highlighted in light blue.  Alternatively if the line is off screen, a calltip is displayed advising of the line number and contents: 
Zooming	You can zoom into and out of the contents of the Code Editor using: • Ctrl+keypad + and • Ctrl+keypad - Zoom can be restored to 100% using Ctrl+keypad /.
Line	If you want to move the cursor to a

Selection	specific line of code, press Ctrl+G and, in response to the prompt, type in the line number. Press the OK button; the editor displays the specified line of code with the cursor at the left.

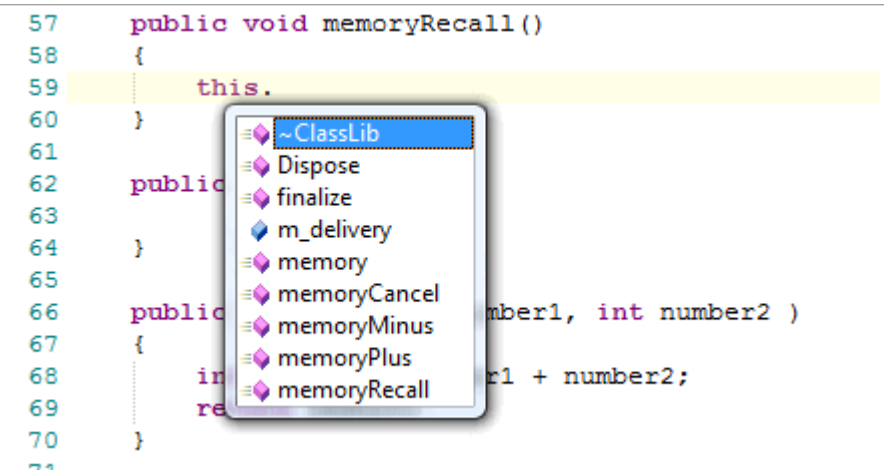
Intelli-sense

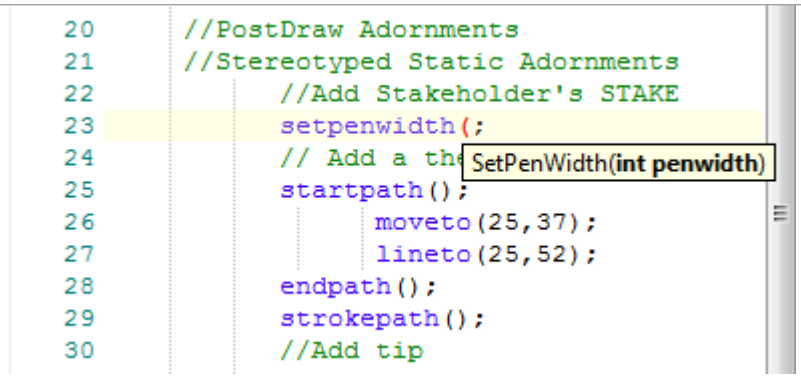
Intelli-sense is a feature that provides choices of code items and values as you type. Not all code editors use Intelli-sense; for example, Intelli-sense is disabled while you record a macro in the Source Code Viewer.

Intelli-sense provides you with context-based assistance through autocompletion lists, calltips and mouseover information.

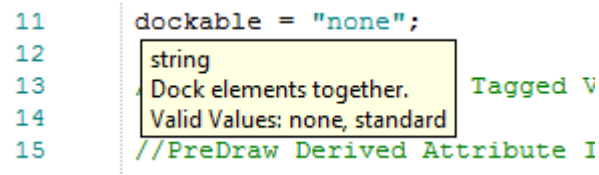
Facilities

Facility	Description
Autocompletion List	An autocompletion list provides a list of possible completions for the current text; the list is automatically invoked when you enter an accessor token (such as a period or pointer accessor) after an object or type that contains members.

	 You can also invoke the autocompletion list manually by pressing Ctrl+Space; the Code Editor then searches for matches for the word leading up to the invocation point. Select an item from the list and press the Enter key or Tab key to insert the item into the code; to dismiss the autocompletion list, press Esc.
Calltips	Calltips display the current method's signature when you type the parameter list token (for example, opening parenthesis); if the method is overloaded, the calltip displays arrows that you can use to navigate through the different method signatures



Mouseover Information	You can display supporting documentation for code elements (for example, attributes and methods) by hovering the cursor over the element in question.



Find and Replace

Each of Enterprise Architect's code editors facilitates searching for and replacing terms in the editor, through the 'Find and Replace' dialog.

Access

Keyboard Shortcuts	Highlight the required text string and press: • Ctrl+F for the find controls only, or • Ctrl+R for both find and replace controls In each instance, the 'Find what' field is populated with the text currently selected in the editor. If no text is selected in the editor, the 'Find what' field is populated with the word at the current cursor position. If no word exists at the current cursor position, the last searched-for term is used.

Basic Operations - Commands

Command	Action
Find Next	Locate and highlight the next instance (relative to the current cursor position) of the text specified in the 'Find what' field.
Replace	Replace the current instance of the text specified in the 'Find what' field with the text specified in the 'Replace with' field, and then locate and highlight the next instance (relative to the current cursor position) of the text specified in the 'Find what' field.
Replace All	Automatically replace all instances of the text specified in the 'Find what' field with the text specified in the 'Replace with' field.

Basic Operations - Options

Option	Action
Match Case	Specify that the case of each character in the text string in the 'Find what' field is

	significant when searching for matches in the code.
Match whole word	Specify that the text string in the 'Find what' field is a complete word and should not be matched with instances of the text that form part of a longer string. For example, searches for ARE should not match those letters in instances of the words AREA or ARENA.
Search up	Perform the search from the current cursor position up to the start of the file, rather than in the default direction of current cursor position to end of file.
Use Regular Expressions	Evaluate specific character sequences in the 'Find what' and 'Replace with' fields as Regular Expressions.

Concepts

Concept	Description
Regular Expressions	A Regular Expression is a formal definition of a Search Pattern, which can

	be used to match specific characters, words or patterns of characters. For the sake of simplicity, the Code Editor's 'find and replace' mechanism supports only a subset of the standard Regular Expression grammar. Text in the 'Find what' and 'Replace with' fields is only interpreted as a Regular Expression if the 'Use Regular Expressions' checkbox is selected in the 'Find and Replace' dialog.
Metasequences	If the 'Use Regular Expressions' checkbox is selected, most characters in the 'Find what' field are treated as literals (that is, they match only themselves). The exceptions are called metasequences; each metasequence recognized in the Code Editor 'Find and Replace' dialog is described in this table: • \< - Indicates that the text is the start of a word; for example: \<cat is matched to *catastrophe* and *cataclysm*, but not *concatenate* • \> - Indicates that the text is the end of a word; for example: hat\> is matched to *that* and *chat*, but not *hate* • (...) - Indicates alternative single characters that can be matched - the

	characters can be specific (chr) or in an alphabetical or numerical range (a-m); for example: (hc) at is matched to *hat* and *cat* but not *bat*, and (a-m) Class is matched to any name in the range *aClass-mClass* • (^...) - Indicates alternative single characters that should be excluded from a match - the characters can be specific (^chr) or in an alphabetical or numerical range (^a-m); for example: (^hc) at is matched to *rat* and *bat*, but *hat* and *cat* are excluded, and (^a-m) Class is matched to any name in the range *nClass* to *zClass*, but *aClass* to *mClass* are excluded • ^ - Matches the start of a line • $ - Matches the end of a line • * - Matches the preceding character (or character set) 0 or more times; for example: ba*t is matched to *bt*, *bat*, *baat*, *baaat* and so on, and b(ea) *t is matched to *bt*, *bet*, *bat*, *beat*, *beet*, *baat* and so on • + - Matches the preceding character (or character set) 1 or more times; for example: ba+t is matched to *bat*, *baat* and *baaat* but not *bt*, and b(ea) +t is matched to *bet*, *bat*, *beat*, *beet* and *baat*

	but not *bt* If a single character metasequence is preceded by a backslash (\) it is treated as a literal character: c\(at\) matches c(at) as the brackets are treated literally. When the 'Use Regular Expressions' checkbox is selected, a metasequence helper menu is available to the right of both of the 'Find what' and 'Replace with' fields; selecting a metasequence from this menu inserts the metasequence into the field, replacing or wrapping the currently selected text as appropriate.
Tagged Regions	When 'find and replacing' with Regular Expressions, up to nine sections of the original term can be substituted into the replacement term. The metasequences '\(' and '\)' denote the start and the end of a tagged region; the section of the matched text that falls within the tagged region can be included in the replacement text with the metasequence '\n' (where *n* is the tagged region number between 1 and 9). For example: Find: \((A-Za-z) +\)*'s things* Replace with *items that belong to \1*

	Original text: *These are all Michael's things*. Replaced text: *These are all items that belong to Michael.*

Search in Files

File Text Searches are provided by the Find in Files window and from within the Code Editors, to search files for data names and structures. These files can be external code files, code files that you have already opened in Enterprise Architect, internal model scripts or the Help subsystem.

The 'File Search' tab maintains a history of the file paths you have explored, helping you to quickly return to frequently-used folders in your file system. You can similarly select a previously-used search string, if you need to repeat a search several times. When you are searching code files, you can also confine the search to files of specific types, by selecting the file extensions, and to include just the selected folder or all of its sub-folders as well. Another useful facility is being able to select to show the results of the search as either a list of every instance of the string, or a list of files containing the string with the instances grouped under the file in which they are found.

For all searches, you can qualify the search to be case-sensitive and/or to match the search string to complete words.

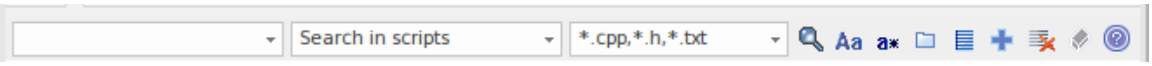
Access

Ribbon	Explore > Search > Files

	Execute > Source > Find Execute > Source > Edit > Search in Files
Context Menu	Right-click on selected text | Search for <selected text> | Search in Files
Keyboard Shortcuts	F12, Ctrl+Shift+Alt+F

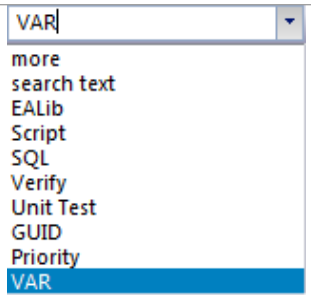
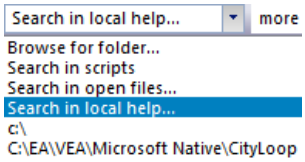
Search Toolbar

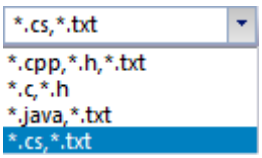
You can use the toolbar options in the Find in Files window to control the search operation. The state of each button persists over time to always reflect your previous search criteria.



Options

Option	Action
	The 'Search Text' field. Type the text string to search for. Any text you type in is automatically

	saved in the drop-down list, up to a maximum of ten strings; text added after that overwrites the oldest text string in the list. You can click on the drop-down arrow and select one of these saved text strings, if you prefer.
	The 'Search Path' field. Specify the folder to search, or the type of search. You can type the folder path to search directly into the text box, or click on the drop-down arrow and select 'Browse for folder' to search using the 'Browse for Folder' dialog. Any paths you enter are automatically saved in the drop-down list, up to a maximum of ten; paths added after that overwrite the oldest path in the list. You can select one of these saved paths if you prefer. Apart from 'Browse for folder', there are three other fixed options in the drop-down list: • 'Search in scripts', which searches the local and user-defined scripts in the Scripting window • 'Search in open files', which confines the search to the files that you have open in Enterprise Architect

	• 'Search in local help', which searches the local Help files that have been installed from the Sparx Systems web site; the results list the Help topics containing the search term, and the line number and line in which the text occurs These options disable the 'Search File Types' list box.
	The 'Search File Types' field. Click on the drop-down arrow and select the file types (file extensions) to search.
	Click on this icon to begin the search. During the course of the search all other buttons in the toolbar are disabled. You can cancel the search at any time by clicking on the Search button again. If you switch any of these toggle buttons, you must run the search again to change the output.
	Click on this icon to toggle the case sensitivity of the search. The tool-tip message identifies the current setting.
	Click on this icon to toggle between searching for any match and searching for

	only those matches that form an entire word. The tool-tip message identifies the current setting.
	Click on this icon to toggle between limiting the search to a single path and including all subfolders under that path. The tool-tip message identifies the current setting.
	Click on this icon to select the presentation format of the search results; you have two options: • List View - (as shown) each result line consists of the file path and line number, followed by the line text; multiple lines from one file are listed as separate entries • Tree View - () each result line consists of the file path that matches the search criteria, and the number of lines matching the search text within that file; you can expand the entry to show the line number and text of each line
	Click on this icon to add a new search tab. You can create up to four new search tabs. Searches can also run concurrently.

	Click on this icon to clear the results.
	If necessary, click on this icon to remove all the entries in the Search Path, Search Text and Search File Types drop-down lists.

Find File

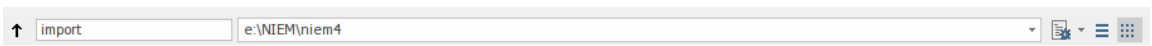
The Find in Files window 'Find File' tab provides a tool that can help you find files quicker. The tab acts as a file system explorer and offers a speedy alternative to the common open file dialog. File searches are quick and simple, allowing you to look up files of interest without losing your current workflow. The display can be switched between report and list view.

Access

Ribbon	Explore > Search > Files > Find File
Keyboard Shortcuts	Ctrl+Shift+Alt+F

Toolbar

The toolbar provides a search filter and folder navigation combo box. The toolbar provides options to remember search locations and alternate between list and report views.



Options

	Click to navigate to the parent folder.
	The filter control allows you to exclude files that do not match the criteria you type. The wildcard symbol * is automatically appended to the text so it is not necessary to add it yourself. To search for all files that contain the term 'jvm' simply type 'jvm'. To find .png images containing the term 'red' you could type *red*.png. Press the Enter key to update the results.
	Enter the path of a directory and press the Enter key to display the files in that location Use the drop down list to select from book-marked locations for the current model. Locations can be managed by using the toolbar menu.
	Allows you to manage the locations displayed in the directory combo.

	• Remember Path - stores the current value of the 'Directory' field so that, when you return to the Find in Files window at a later point the 'Directory' field either defaults to that value (if it is the only 'remembered' value) or offers the value in the drop-down list • Forget Path - clears the current value from memory so that it is not offered as a possible value for the 'Directory' field • Remember Filter - stores the current value in the 'Filter' field so that when you return to the Find in Files window at a later point the 'Filter' field defaults to that value • Forget Filter - removes the 'Filter' field value from memory so that it is not placed in the field next time you access the window
	In this view the list displays the columns 'Name', 'Modified Date', 'Type' and 'Size'. Columns can be sorted in either ascending or descending order. Click the column a third time to remove the sort order.
	The list view removes columns and is convenient when a folder contains many

	files.

Keyboard Shortcuts

	Sets focus to the filter control.
	Navigates to the parent folder.
	Navigates to the parent folder.
	If a folder is selected, opens the folder, otherwise opens the selected files.

Search Intelli-sense

The Intelli-sense capabilities of Enterprise Architect are built using Sparx Systems' Code Miner tool. The Code Miner provides fast and comprehensive access to the information in an existing code base. The system provides complete access to all aspects of the original source code, either 'on the fly' as one might search in a code editor, or as search results produced by queries written in the Code Miner mFQL language.

Access

On the Find in Files window, click on the 'Code Miner' tab.

Ribbon	Explore > Search > Files
Keyboard Shortcuts	Ctrl+Shift+Alt+F

The Code Miner Control

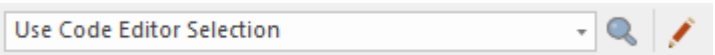
This control presents an interface for performing queries on several code bases at once. The code bases it uses are databases built using Enterprise Architect's Code Miner tool.

These databases form a library, which can also be shared when deployed as a service. The queries that can be run are listed and selected using the toolbar, which allows easy access to the source code for the queries, for editing and composition. Queries do not need to be compiled; they are viewed, edited and saved as one would any source code file. Queries that take a single parameter can utilize any selection in an open code editor. The interface also supports manual parameter entry for queries that take multiple arguments.

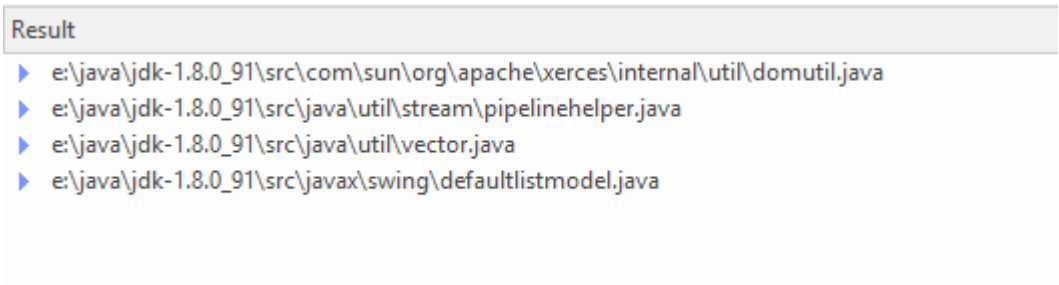
The first control on the toolbar lists the namespaces available. Selecting a namespace limits the queries that are displayed to those within that namespace.

The next control provides a drop-down list of all the queries in the query file for the selected namespace.

The third control is an edit combo box. By default a single query parameter is taken from the selected text in an open code editor, but you can also type the parameter(s) directly into this field. Multiple parameters should be separated by commas. This is followed by the Search button to run the query. Queries can be edited at any time using the Edit button next to the Search button.



The 'Result' panel is a tree control that lists the results of the query grouped by file.



Code Miner Libraries

Code Miner libraries are a collection of databases that can be used by Enterprise Architect Intelli-sense providers to obtain and query for information across several code bases. Each database is created from the root source code directory of a code base, using a specialized grammar appropriate for its language (C++, Java or C#).

The libraries are created, updated, removed or added in the 'Analyzer Script Editor'. A typical scenario for using this feature would be to create a database for a development project and additional databases for frameworks referenced by the project. Your development database can be updated frequently as code changes accrue, while the static frameworks would be updated less often. Libraries can be searched in a similar way to the 'File Search' tool, but Code Miner offers advanced search capabilities due to its mFQL language.

- Multiple domains / frameworks can be searched at once
- A query can be run in a fraction of the time required for a File Search
- Queries can be coded to assist with complex search

criteria
- Queries can take multiple parameters
- All files are indexed based on equivalent UML constructs, allowing intelligent searches producing meaningful results in a modeling setting

Code Miner Query Files

Code Miner queries are maintained in a single source code file which should have the .mFQL extension. A basic set of queries is provided with each Enterprise Architect installation; these can be located in the config\codeminer sub directory. This query file should be named by default in any Analyzer Script you edit.

Before editing any queries it is advisable that you copy this file to a working location and name the copy in any Analyzer Script you use. This way you will always have a reference file to go back to.

Queries are best considered as functions that are written in the mFQL language. As such they have unique names, can be qualified by a single namespace and can specify parameters. The file provides the queries listed in the Intelli-sense control's toolbar. Whenever edits to a query file are saved, the queries listed in the search toolbar combo box will be updated accordingly. This image is an example of a simple query written in mFQL.

```
188
189 namespace java
190 {
191 //
192 // Find all references
193 //
194 query::findByName($param1)
195 {
196     distinct(GetByValue( $param1 +))
197 }
198
199 query::findMethodByName($name)
200 {
201     move( 1, "METHOD", intersect( GetByNode("NAME"), GetByValue( $name ) )  )
202 }
203
204 query::findMethodCall($name)
205 {
206     filter( "METHOD_ACCESS", intersect(GetByNode("NAME"), GetByValue( $name )) )
207 }
208
```

Code Editor Key Bindings

Keys

Key	Description
Ctrl+G	Move cursor to a specified line
↓	Move cursor down one line
Shift+↓	Extend selection down one line
Ctrl+↓	Scroll down one line
Alt+Shift+↓	Extend rectangular selection down one line
↑	Move cursor up one line
Shift+↑	Extend selection up one line
Ctrl+↑	Scroll up one line
Alt+Shift+↑	Extend rectangular selection up one line
Ctrl+(	Move cursor up one paragraph

Ctrl+Shift+(	Extend selection up one paragraph
Ctrl+)	Move cursor down one paragraph
Ctrl+Shift+)	Extend selection down one paragraph
←	Move cursor left one character
Shift+←	Extend selection left one character
Ctrl+←	Move cursor left one word
Ctrl+Shift+←	Extend selection left one word
Alt+Shift+←	Extend rectangular selection left one character
→	Move cursor right one character.
Shift+→	Extend selection right one character
Ctrl+→	Move cursor right one word
Ctrl+Shift+→	Extend selection right one word
Alt+Shift+→	Extend rectangular selection right one

	character
Ctrl+/	Move cursor left one word part
Ctrl+Shift+/	Extend selection left one word part
Ctrl+\	Move cursor right one word part
Ctrl+Shift+\	Extend selection right one word part
Home	Move cursor to the start of the current line
Shift+Home	Extend selection to the start of the current line
Ctrl+Home	Move cursor to the start of the document
Ctrl+Shift+Home	Extend selection to the start of the document
Alt+Home	Move cursor to the absolute start of the line
Alt+Shift+Home	Extend rectangular selection to the start of the line
End	Move cursor to the end of the current line

Shift+End	Extend selection to the end of the current line
Ctrl+End	Move cursor to the end of the document
Ctrl+Shift+End	Extend selection to the end of the document
Alt+End	Move cursor to the absolute end of the line
Alt+Shift+End	Extend rectangular selection to the end of the line
Page Up	Move cursor up a page
Shift+Page Up	Extend selection up a page
Alt+Shift+Page Up	Extend rectangular selection up a page
Page Down	Move cursor down a page
Shift+Page Down	Extend selection down a page
Alt+Shift+Pa	Extend rectangular selection down a page

ge Down	
Delete	Delete character to the right of the cursor
Shift+Delete	Cut selection
Ctrl+Delete	Delete word to the right of the cursor
Ctrl+Shift+Delete	Delete until the end of the line
Insert	Toggle overtype
Shift+Insert	Paste
Ctrl+Insert	Copy selection
Backspace	Delete character to the left of the cursor
Shift+Backspace	Delete character to the left of the cursor
Ctrl+Backspace	Delete word to the left of the cursor
Ctrl+Shift+Backspace	Delete from the start of the line to the cursor

Alt+Backspace	Undo delete
Tab	Indent cursor one tab
Ctrl+Shift+I	Indent cursor one tab
Shift+Tab	Unindent cursor one tab
Ctrl+keypad(+)	Zoom in
Ctrl+keypad(-)	Zoom out
Ctrl+keypad(/)	Restore Zoom
Ctrl+Z	Undo
Ctrl+Y	Redo
Ctrl+X	Cut selection
Ctrl+C	Copy selection
Ctrl+V	Paste

Ctrl+L	Cut line
Ctrl+T	Transpose line
Ctrl+Shift+T	Copy line
Ctrl+A	Select entire document
Ctrl+D	Duplicate selection
Ctrl+U	Convert selection to lowercase
Ctrl+Shift+U	Convert selection to uppercase
Ctrl+E	Move cursor to matching brace
Ctrl+Shift+E	Extend selection to matching brace
Ctrl+Shift+C	Toggle line comment on selection
Ctrl+Shift+X	Toggle stream comment on selection.
Ctrl+F2	Toggle bookmark
F2	Go to next bookmark
Shift+F2	Go to previous bookmark

Ctrl+Shift+F2	Clear all bookmarks in current file
Ctrl+Shift+W	Toggle whitespace characters
Ctrl+Shift+L	Toggle EOL characters
Ctrl+Space	Invoke autocomplete.
Ctrl+-	Go backwards in cursor history
Ctrl+Shift+-	Go forwards in cursor history
F12	Start/Cancel search for keyword in file(s).
Ctrl+F	Find text
Ctrl+R	Replace text

Notes

- In addition to these keys, you can assign (Ctrl+Alt+<n>) key combinations to macros that you define within the Source Code Editor

Application Patterns (Model + Code)

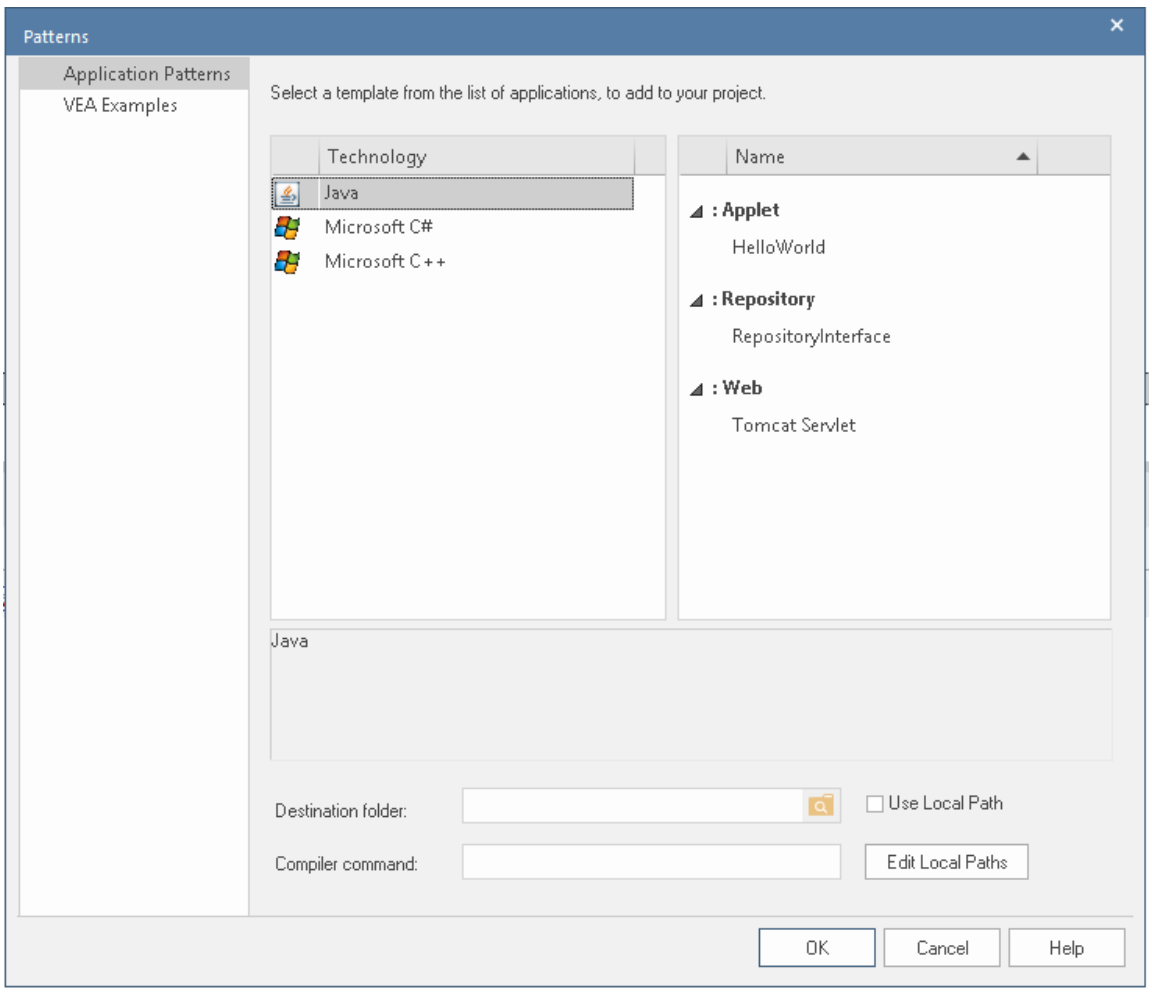
To get you going with a code based project as fast as possible, Enterprise Architect helps you to generate starter projects including model information, code and build scripts for one of several basic application types. Patterns include:

- MFC Windows applications
- Java programs
- ASP.NET web services

Access

Ribbon	Develop > Source Code > Create From Pattern > Application Patterns

Generate Models



Option	Action
Technology	Select the appropriate technology.
Name	Displays the Application Patterns available for the selected technology; select the required Pattern to import.
<description>	Displays a description of the selected Pattern.
Destination folder	Browse for and select the directory in which to load the source code for the

	application.
Use Local Path	Enable the selection of an existing local path to place the source code under; changes the 'Destination folder' field to a drop-down selection.
Compiler command	Displays the default compiler command path for the selected technology; you must either: • Confirm that the compiler can be found at this path, or • Edit the path to the compiler location
Edit Local Paths	Many application Patterns specify their compiler using a local path. The first time you use any Pattern you must click on this button to ensure the local path points to the correct location. The 'Local Paths' dialog displays.

Notes

- If required, you can publish custom application Patterns by adding files to the *AppPatterns* directory where Enterprise Architect is installed; top level directories are

listed as Technologies and can contain an icon file to customize the icon displayed for the technology
Directories below this are defined as groups in the Patterns list; the Patterns are identified by the presence of four files with a matching name: a zip file (.zip), XMI file (.xml), config file (.cfg) and optional icon (.ico)

- The config file supports these fields:
  - [provider], [language], [platform], [url], [description], [version] - all displayed in the <description> field
  - [xmirootpaths] - the root path of the source code in the exported XMI; this is replaced with the selected destination folder when the user applies the Application Pattern

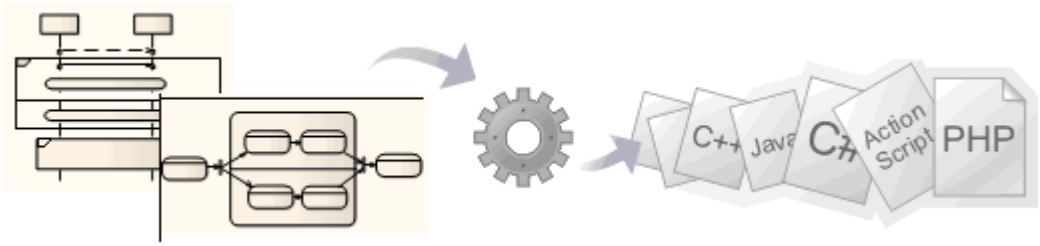
MDG Integration and Code Engineering

MDG Integration for Eclipse and MDG Integration for Visual Studio are products that help you to create and maintain your UML models directly inside these two popular Integrated Development Environments, using the Enterprise Architect Browser window. Models can be generated to source code using the rich and flexible template engine that gives the engineer complete control over how the code is generated. Existing source code can also be reverse engineered and synchronized with the UML models. With the integration installed the IDE will become a feature-rich modeling platform, saving time and effort and reducing the risk of error by linking Requirement Management, Architecture and Design to Source Code Engineering.

Rich and expressive documentation can be generated automatically into a wide range of formats including DOCX, PDF and HTML. The documentation can include diagrams of requirements, design and architecture as well as source code descriptions, putting the source code into context.

You can purchase MDG Integration for Eclipse™ and MDG Integration for Visual Studio™ or download Trial Editions, from the Sparx Systems web site.

Behavioral Models



Enterprise Architect's multi-featured system engineering capability can be used to generate code for software, system and hardware description languages directly from behavioral models, such as StateMachine, Sequence (Interaction) and Activity diagrams. The supported languages include C(OO), C++, C#, Java, VB.Net, VHDL, Verilog and SystemC.

Software code can be generated from StateMachine, Sequence and Activity diagrams, and hardware description languages from StateMachine diagrams (using the Legacy StateMachine templates).

Generate code from behavioral diagrams using the EAExample project

Step	Action
1	Open the EAExample.eap file by selecting the 'Start > Help > Help > Open the Example Model' ribbon option.

2	From the Browser window, select any of these Packages: Software Language Examples: • Example Model > Software Engineering > Java Model With Behaviors Generate the Account and Order classes • Example Model > Systems Engineering > Implementation Model > Software > C# Generate the DataProcessor Class • Example Model > Systems Engineering > SysML Example > Implementation Model > Software > C++ Generate the IO Class • Example Model > Systems Engineering > SysML Example > Implementation Model > Software > Java Generate the IO Class • Example Model > Systems Engineering > SysML Example > Implementation Model > Software > VBNet Generate the IO Class Hardware Language Examples: • Example Model > Systems Engineering > SysML Example: Portable Audio Player > Implementation Model > Hardware > SystemC Generate the PlayBack Class • Example Model > Systems Engineering > SysML

	Example: Portable Audio Player > Implementation Model > Hardware > VHDL Generate the PlayBack Class • Example Model > Systems Engineering > SysML Example: Portable Audio Player > Implementation Model > Hardware > Verilog Generate the PlayBack Class
3	When completed, press Ctrl+E to open the generated source code. You should see methods generated in the code.

Notes

- Software code generation from behavioral models is available in the Unified and Ultimate Editions of Enterprise Architect
- Hardware code generation from StateMachine models is available in the Unified and Ultimate Editions of Enterprise Architect
- For C(OO), on the 'C Specifications' page of the 'Manage Project Options' dialog, set the 'Object Oriented Support' option to True.
  See the *C Options - Model* Help topic.
- To be able to generate code from behavioral models, all

behavioral constructs should be contained within a Class; if the behavioral constructs refer to external elements outside the current Package, you must add an Import connector from the current Package to the Package containing the external elements

- Code synchronization is not supported for behavioral code.

Code Generation - Activity Diagrams

Code generation from Activity diagrams in a Class requires a validation phase, during which Enterprise Architect uses the system engineering graph optimizer to analyze the diagram and render it into various constructs from which code can be generated. Enterprise Architect also transforms the constructs into one of the various action types (if appropriate), similar to the Interaction diagram constructs.

Actions

Action	Description
Call Actions (Invocation Actions)	Used to invoke operations or behaviors in an Activity diagram; the two main variants of Call Actions supported in behavioral code generation are: • CallOperation Action - used to invoke operations, which can be within the same Class or in other Classes within the same Package; if referencing operations from other Classes within the same Package, you must have a target to which the request is passed • CallBehavior Action - used to invoke another Activity in an activity flow; the

	referenced Activity is expected to be within the same Class **Arguments** Call Actions can specify argument values corresponding to the parameters in the associated behavior or behavioral feature. You can add the arguments manually or create them automatically using the Synchronize button of the 'Arguments' dialog.
CreateObjectAction	Used to denote an object creation in the activity flow; you can set the result Pin of the CreateObjectAction as the object to be created, using the 'Assign Action Pins' dialog. The Classifier of the CreateObjectAction signifies the Classifier for which an instance is to be created.
DestroyObjectAction	Used to denote an object deletion in the activity flow; you can set the target Pin of the DestroyObjectAction as the object to be destroyed, using the 'Assign Action Pins' dialog.
Loops	Enterprise Architect's system engineering graph optimizer is also capable of analyzing and identifying loops; an

	identified loop is internally rendered as an Action Loop, which is translated by the EASL code generation macros to generate the required code. You can have a single loop, nested loops, and multiple levels of nested loops.
Conditional Statements	To model a conditional statement, you use Decision/Merge nodes. Alternatively, you can imply Decisions/Merges internally; the graph optimizer expects an associated Merge node for each Decision node, to facilitate efficient tracking of various branches and analysis of the code constructs within them.

Notes

- To be able to generate code from behavioral models, all behavioral constructs should be contained within a Class

Code Generation - Interaction Diagrams

During code generation from Interaction (Sequence) diagrams in a Class, Enterprise Architect applies its system engineering graph optimizer to transform the Class constructs into programmatic paradigms. Messages and Fragments are identified as two of the several action types based on their functionality, and Enterprise Architect uses the code generation templates to render their behavior accordingly.

Actions

Action	Description
Action Call	A Message that invokes an operation.
Action Create	A Message with Lifecycle = New.
Action Destroy	A Message with Lifecycle = Delete.
Action Loop	A Combined Fragment with Type = Alt.

Action If	A Combined Fragment with Type = loop.
Assign To	A Call Message with a valid target attribute set using the 'Assign To' field is rendered in the code as the target attribute of a Call Action.

Notes

- To be able to generate code from behavioral models, all behavioral constructs should be contained within a Class
- For an Interaction (Sequence) diagram, the behavioral code generation engine expects the Sequence diagram and all its associated messages and interaction fragments to be encapsulated within an Interaction element

Code Generation - StateMachines

A StateMachine illustrates how an object (represented by a Class) can change state, each change of state being a transition initiated by a trigger arising from an event, often under conditions or constraints defined as guards. As you model how the object changes state, you can generate and build (compile) code from it in the appropriate software language and execute the code, visualizing the execution via the Model Simulator.

It is also possible, in Enterprise Architect, to combine the StateMachines of separate but related objects to see how they interact (via Broadcast Events), and to quickly create and generate code from variants of the model. For example, you might model the behavior of:

- The rear off-side wheel of a vehicle in rear-wheel drive and front-wheel drive modes (one StateMachine)
- The steering wheel and all four drive wheels of a vehicle in 4-wheel drive mode (five StateMachines)
- The wheels of an off-road vehicle and of a sports car (two Artifacts, instances of a combination of StateMachines)

Of critical importance in generating and testing code for all of these options is the Executable StateMachine Artifact element. This acts as the container and code generation unit for your StateMachine models.

You do not use this method to generate code for Hardware Definition Languages, but you can also generate both HDL code and software code from StateMachines using the

generic Code Generation facilities in Enterprise Architect (see the *Generate Source Code* procedures).

Prerequisites

- Select 'Settings > Model > Options > Source Code Engineering' and, for the appropriate software coding language (Java, C, C# or ANSI C++), set the 'Use the new Statemachine Template' option to 'True'
- If working in C++, select 'Settings > Model > Options > Source Code Engineering > C++' and set the 'C++ Version' option to 'ANSI'

This code generation method does not apply to the Legacy StateMachine code generation templates developed prior to Enterprise Architect Release 11.0, nor to generating Hardware Definition Language code.

Access

Drag an Executable StateMachine Artifact from the 'Simulation' page of the Diagram Toolbox, onto your diagram. The 'Simulation' page of the Diagram Toolbox can be accessed using any of the methods outlined in this table.

Ribbon	Design > Diagram > Toolbox > Simulation

Keyboard Shortcuts	Ctrl+Shift+3 > Simulation
Other	You can display or hide the Diagram Toolbox by clicking on the » or « icons at the left-hand end of the Caption Bar at the top of the Diagram View.

Prepare your StateMachine diagram(s)

Step	Action
1	For each StateMachine you want to model, create a Class diagram.
2	From the 'Class' page of the Diagram Toolbox, drag the 'Class' icon onto your diagram and give the element an appropriate name.
3	Right-click on the Class element and select the 'New Child Diagram | StateMachine' context menu option. Give the StateMachine diagram an appropriate name.
4	Create the StateMachine model to reflect the

	appropriate transitions between States.

Set up the Executable StateMachine Artifact

Step	Action
1	Create a new Class diagram to contain the modeled StateMachine(s) from which you intend to generate code.
2	From the 'Simulation' page of the Diagram Toolbox, drag the 'Executable StateMachine' icon onto the diagram to create the Artifact element. Name the element and drag its borders out to enlarge it.
3	From the Browser window, drag the (first) Class element containing a StateMachine diagram onto the Artifact element on the diagram. The 'Paste <element name>' dialog displays. In the 'Drop as' field, click on the drop-down arrow and select the value 'Property'. (If the dialog does not display, press Ctrl as you drag the Class element from the Browser window.)
	Click on the OK button. The Class element is pasted

4	inside the Artifact as a Part.
5	Repeat steps 3 and 4 for any other Classes with StateMachines that you want to combine and generate code for. These might be: • Repeat 'drops' of the same Class and StateMachine, modeling parallel objects • Different Classes and StateMachines, modeling separate interacting objects
6	Right-click on the Artifact element and select the 'Properties > Properties' option, expand the 'Advanced' category and, in the 'Language' field, click on the drop-down arrow and set the code language to the same language as is defined for the Class elements. You can now drag this Executable StateMachine Artifact element from the Browser window onto the diagram any number of times, and modify the Parts to model variations of the system or process, or the same system or process with different programming languages.

Generate Code From Artifact

Ste	Action

p	
1	Click on the Executable StateMachine Artifact element and select the 'Simulate > Executable States > Statemachine > Generate' ribbon option. The 'Executable Statemachine Code Generation' dialog displays.
2	In the 'Project output directory' field, type or browse for the directory path under which to create the output files. During code generation, all existing files in this directory are deleted.
3	Select the Target System. If you are running on Windows select the 'Local' option. If you are working on Linux choose the 'Remote' option. The choice affects the scripts generated to support the Simulation.
4	In the 'Location of <compiler> installation directory' field, type or browse for the path of the compiler installation directory, to be automatically mapped to the local path (displayed to the left of the field). For each programming language, the paths might resemble these examples: • Java JAVA_HOME C:\Program Files (x86)\Java\jdk1.7.0_17

	• C/C++ VC_HOME C:\Program Files (x86)\Microsoft Visual Studio 9.0 • C# CS_HOME C:\Windows\Microsoft.NET\Framework\V3.5
5	Click on the Generate button. The code files are created appropriate to the programming language. The System Output window displays with an 'Executable StateMachine Output' tab, showing the progress and status of the generation. During code generation, an automatic validation function is executed to check for diagram or model errors against the UML constraints. Any errors are identified by error messages on the 'Executable StateMachine Output' tab. Double-click on an error message to display the modeling structure in which the error occurs, and correct the mistake before re-generating the code.
6	When the code generates without error, click on the Artifact element and select the 'Simulate > Executable States > Statemachine > Build' ribbon option to compile the code. The System Output window displays with a 'Build' tab, showing the progress and status of the compilation. Notice that the compilation includes configuration of the simulation operation.

Code Generation Macros

You can also use two macros in the code generation for StateMachines.

Macro Name	Description
SEND_EVENT	Send an event to a receiver (the Part). For example: %SEND_EVENT("event1", "Part1")%
BROADCAST_EVENT	Broadcast an event to all receivers. For example: %BROADCAST_EVENT("event2")%

Execute/Simulate Code From Artifact

Step	Action
1	Select the ribbon option 'Simulate > Dynamic

	Simulation > Simulator > Apply Workspace' to display the Simulation window and the Simulation Events window together Dock the two windows in a convenient area of the screen.
2	On the diagram or Browser window, click on the Artifact element and select the 'Simulate > Executable States > Statemachine > Run' ribbon option. The first StateMachine diagram in the series displays with the simulation of the process already started. In the Simulation window, the processing steps are indicated in this format: [03516677] Part1[Class1].Initial_367_TO_State4_142 Effect [03516683] Part1[Class1].StateMachine_State4 ENTRY [03516684] Part1[Class1].StateMachine_State4 DO [03518375] Blocked
3	Click on the appropriate Simulation window toolbar buttons to step through the simulation as you prefer. When the simulation finishes at the Exit or Terminate element, click on the Stop button in the Simulation window toolbar.

4	Where the trace shows Blocked, the simulation has reached a point where a Trigger event has to occur before processing can continue. On the Simulation Events window, in the 'Waiting Triggers' column, double-click on the appropriate Trigger. When the Trigger is fired, the simulation continues to the next pause point, Trigger or exit.

Notes

- If you are making small changes to an existing StateMachine model, you can combine the code generation, build and run operations by selecting the 'Simulate > Executable States > Statemachine > Generate, build and run' ribbon option
- You can also generate code in JavaScript

Legacy StateMachine Templates

Code generation operates using a set of generation templates. From Release 11.0 of Enterprise Architect, a different set of templates are available as the default for software code generation from a StateMachine diagram into Java, C, ANSI C++ or C# code. You can still use the original templates, as described here, for models developed in earlier releases of Enterprise Architect, if you do not want to upgrade them for the new template facilities.

Switch Between Legacy and Release 11 templates

Access

Display the 'Manage Project Options' dialog, then show the 'Language Specifications' page for your chosen language, using one of the methods outlined in this table. If necessary, expand the 'StateMachine Engineering (for current model)' grouping and set the 'Use the new StateMachine Template' option to True (to use the later templates) or False (to use the Legacy templates).

Ribbon	Settings > Model > Options > Source

	Code Engineering > [language name]

Legacy Template Transformations

A StateMachine in a Class internally generates a number of constructs in software languages to provide effective execution of the States' behaviors (do, entry and exit) and also to code the appropriate transition's effect when necessary.

Model Objects	Code Objects
Enumerations	• StateType - consists of an enumeration for each of the States contained within the StateMachine • TransitionType – consists of an enumeration for each transition that has a valid effect associated with it; for example, ProcessOrder_Delivered_to_ProcessOrder_Closed • CommandType – consists of an enumeration for each of the behavior types that a State can contain (Do, Entry, Exit)

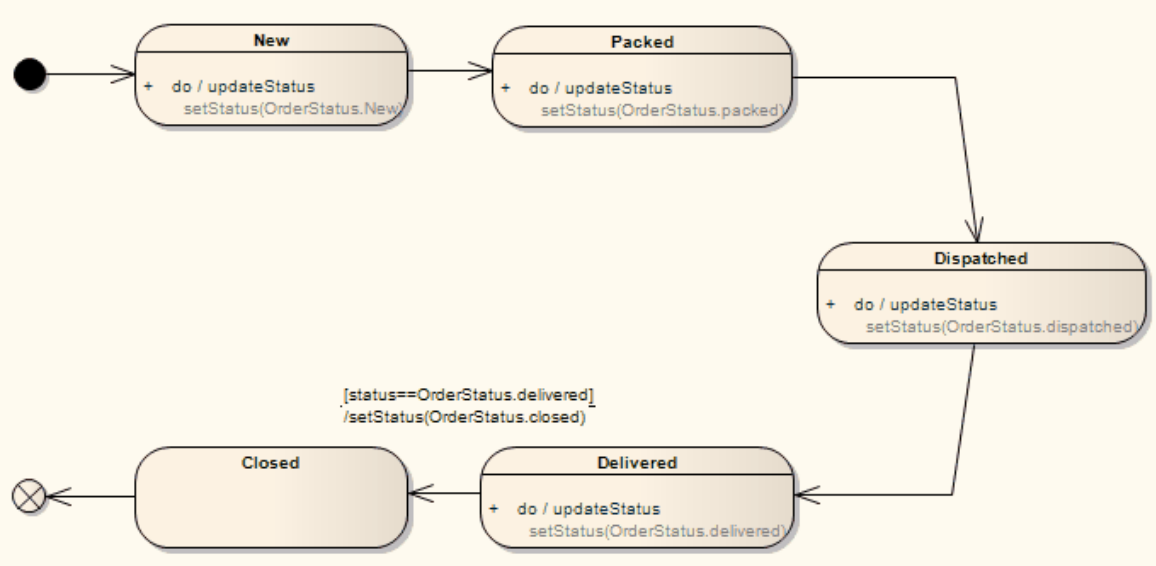
Attributes	• currState:StateType - a variable to hold the current State's information • nextState:StateType - a variable to hold the next State's information, set by each State's transitions accordingly • currTransition:TransitionType - a variable to hold the current transition information; this is set if the transition has a valid effect associated with it • transcend:Boolean - a flag used to advise if a transition is involved in transcending between different StateMachines (or Submachine states) • xx_history:StateType - a history variable for each StateMachine/Submachine State, to hold information about the last State from which the transition took place
Operations	• StatesProc - a States procedure, containing a map between a State's enumeration and its operation; it de-references the current State's information to invoke the respective State's function • TransitionsProc - a Transitions procedure, containing a map between the Transition's enumeration and its effect; it invokes the Transition's effect

	• <<State>> - an operation for each of the States contained within the StateMachine; this renders a State's behaviors based on the input CommandType, and also executes its transitions • initializeStateMachine - a function that initializes all the framework-related attributes • runStateMachine - a function that iterates through each State, and executes their behaviors and transitions accordingly

Notes

- To be able to generate code from behavioral models, all behavioral constructs should be contained within a Class

Java Code Generated From Legacy StateMachine Template




```
private enum StateType: int
{
    ProcessOrder_Delivered,
    ProcessOrder_Packed,
    ProcessOrder_Closed,
    ProcessOrder_Dispatched,
    ProcessOrder_New,
    ST_NOSTATE
}
private enum TransitionType: int
{
    ProcessOrder_Delivered_to_ProcessOrder_Closed,
    TT_NOTRANSITION
```

```
    }
    private enum CommandType
    {
        Do,
        Entry,
        Exit
    }
    private StateType currState;
    private StateType nextState;
    private TransitionType currTransition;
    private boolean transcend;
    private StateType ProcessOrder_history;
    private void processOrder_Delivered(CommandType
command)
    {
        switch(command)
        {
            case Do:
            {
                // Do Behaviors..
                setStatus(Delivered);
                // State's Transitions
                if((status==Delivered))
                {
                    nextState = StateType.ProcessOrder_Closed;
```

```
                    currTransition =
TransitionType.ProcessOrder_Delivered_to_ProcessOrder_
Closed;
                }
                break;
            }
            default:
            {
                break;
            }
        }
    }
    private void processOrder_Packed(CommandType
command)
    {
        switch(command)
        {
            case Do:
            {
                // Do Behaviors..
                setStatus(Packed);
                // State's Transitions
                nextState =
StateType.ProcessOrder_Dispatched;
                break;
```

```
            }
            default:
            {
                break;
            }
        }
    }
    private void processOrder_Closed(CommandType
command)
    {
        switch(command)
        {
            case Do:
            {
                // Do Behaviors..
                // State's Transitions
                break;
            }
            default:
            {
                break;
            }
        }
    }
    private void processOrder_Dispatched(CommandType
```

```
command)
    {
        switch(command)
        {
            case Do:
            {
                // Do Behaviors..
                setStatus(Dispatched);
                // State's Transitions
                nextState = StateType.ProcessOrder_Delivered;
                break;
            }
            default:
            {
                break;
            }
        }
    }
    private void processOrder_New(CommandType
command)
    {
        switch(command)
        {
            case Do:
            {
```

```
                // Do Behaviors..
                setStatus(new);
                // State's Transitions
                nextState = StateType.ProcessOrder_Packed;
                break;
            }
            default:
            {
                break;
            }
        }
    }
    private void StatesProc(StateType currState,
CommandType command)
    {
        switch(currState)
        {
            case ProcessOrder_Delivered:
            {
                processOrder_Delivered(command);
                break;
            }
            case ProcessOrder_Packed:
            {
                processOrder_Packed(command);
```

```
            break;
        }
        case ProcessOrder_Closed:
        {
            processOrder_Closed(command);
            break;
        }
        case ProcessOrder_Dispatched:
        {
            processOrder_Dispatched(command);
            break;
        }
        case ProcessOrder_New:
        {
            processOrder_New(command);
            break;
        }
        default:
        break;
    }
}
private void TransitionsProc(TransitionType transition)
{
    switch(transition)
    {
```

```
            case
ProcessOrder_Delivered_to_ProcessOrder_Closed:
            {
                setStatus(closed);
                break;
            }
            default:
            break;
        }
    }
    private void initalizeStateMachine()
    {
        currState = StateType.ProcessOrder_New;
        nextState = StateType.ST_NOSTATE;
        currTransition =
TransitionType.TT_NOTRANSITION;
    }
    private void runStateMachine()
    {
        while (true)
        {
            if (currState == StateType.ST_NOSTATE)
            {
                break;
            }
```

```
            currTransition =
TransitionType.TT_NOTRANSITION;
            StatesProc(currState, CommandType.Do);
            // then check if there is any valid transition
assigned after the do behavior
            if (nextState == StateType.ST_NOSTATE)
            {
                break;
            }
            if (currTransition !=
TransitionType.TT_NOTRANSITION)
            {
                TransitionsProc(currTransition);
            }
            if (currState != nextState)
            {
                StatesProc(currState, CommandType.Exit);
                StatesProc(nextState, CommandType.Entry);
                currState = nextState;
            }
        }
    }
```

StateMachine Modeling For HDLs

To efficiently generate Hardware Description Language (HDL) code from StateMachine models, apply the design practices described in this topic. Hardware Description Languages include VHDL, Verilog and SystemC.

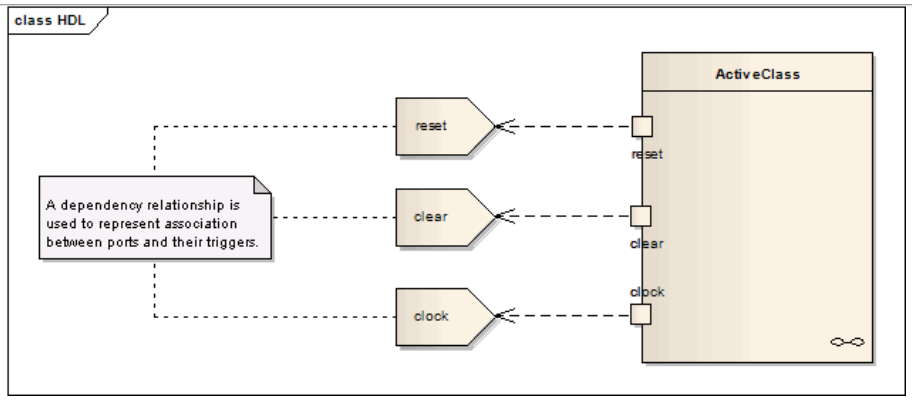
In an HDL StateMachine model, you might expect to:

- Designate Driving Triggers
- Establish Port–Trigger Mapping
- Add to Active State Logic

Operations

Operation	Description
Designate Driving Triggers	• A 'Change' Trigger is deemed to be an asynchronousTrigger if: - There is a transition from the actual SubMachine State (which encapsulates the actual logic) that it triggers, and - The target State of that transition has a self transition triggered by the same Trigger • Asynchronous Triggers should be modeled according to this pattern: - The Trigger should be of type

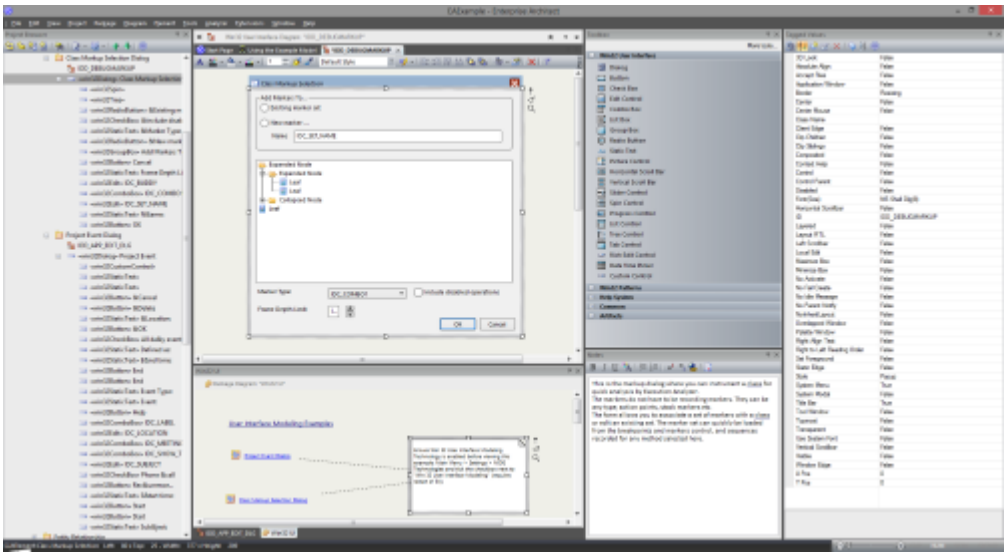
	Change (specification: True / False) - The active State (SubMachine State) should have a transition triggered by it - The target State of the triggered transition should have a self transition with the same Trigger • A Trigger of type 'Time', which triggers the transitions to the active state (SubMachine State), is deemed to be the Clock; the specification of this trigger should conform to the target language: - VHDL - rising_edge / falling_edge - Verilog - posedge / negedge - SystemC - positive / negative
Establish Port-Trigger Mapping	After successfully modeling the different operating modes of the component, and the Triggers associated with them, you must associate the Triggers with the component's Ports. A Dependency relationship from the Port to the associated Trigger is used to signify that association.

	
Active State Logic	Designating the driving Trigger and establishing the Port-Trigger mapping put in place the preliminaries required for efficiently interpreting the hardware components. We now model the actual StateMachine logic within the Active (SubMachine) State.

Notes

- To be able to generate code from behavioral models, all behavioral constructs should be contained within a Class
- The current code generation engine supports only one clock Trigger for a component

Win32 User Interface Dialogs



Using the MDG Win32 UI Technology, you can design user interface screens that render as Win32® controls. The user interface produced can be used in any resource definition script. Resource definition scripts, or RC files, are a Microsoft technology that - as for other code - can be compiled and the assets used by native desktop applications. User interface screens or dialogs can be created from scratch or reverse engineered. User interface models can also be forward engineered using the synchronize code function (F7). Interface modeling takes place on diagrams in the exact same fashion as you would work with any technology in Enterprise Architect. An interesting aspect of User Interface design in Enterprise Architect is that components can take an active role in the simulation of StateMachines and Activities, enabling a simulation to interact with users, much like a real program!

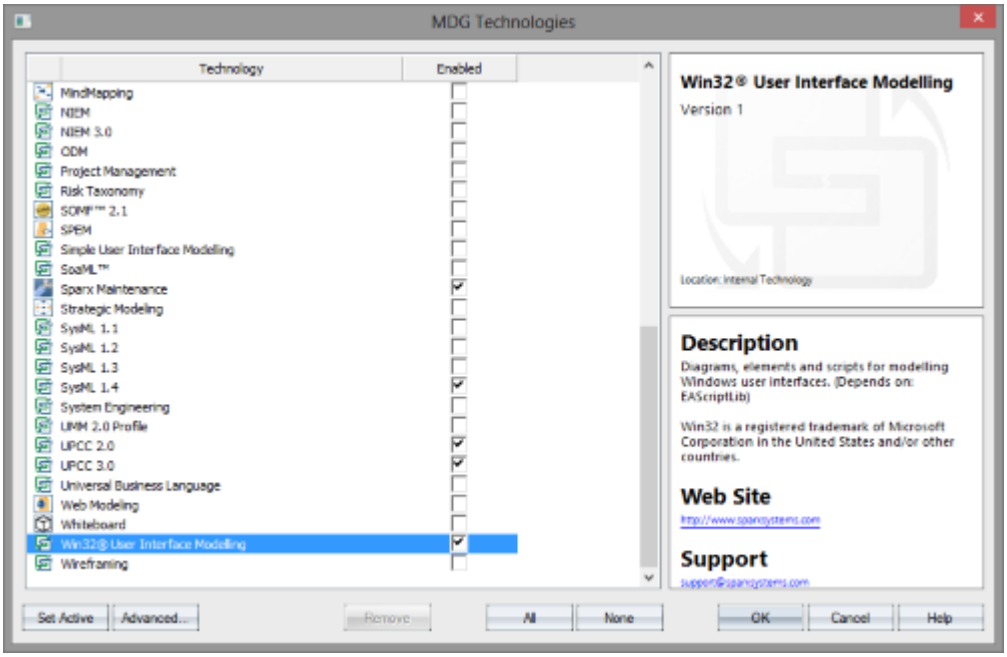
Access

Ribbon	Design > Diagram > Add > Type > User Interface Win32
Context Menu	Right-click on Package | Add Diagram > Type | User Interface Win32
Other	Browser window caption bar menu | New Diagram | User Interface Win32

Support

The MDG Win32® User Interface Technology is available in the Enterprise Architect Professional, Corporate, Unified and Ultimate editions

Enabling Win32 User Interface Technology



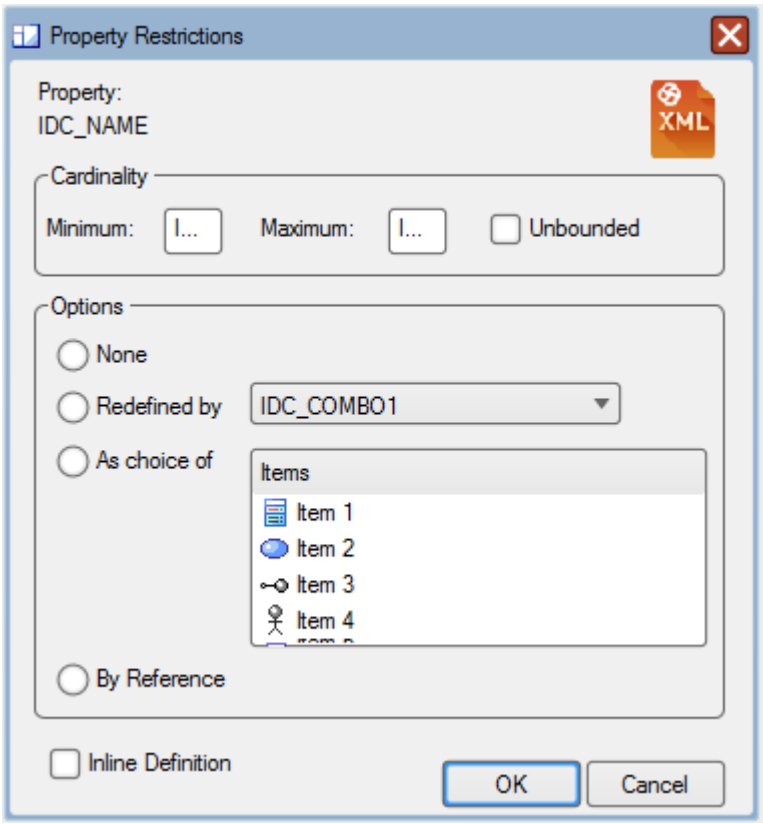
The Win32® UI Technology in Enterprise Architect is enabled or disabled using the 'MDG Technologies' dialog (select the 'Specialize > Technologies > Manage-Tech' ribbon option).

Default technology

You can set the MDG Win32® UI Technology as the active default technology to access the Toolbox pages directly.

Modeling UI Dialogs

The Win32 User Interface MDG Technology provides the tools to help you design a user interface that closely emulates the visual style and available options for Windows dialogs.



Win32 Dialog

These user interface components are supported, each matching the equivalent-named RC resource.

Component	Details
win32Dialog	The equivalent of the RC format

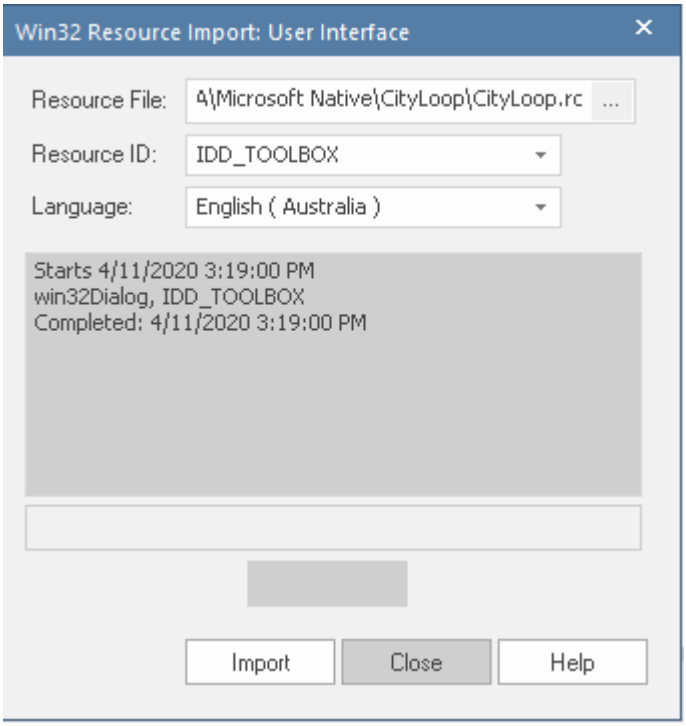
	DIALOG and DIALOGEX resources.
win32StaticText	The equivalent of the RC format LTEXT, RTEXT, CTEXT resources.
win32Edit	The equivalent of the RC format EDITTEXT resource.
win32Button	The equivalent of the RC format BUTTON, DEFPUSHBUTTON and other resources.
win32CheckBox	The equivalent of the RC format CHECKBOX resource.
win32ScrollBarH	The equivalent of the RC format SCROLLBAR resource with SBS_HORZ style
win32ScrollBarV	The equivalent of the RC format SCROLLBAR resource with SBS_VERT style.
win32GroupBox	The equivalent of the RC format GROUPBOX resource.
win32ComboBox	The equivalent of the RC format COMBOBOX resource. Note: When you initially drag the 'Combo

	Box' icon - of type 'Drop Down' or 'Drop Down List' - onto a diagram, the middle 'tracking handle' on each side of the element is white, indicating that you can only adjust the width of the element. To adjust the height of the element as well as the width, click on the drop-down arrow part of the image; the middle 'tracking handle' on the bottom edge is now white, indicating that you can drag the base down to set the virtual height (the height of the element when it is expanded to show all possible values in the drop-down list).
win32ListBox	The equivalent of the RC format LISTBOX resource.
win32RadioButton	The equivalent of the RC format RADIOBUTTON resource.
win32TabPane	The equivalent of the RC format TABPANE resource.
win32Picture	The equivalent of the RC format STATIC resource with SS_BITMAP style. The control can render an image when applied from your model. An image can be applied by selecting it first and

	pressing Ctrl+Shift+W to display the Image Manager. Afterwards, you might need to change the value of the resource ID in the appropriate Tagged Value.
win32CustomControl	The equivalent of the RC format CONTROL resource.

Import Single Dialog from RC File

You can quickly import a single dialog by name.



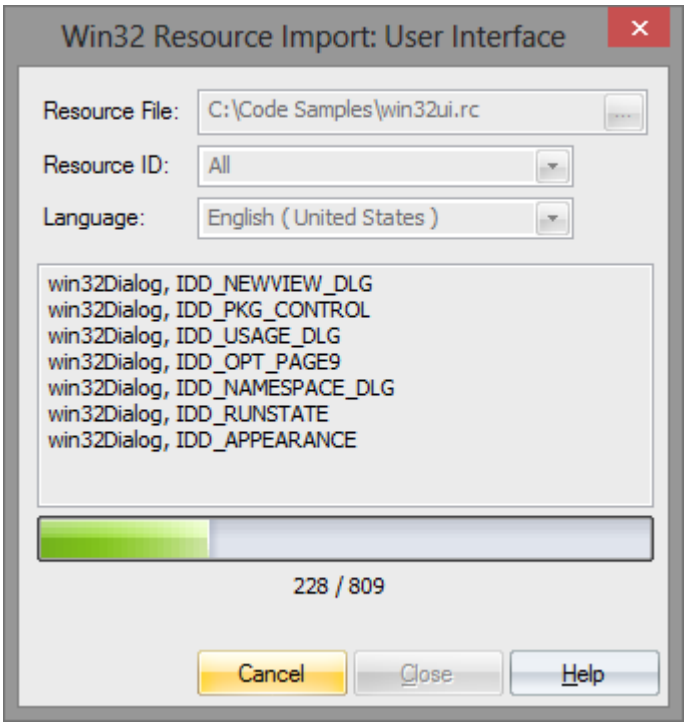
Access

In the Browser window, click on the target Package.

Ribbon	Develop > Source Code > Files > Import Resource Script

Import All Dialogs from RC File

All dialogs in a single RC file can be imported into your model. This image was captured one minute into the import, at which time over 200 large dialog definitions had been imported.

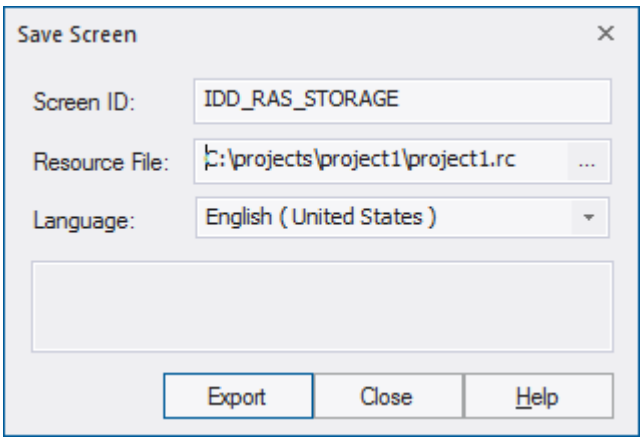


Access

Ribbon	Develop > Source Code > Files > Import Resource Script

Export Dialog to RC File

Once a screen design is modified or a new one created, you might want to get it back to the RC file you use to build your application, so that you can see how it looks with real data. Begin by selecting the Win32Dialog element in the Browser window, then use the ribbon to perform the synchronization.



Access

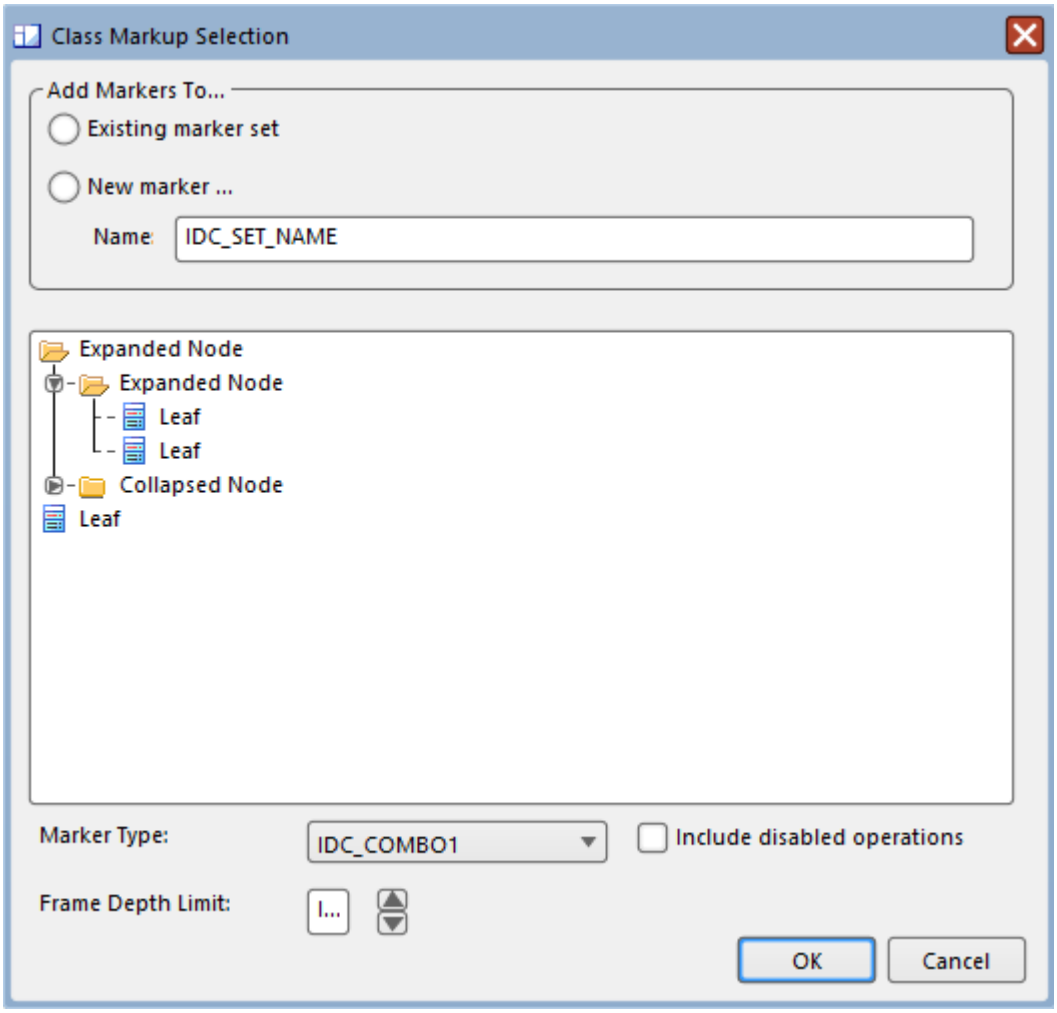
Click on the win32Dialog element.

Ribbon	Develop > Source Code > Generate > Generate Single Element
Keyboard Shortcuts	F11

Design a New Dialog

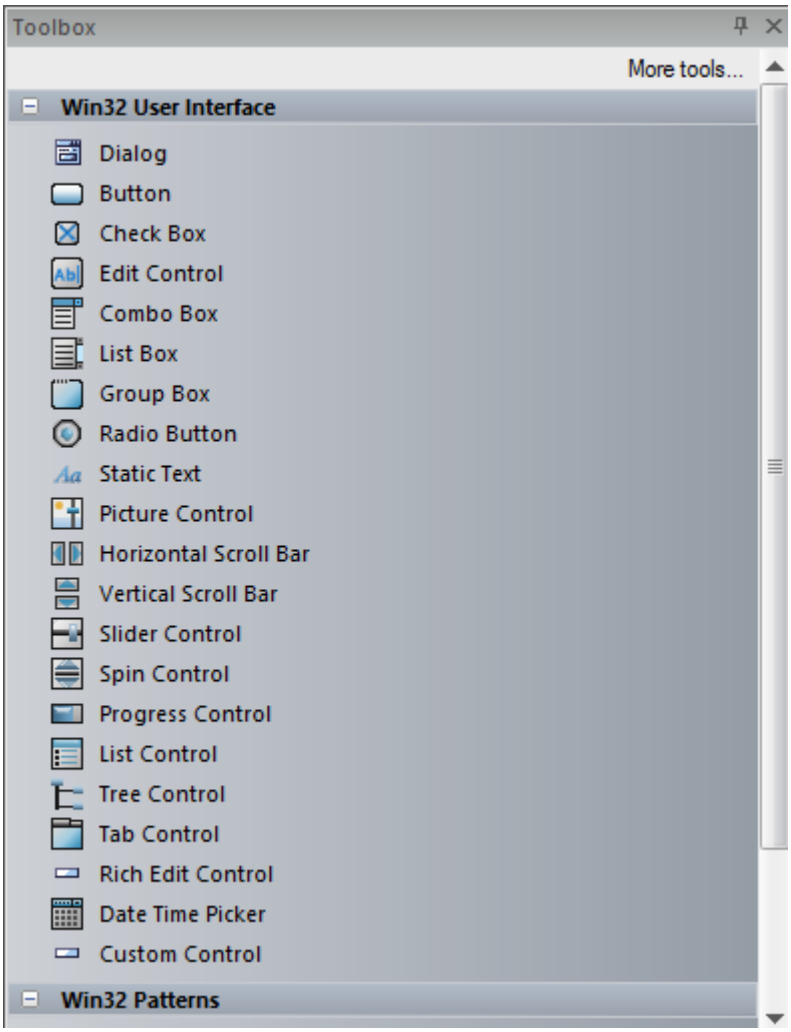
Creating a new Win32 dialog is easy and mostly visual. You will probably need a workspace that shows:

- The new diagram (select the 'Design > Diagram > Add > User Interface - Win32 > User Interface - Win32' ribbon path)
- The Win32 User Interface Toolbox (select the 'Design > Diagram > Tooolbox' ribbon option) and
- The Tagged Values tab of the Properties window



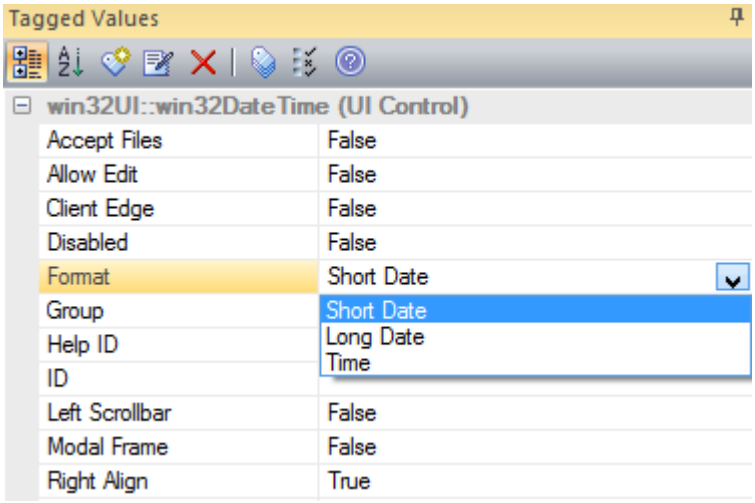
The UI Toolbox

All of the common RC elements can be found on the UI toolbox



The Tags Tab

This tab is provided on the Properties window and 'Properties' dialog for an object, and is where all the properties of a control can be viewed and edited.



Using the Picture Control

Images from your model (see *Image Manager*) can be applied by selecting the control on the dialog and pressing Ctrl+Shift+W. You might have to enter the value of the resource ID in the appropriate Tagged Value.

Note

- You can copy and paste dialog Packages

Gang of Four (GoF) Patterns

A Design Pattern is a template for solving commonly recurring design problems; it consists of a series of elements and connectors that can be reused in a new context. The advantage of using Patterns is that they have been tested and refined in a number of contexts and so are typically robust solutions to common problems. Enterprise Architect provides the Gang of Four Patterns as an MDG Technology that can be loaded into the current repository.

The Gang of Four (Gof) Patterns are a group of twenty three Design Patterns originally published in a seminal book entitled *Design Patterns: Elements of Reusable Object-Oriented Software*; the term 'Gang of Four' refers to the four authors. Enterprise Architect displays these Patterns in its Pattern engine, helping you to visualize the elements of the Pattern and adjust the Pattern to the context of your software design problem.

GoF Patterns in Enterprise Architect

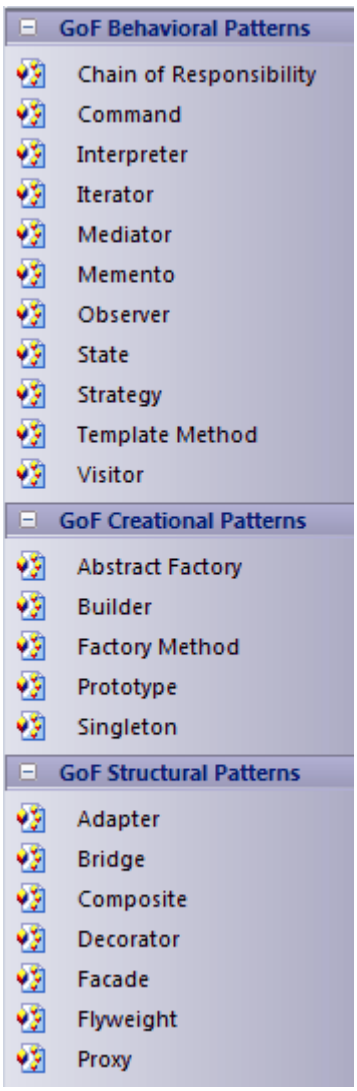
Features	Description
GoF Pattern Facilities	The GoF Patterns are provided in the form of: • GoF Behavioral Patterns, GoF Creational Patterns and GoF Structural

Patterns pages in the Toolbox

- Gang of Four Pattern entries in the Toolbox Shortcut Menu

GoF Pattern Toolbox Pages

You can access the 'GoF Pattern' pages of the Toolbox by clicking on  to display the 'Find Toolbox Item' dialog and specifying 'GoF Patterns'; these icons are available:



When you drag one of the Pattern

	elements onto a new diagram, the 'Add Pattern GoF <pattern group><pattern type>' dialog displays; if necessary, modify the action and/or default for the component elements, then click on the OK button to create a diagram based on the Pattern.

ICONIX

The ICONIX process is a proprietary software development methodology based on UML. The process is Use Case driven and uses UML-based diagrams to define four milestones. The main feature of the process is a concept called robustness modeling, based on the early work of Ivar Jacobson, which helps bridge the gap between analysis and design.

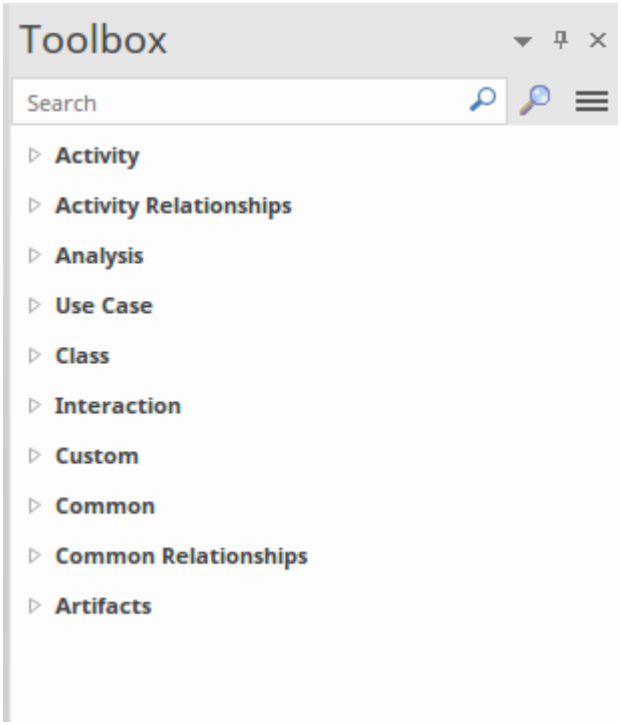
This text is derived from the ICONIX entry in the online Wikipedia:

'The ICONIX Process is a minimalist, streamlined approach to Use Case driven UML modeling that uses a core subset of UML diagrams and techniques to provide thorough coverage of object-oriented analysis and design. Its main activity is robustness analysis, a method for bridging the gap between analysis and design. Robustness analysis reduces the ambiguity in use case descriptions, by ensuring that they are written in the context of an accompanying domain model. This process makes the use cases much easier to design, test and estimate.'

The ICONIX Process was developed by Doug Rosenberg; for more information on ICONIX, refer to ICONIX Software Engineering Inc.

Aspects

Aspect	Detail
ICONIX in Enterprise Architect	Enterprise Architect enables you to develop models under ICONIX quickly and simply, through use of an MDG Technology integrated with the Enterprise Architect installer. The ICONIX facilities are provided in the form of: • A set of ICONIX pages in the Toolbox • ICONIX element and relationship entries in the 'Toolbox Shortcut' menu and Quick Linker To further help you develop and manage a project under ICONIX, Enterprise Architect also provides a white paper on the ICONIX Roadmap.
ICONIX Toolbox Pages	Within the Toolbox, Enterprise Architect provides ICONIX versions of the pages for UML Analysis, Use Case, Class, Interaction (Sequence), Activity and Custom diagrams (which often form the basis for Robustness diagrams). Compared to the standard Toolbox pages, these have slightly different element and relationship sets; you can access them by either:

	• Specifying 'ICONIX' in the 'Find Toolbox Item' dialog and selecting a specific Toolbox page • Selecting the 'ICONIX' option in the drop-down field of the Default Tools toolbar, which adds all six pages to the Toolbox; all pages are closed up 

Configuration Settings

You can set the default code options such as the editors for each of the programming languages available for Enterprise Architect and special options for how source code is generated or reverse engineered. These options are defined according to whether they apply to:

- All users of the current model, set on the 'Manage Project Options' dialog, or
- All models that you access (other users can define their own settings that apply to the same models), set on the 'Preferences' dialog

You can also:

- For each programming language used in the model, for all users working on the model, define Collection Classes for generating code from Association connectors where the target role has a multiplicity setting greater than 1
- Define a local path for yourself, using the 'Local Path' dialog; these settings apply to all Enterprise Architect models that you access
- Define language macros within the model, which are useful in reverse engineering and can be exported from and imported to the model

Source Code Engineering Options

The 'Source Code Engineering' options apply to the languages in which you generate code from Enterprise Architect. They are divided into Model-specific options and User-specific options, as explained here.

Model-Specific Options

These options are defined on the 'Manage Project Options' dialog.

Access

Ribbon	Settings > Model > Options > Source Code Engineering

Types of Option

Option Type	Detail
Source Code	You can define a number of settings for

Generation Options	generating code in the model, such as the default language to generate code in and the Unicode character set for code generation.
Options - Object Lifetimes	You can configure various options concerning Object Lifetimes.
Code Language Options	For each of the code languages that Enterprise Architect supports, you can define the model-specific options and set any Collection Classes required.

User-Specific Options

These options are defined on the 'Preferences' dialog.

Access

On the 'Preferences' dialog, click on 'Source Code Engineering' in the left-hand list.

Ribbon	Start > Appearance > Preferences >

	Preferences
Keyboard Shortcuts	Ctrl+F9

Types of Option

Option Type	Detail
Source Code Generation Options	You can define a number of settings for generating code in any model that you access under the same user ID.
Code Editors	These are options for accessing and configuring the source code editor.
Attributes/Operations	Use these options for configuring attributes and operations.
Code Language Options	For each of the code languages that Enterprise Architect supports, you can define the user-specific options that apply to any model that you access under your user ID.

Code Generation Options

When you generate code for your model, you can set certain options. These include:

- The default language
- Whether to generate methods for implemented interfaces
- The Unicode options for code generation

Access

Ribbon	Settings > Model > Options > Source Code Engineering

Configure code generation options

Option	Action
Always synchronize with existing file (recommended)	Select the radio button to synchronize imported code with an existing file.

Replace (overwrite) existing source file	Select the radio button to overwrite the existing source file with imported code.
Component Types	Click on this button to open the 'Import component types' dialog, to set up the importation of component types.
Default Language for Code Generation	Click on the drop-down arrow and select the default language for code generation.
DDL Name Templates	Click on the  button to define the template names for Primary Key, Unique Constraint, Foreign Key and Foreign Key Index Name templates.
Default name for associated attrib	Type in a default name to be generated from imported attributes.
Generate methods for implemented interfaces	Select the checkbox to indicate that methods are generated for implemented interfaces.

Code page for source editing	Click on the drop-down arrow and select the appropriate Unicode character embedding format to apply.

Notes

- It is worthwhile to configure these settings, as they serve as the defaults for all Classes in the model; you can override most of these on a per-Class basis using the custom settings (from the 'Code Generation' dialog)

Import Component Types

Using the 'Import Component Types' dialog you can configure what elements you want to be created for files of any extension found while importing a source code directory.

Access

Ribbon	Settings > Model > Options > Source Code Engineering: Component Types

Define Import Component Types

Option	Action
Extension	Type in the extension name for a component type.
Type	Click on the drop-down arrow and select the component type.
Stereotype	Type in any stereotype name that further

	identifies a component of this type.
Component List	Lists the currently-defined component types.
Save	Click on this button to saves the component definition and add it to the component list.
New	Click on this button to clear the dialog fields so that you can define a new component type.
Delete	Click on this button to delete the selected component type from the component list.

Notes

- You can transport these import component types between models, using the 'Settings > Model > Transfer > Export Reference Data' and 'Import Reference Data' ribbon options

Source Code Options

You can set a wide range of options for generating code in the models you work with. These include:

- How to format the generated code
- How to respond to certain events during code generation
- Whether to generate a diagram from the code

Access

On the 'Preferences' dialog, select the 'Source Code Engineering' option

Ribbon	Start > Application > Preferences > Preferences
Keyboard Shortcuts	Ctrl+F9

Configure code generation options

Field	Action
Wrap long	Type in the number of characters to allow

comment lines at	in a comment line before wrapping the text to the next line.
Auto Layout Diagram on Import	Click on the drop-down arrow and select if and when a diagram is automatically generated on code import.
Output files use both CR & LF	Select the checkbox to include carriage returns and line feeds; set this option according to what operating system is currently in use, as code might not render correctly.
Prompt when synchronizing (reversing)	Select the checkbox to display a prompt when synchronization occurs.
Remove hard breaks from comments on import	Select the checkbox to remove hard breaks from commented sections on importation.
Auto generate role names when creating code	Select the checkbox to generate role names when creating code.
Do not generate	Select the checkbox to prevent generation of members if the Association direction is

members where association direction is 'Unspecified'	unspecified.
Create dependencies for operation returns and parameter types	Select the checkbox to generate dependencies for operation returns and parameter types.
Comments: Generate	Select the checkbox to generate comments.
Comments: Reverse	Select the checkbox to generate reverse comments.
Remove prefixes when generating Get/Set properties	Type in the prefixes, separated by semi-colons, used in your variable naming conventions, to be removed in the variables' corresponding get/set functions.
Treat as suffixes	Select the checkbox to use the prefixes defined in the 'Remove prefixes when generating Get/Set properties' field as

	suffixes.
Capitalized Attribute Name for Properties	Select the checkbox to capitalize attribute names for properties.
Use 'Is' for Boolean property Get()	Select the checkbox to use the Is keyword for the Boolean property Get().

Notes

- It is worthwhile to configure these settings, as they serve as the defaults for all Classes in the model; you can override most of these on a per-Class basis using the custom settings (from the 'Code Generation' dialog)

Options - Code Editors

You access the source code editor options via the 'DDL' page of the 'Preferences' dialog. On this page you can configure options for Enterprise Architect's internal editor, as well as the default editor for DDL scripts. You can configure external editors for code languages on each language options page.

Access

On the 'Preferences' dialog, select the 'Source Code Engineering > Code Editors' option.

Ribbon	Start > Appearance > Preferences > Preferences
Keyboard Shortcuts	Ctrl+F9

Options

Option	Action

DDL Editor	Defaults to blank, indicating that the Enterprise Architect code editor is the DDL editor in use. You can select a different default editor if necessary; click on the ... button to browse for and select the required DDL editor. The editor name then displays in the 'DDL Editor' field.
Default Database	Click on the drop-down arrow and select the default database to be used.
MySQL Storage	Click on the drop-down arrow and select the MySQL storage engine to be used.
Use inbuilt editor if no external editor set	Select the checkbox to use the inbuilt editor for code in any language if no external editor is defined for that language in the user-specific options.
Show Line Numbers	Select the checkbox to display line numbers in the editor.
Show Structure Tree	Select the checkbox to show a tree with the results of parsing the open file (if the file is parsed successfully).
Automatically Reverse	If you select this checkbox, pressing Ctrl+S to save in the source code editor

Engineer on File Save	automatically reverse engineers the code in the same way as the Save Source and Re-Synchronize Class button does.
Don't parse files larger than	Click on the drop-down arrow and select the upper limit on file size for parsing. Setting this option prevents performance decrease due to parsing very large files.
Syntax Highlighting Options	Click on the  button to display the 'Editor Language Properties' dialog, in which you can set both global and language-specific editor language properties.
Configure Enterprise Architect File Associations	Click on the  button to display the 'Set Associations for a Program' dialog, and select the file extensions for files that you want to open through the Enterprise Architect Document Handler.

Editor Language Properties

Using the 'Editor Language Properties' dialog, you can specify syntax highlighting properties for any of the programming languages that Enterprise Architect supports at installation.

Access

In the 'Preferences' dialog, select the 'Source Code Engineering | Code Editors' option and click on the ... button next to 'Syntax Highlighting Options'.

Ribbon	Start> Appearance > Preferences > Preferences, select 'Source Code Engineering | Code Editors' option > click on the ... button next to 'Syntax Highlighting Options'
Other	In the Code Editor window, click on the toolbar icon  | Syntax Highlighting Options

Options

Panel	Description
Language Panel	The panel on the left of the dialog lists the languages for which you can set properties. At the top of the list are three non-language options: • (Dark Theme) - assigns a dark background to the property fields and to the code panel in the code editor screen (you can apply a different color to specific properties) • (Light Theme) - assigns a pale background to the property fields and to the code panel in the code editor screen (you can apply a different color to specific properties) You can also set the background themes on the 'Application Look' dialog • (Global) provides properties that you can set for all programming languages; however, you can reset a global property to a different value for a particular language, in the properties specifically for that language Resetting a global property for one language does not affect that property's value for the other languages

	Click on the required language in the list, to display the properties for that language: • Properties shown in bold indicate that this is the highest level at which this property can be defined (for most language options other than 'Global', this is effectively the only point at which the property is defined) • Properties shown in normal font are generally the global properties that you can reset just for the current language
Properties Panel	Scroll through the property categories and individual properties for the language. You can collapse and expand categories as necessary, using the expansion box next to the category name (⊟). When you click on a property name, an explanation of that property displays in the panel at the bottom right of the dialog. To define a property, click on the value field following the property name; depending on the type of property, either the field is enabled for direct editing or a drop-down arrow or ... button displays (as described for the 'Tags' tab of the Properties window) so that you can select the values to define the property.

	Select or type in the required values. Use the Toolbar icons to: • Save your changes to the properties • Reset all properties fields to the default settings shipped with Enterprise Architect • Reset the current style field to the default setting (not enabled for non-style fields)
Assign Keys to Macros	In the 'Macros' category of the properties, you can assign (Ctrl+Alt+<n>) keystroke combinations to coding macros that you have created yourself in the 'Source Code Viewer'. When you click on the Browse button in a selected 'Macro' field, the 'Open Macro' dialog displays; this dialog lists the existing macros and, if a key combination has been assigned to a macro, what that key combination is. Click on the name of the macro and on the Open button to assign the selected keys to the macro.

Notes

- You cannot currently set properties for any additional languages you include through an MDG Technology
- You can resize this dialog, if required

Options - Object Lifetimes

You can use these options to configure various Object Lifetime settings such as:

- Defining constructor details when generating code
- Specifying whether to create a copy constructor
- Defining Destructor details

Access

Ribbon	Settings > Model > Options > Source Code Engineering > Object Lifetimes

Options

Option	Action
Constructor	If necessary, select the checkboxes to specify that a constructor is generated and (for C++) that the constructor is in-line. Click on the drop-down arrow and select the appropriate visibility of the default

	constructor - Private, Protected or Public.
Copy Constructor	If necessary, select the checkboxes to specify that a copy constructor is generated and (for C++) that the copy constructor is in-line. Click on the drop-down arrow and select the appropriate visibility of the default copy constructor - Private, Protected or Public.
Destructor	If necessary, select the checkboxes to specify that a destructor is generated and (for C++) that the destructor is in-line and/or virtual. Click on the drop-down arrow and select the appropriate visibility of the default destructor - Private, Protected or Public.

Options - Attribute/Operations

Your use of attributes and operations can be configured in a number of ways. You can set options to:

- Delete model attributes not included in the code during reverse synchronization
- Delete model methods not included in the code during reverse synchronization
- Delete code from features contained in the model during forward synchronization
- Delete model associations and aggregations that correspond to attributes not included in the code during reverse synchronization
- Define whether or not the bodies of methods are included and saved in the model when reverse engineering
- Create features in quick succession, clearing the Properties window when you click on 'Save' so that you can enter another feature name

You configure these options on the 'Attribute/Operations' page of the 'Preferences' dialog.

Access

On the 'Preferences' dialog, select the 'Source Code Engineering > Attribute/Operations' option.

Ribbon	Start > Appearance > Preferences > Preferences
Keyboard Shortcuts	Ctrl+F9

Options

Field	Action
On reverse synch, delete model attributes not in code	Select the checkbox to indicate that on reverse synchronization, attributes in the model that are not included within code are automatically removed from the model.
On reverse synch, delete model associations not in code	Select the checkbox to indicate that on reverse synchronization, associations in the model that are not included within code are automatically removed from the model.
On reverse synch, delete model methods not	Select the checkbox to indicate that on reverse synchronization, methods in the model that are not included within code are automatically removed from the

in code	model.
Include method bodies in model when reverse engineering	Select the checkbox to indicate that on reverse engineering code, method bodies in the code are included within your model.
After Save, re-select edited item	Select the checkbox to indicate that after saving an attribute or operation, the properties definition continues to display the details of the selected feature. If deselected, indicates that the fields of the properties definition will clear so that you can enter another attribute or operation name and details immediately.
On forward synch, prompt to delete code features not in model	Select the checkbox to indicate that, during forward synchronization, the 'Synchronize Element <package name>.<element name>' dialog displays, so that you can either ignore, reassign or delete features in the code that are not in the model.

Modeling Conventions

The synchronization between UML models and programming code is achieved using a set of modeling conventions (mappings) between UML constructs and programming code syntax. The Software Engineer is advised to become familiar with these conventions in order to work with the code generation process for the programming languages they intend to target. There are a range of constructs used, including elements, features, connectors, connector ends, stereotypes and Tagged Values. The newcomer will require a little time to become familiar with these conventions but after a short time they will be translating between programming code and UML constructs without effort.

Supported Languages

Language
Action Script
Ada 2012 (Unified and Ultimate Editions)

C
C#
C++
Delphi
Java
PHP
Python
SystemC (Unified and Ultimate Editions)
Verilog (Unified and Ultimate Editions)
VHDL (Unified and Ultimate Editions)
Visual Basic
Visual Basic .NET

Notes

Enterprise Architect incorporates a number of visibility indicators or scope values for its supported languages; these include, for:

- All languages - Public (+), Protected (#) and Private (-)
- Java - Package (~)
- Delphi - Published (^)
- C# - Internal (~), Protected Internal (^)
- ActionScript - Internal (~)
- VB.NET - Friend (~), Protected Friend (^)
- PHP - Package (~)
- Python - Package (~)
- C - Package (~)
- C++ - Package (~)

ActionScript Conventions

Enterprise Architect supports round trip engineering of ActionScript 2 and 3, where these conventions are used.

Stereotypes

Stereotype	Applies To
literal	Operation Corresponds To: A literal method referred to by a variable.
property get	Operation Corresponds To: A 'read' property.
property set	Operation Corresponds To: A 'write' property.

Tagged Values

Tag	Applies To

attribute_name	Operation with stereotype property get or property set Corresponds To: The name of the variable behind this property.
dynamic	Class or Interface Corresponds To: The 'dynamic' keyword.
final	ActionScript 3: Operation Corresponds To: The 'final' keyword.
intrinsic	ActionScript 2: Class Corresponds To: The 'intrinsic' keyword.
namespace	ActionScript 3: Class, Interface, Attribute, Operation Corresponds To: The namespace of the current element.
override	ActionScript 3: Operation Corresponds To: The 'override' keyword.
prototype	ActionScript 3: Attribute Corresponds To: The 'prototype' keyword.
rest	ActionScript 3: Parameter

	Corresponds To: The rest parameter (...)

Common Conventions

- Package qualifiers (ActionScript 2) and Packages (ActionScript 3) are generated when the current Package is not a namespace root
- An unspecified type is modeled as 'var' or an empty 'Type' field

ActionScript 3 Conventions

- The Is Leaf property of a Class corresponds to the sealed keyword
- If a namespace tag is specified it overrides the Scope that is specified

Ada 2012 Conventions

Enterprise Architect supports round trip engineering of Ada 2012, where these conventions are used.

Stereotypes

Stereotype	Applies To
adaPackage	Class Corresponds To: A Package specification in Ada 2012 without a tagged record.
adaProcedure	Class Corresponds To: A procedure specification in Ada 2012.
delegate	Operation Corresponds To: Access to a subprogram.
enumeration	Inner Class Corresponds To: An enumerated type.
struct	Inner Class Corresponds To: A record definition.

typedef	Inner Class Corresponds To: A type definition, subtype definition, access type definition, renaming.

Tagged Values

Tag	Applies To
Aspect	Inner Class with stereotype typedef Operation Corresponds to: Aspect specification (Precondition and Postcondition of Subprogram type 'invariant', subtype 'predicate').
InstantiatedUnitType	Inner Class with stereotype typedef Corresponds To: The instantiated unit's type (Package / Procedure / Function).
IsAccess	Parameter Corresponds To: Determination of whether the parameter is an access variable.

IsAliased	Function parameter Corresponds to: Aliased function parameter.
Discriminant	Inner Class with stereotype typedef Corresponds To: The type's discriminant.
PartType	Inner Class with stereotype typedef Corresponds To: The part type ('renames' or 'new').
Type	Inner Class with stereotype typedef Corresponds To: If 'Value' = 'SubType', set 'subtype' If 'Value' = 'Access', set 'access type'.

Other Conventions

- Appropriate type of source files: Ada specification file, .ads
- Ada 2012 imports Packages defined as either <<adaPackage>> Class or Class, based on the settings in the Ada 2012 options
- A Package in the Ada specification file is imported as a Class if it contains a Tagged Record, the name of which is

governed by the options 'Use Class Name for Tagged Record' and 'Alternate Tagged Record Name'; all attributes defined in that Tagged Record are absorbed as the Class's attributes

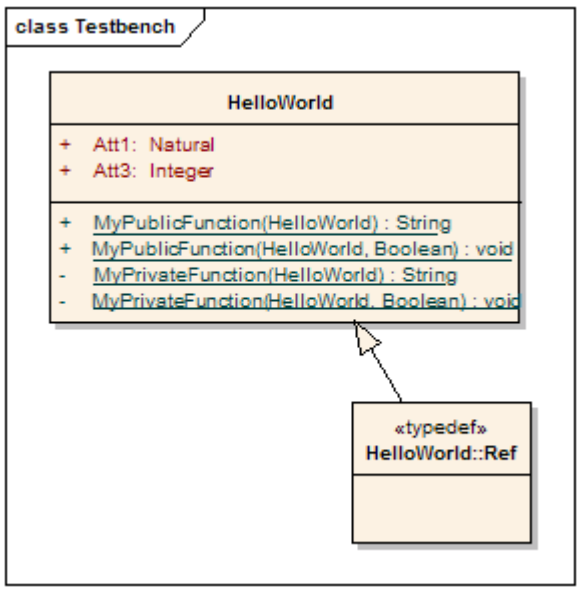
- A procedure / function in an Ada specification file is considered as the Class's member function if its first parameter satisfies the conditions specified in the options 'Ref Param Style', 'Ignore Reference parameter name' and 'Ref parameter name'
- The option 'Define Reference for Tagged Record', if enabled, creates a reference type for the Class, the name of which is determined by the option 'Reference Type Name'; for example:

```
HelloWorld.ads
package HelloWorld is
    type HelloWorld is tagged record
        Att1: Natural;
        Att3: Integer;
    end record;

    -- Public Functions
    function MyPublicFunction (P: HelloWorld)
return String;
    procedure MyPublicFunction (P1: in out
HelloWorld; AFlag: Boolean);
private
    -- Private Functions
```

function MyPrivateFunction (P: HelloWorld) return String;

procedure MyPrivateFunction (P1: in out HelloWorld; AFlag: Boolean);

end HelloWorld;



Notes

- Ada 2012 support is available in the Unified and Ultimate Editions of Enterprise Architect

C Conventions

Enterprise Architect supports round trip engineering of C, where these conventions are used:

Stereotype

Stereotype	Applies To
enumeration	Inner Class Corresponds To: An enumerated type.
struct	Inner Class Corresponds To: A 'struct' type.
Attribute	A keyword struct in variable definition.
typedef	Inner Class Corresponds To: A 'typedef' statement, where the parent is the original type name.
union	Inner Class Corresponds To: A union type.
Attribute	A keyword union in variable definition.

Tagged Values

Tag	Applies To
anonymous	Class also containing the Tagged Value typedef Corresponds To: The name of this Class being defined only by the typedef statement.
bitfield	Attribute Corresponds To: The size, in bits, allowed for storage of this attribute.
bodyLocation	Operation Corresponds To: The location the method body is generated to; expected values are header, classDec or classBody.
typedef	Class with stereotype other than 'typedef' Corresponds To: This Class being defined in a 'typedef' statement.
typeSynonym	Class

s	Corresponds To: The 'typedef' name and/or fields of this type.

C Code Generation for UML Model

UML	C Code
A Class	A pair of C files (.h + .c) Notes: File name is the same as Class name
Operation (public & protected)	Function declaration in .h file and definition in .c file Notes:
Operation (private)	Function definition in .c file only Notes:
Operation (static)	Function definition in .c file only Notes: Static functions will only appear in the .c file regardless of their scope.
Attribute (public & protected)	Variable definition in .h file Notes:

Attribute (private)	Variable definition in .c file Notes:
Inner Class (without stereotype)	(N/A) Notes: This inner Class would be ignored

Capture #define value to be generated in C code

For example, #define PI 3.14.

Step	Process
1	Add an attribute to the Class, with Name = PI and Initial Value = 3.14.
2	In the properties panel of the 'Attributes' page, update the 'Static' and 'Const' fields.
3	On the 'Tagged Values' tab of the 'Attributes' page, add a tag called 'define' with the value True.

Notes

- Separate conventions apply to Object Oriented programming in C

Object Oriented Programming In C

In Enterprise Architect, you apply a number of conventions for Object-Oriented programming in C.

To configure the system to support Object-Oriented programming using C, you must set the 'Object Oriented Support' option to True on the 'C Specifications' page of the 'Preferences' dialog.

Stereotypes

Stereotype	Applies To
enumeration	Class Corresponds To: An enumerated type.
struct	Class Corresponds To: A 'struct' type.
Attribute	A keyword struct in variable definition.
typedef	Class Corresponds To: A 'typedef' statement, where the parent is the original type name.

union	Class Corresponds To: A union type.
Attribute	A keyword union in variable definition.

Tagged Values

Tag	Applies To
anonymous	Class with stereotype of 'enumeration', 'struct' or 'union' Corresponds To: The name of this Class being defined only by the typedef statement.
bodyLocation	Operation Corresponds To: The location the method body is generated to; expected values are 'header', 'classDec' or 'classBody'.
define	Attribute Corresponds To: '#define' statement.
typedef	Class with stereotype of 'enumeration', 'struct' or 'union'

	Corresponds To: This Class being defined in a 'typedef' statement.

Object-Oriented C Code Generation for UML Model

The basic idea of implementing a UML Class in C code is to group the data variable (UML attributes) into a structure type; this structure is defined in a .h file so that it can be shared by other Classes and by the client that referred to it.

An operation in a UML Class is implemented in C code as a function; the name of the function must be a fully qualified name that consists of the operation name, as well as the Class name to indicate that the operation is for that Class.

A delimiter (specified in the 'Namespace Delimiter' option on the 'C Specifications' page) is used to join the Class name and function (operation) name.

The function in C code must also have a reference parameter to the Class object - you can modify the 'Reference as Operation Parameter', 'Reference Parameter Style' and 'Reference Parameter Name' options on the 'C Specifications' page to support this reference parameter.

Limitations of Object-Oriented Programming

in C

- No scope mapping for an attribute: an attribute in a UML Class is mapped to a structure variable in C code, and its scope (private, protected or public) is ignored
- Currently an inner Class is ignored: if a UML Class is the inner Class of another UML Class, it is ignored when generating C code
- Initial value is ignored: the initial value of an attribute in a UML Class is ignored in generated C code

C# Conventions

Enterprise Architect supports the round trip engineering of C#, where these conventions are used.

Stereotypes

Stereotype	Applies To
enumeration	Class Corresponds To: An enumerated type.
event	Operation Corresponds To: An event.
extension	Operation Corresponds To: A Class extension method, represented in code by a 'this' parameter in the signature.
indexer	Operation Corresponds To: A property acting as an index for this Class.
partial	Operation Corresponds To: The 'partial' keyword on

	an operation.
property	Operation Corresponds To: A property possibly containing both read and write code.
record	Class Corresponds To: A 'record' type.
struct	Class Corresponds To: A 'struct' type.

Tagged Values

Tag	Applies To
argumentName	Operation with stereotype extension Corresponds To: The name given to this parameter.
attribute_name	Operation with stereotype property or event Corresponds To: The name of the variable behind this property or event.

className	Operation with stereotype extension Corresponds To: The Class that this method is being added to.
const	Attribute Corresponds To: The const keyword.
definition	Operation with stereotype partial Corresponds To: Whether this is the declaration of the method, or the definition.
delegate	Operation Corresponds To: The 'delegate' keyword.
enumType	Operation with stereotype property Corresponds To: The datatype that the property is represented as.
expressionBody	Operation, Operation with stereotype property or indexer Corresponds To: 'True' if the 'Behavior Code' is from an expression body function member.
extensionAttribute	Operation with stereotype extension. Corresponds to: The attribute given to this parameter.

extern	Operation Corresponds To: The 'extern' keyword.
fixed	Attribute Corresponds To: The 'fixed' keyword.
generic	Operation Corresponds To: The generic parameters for this operation.
genericConstraints	Templated Class or Interface, Operation with tag 'generic' Corresponds To: The constraints on the generic parameters of this type or operation.
Implements	Operation Corresponds To: The name of the method this implements, including the interface name.
ImplementsExplicit	Operation Corresponds To: The presence of the source interface name in this method declaration.
initializer	Operation

	Corresponds To: A constructor initialization list.
new	Class, Interface, Operation Corresponds To: The 'new' keyword.
override	Operation Corresponds To: The 'override' keyword.
params	Parameter Corresponds To: A parameter list using the 'params' keyword.
partial	Class, Interface Corresponds To: The 'partial' keyword.
propertyInitializer	Operation with stereotype property Corresponds To: A property initializer.
readonly	Operation, <<struct>>Class Corresponds To: The 'readonly' keyword.
record	PositionalParameters Corresponds To: The position parameter in the record definition
ref	Operation, <<struct>>Class

	Corresponds To: The 'ref' keyword.
sealed	Operation Corresponds To: The 'sealed' keyword.
static	Class Corresponds To: The 'static' keyword.
unsafe	Class, Interface, Operation Corresponds To: The 'unsafe' keyword.
virtual	Operation Corresponds To: The 'virtual' keyword.
writeonly	Operation with stereotype property Corresponds To: This property only defining 'write' code.

Other Conventions

- Namespaces are generated for each Package below a namespace root
- The Const property of an attribute corresponds to the readonly keyword, while the tag const corresponds to the const keyword

- The value of inout for the Kind property of a parameter corresponds to the ref keyword
- The value of out for the Kind property of a parameter corresponds to the out keyword
- Partial Classes can be modeled as two separate Classes with the partial tag
- The Is Leaf property of a Class corresponds to the sealed keyword

C++ Conventions

Enterprise Architect supports round trip engineering of C++, including the Managed C++ and C++/CLI extensions, where these conventions are used.

Stereotypes

Stereotype	Applies To
enumeration	Class Corresponds To: An enumerated type.
friend	Operation Corresponds To: The 'friend' keyword.
property get	Operation Corresponds To: A 'read' property.
property set	Operation Corresponds To: A 'write' property.
struct	Class Corresponds To: A 'struct' type.
typedef	Class

	Corresponds To: A 'typedef' statement, where the parent is the original type name.
alias	Class Corresponds to an 'Alias' declaration, where the parent is the original type name.
union	Class Corresponds To: A union type.

Tagged Values

Tag	Applies To
afx_msg	Operation Corresponds To: The afx_msg keyword.
anonymous	Class also containing the Tagged Value typedef Corresponds To: The name of this Class being only defined by the typedef statement.

attribute_name	Operation with stereotype property get or property set Corresponds To: The name of the variable behind this property.
bitfield	Attribute Corresponds To: The size, in bits, allowed for storage of this attribute.
bodyLocation	Operation Corresponds To: The location the method body is generated to; expected values are header, classDec or classBody.
callback	Operation Corresponds To: A reference to the CALLBACK macro.
constexpr	Attribute and Operation Corresponds To: The constexpr keyword.
explicit	Operation Corresponds To: The 'explicit' keyword.
initializer	Operation Corresponds To: A constructor initialization list.

inline	Attribute and Operation Corresponds To: The 'inline' keyword and inline generation of the member variable definition and method body.
mutable	Attribute Corresponds To: The 'mutable' keyword.
scoped	Class with stereotype enumeration Corresponds To: Either the 'class' or 'struct' keyword.
throws	Operation Corresponds To: The exceptions that are thrown by this method.
typedef	Class with stereotype other than 'typedef' Corresponds To: This Class being defined in a 'typedef' statement.
typeSynonyms	Class Corresponds To: The 'typedef' name and/or fields of this type.
volatile	Operation Corresponds To: The 'volatile' keyword.

Other Conventions

- Namespaces are generated for each Package below a namespace root
- By Reference attributes correspond to a pointer to the type specified
- The Transient property of an attribute corresponds to the volatile keyword
- The Abstract property of an attribute corresponds to the virtual keyword
- The Const property of an operation corresponds to the const keyword, specifying a constant return type
- The Is Query property of an operation corresponds to the const keyword, specifying the method doesn't modify any fields
- The Pure property of an operation corresponds to a pure virtual method using the "= 0" syntax
- The Fixed property of a parameter corresponds to the const keyword

Managed C++ Conventions

These conventions are used for managed extensions to C++ prior to C++/CLI. In order to set the system to generate managed C++ you must modify the C++ version in the C++ Options.

Stereotypes

Stereotype	Applies To
property	Operation Corresponds To: The '__property' keyword.
property get	Operation Corresponds To: The '__property' keyword and a read property.
property set	Operation Corresponds To: The '_ _property' keyword and a 'write' property.
reference	Class Corresponds To: The '__gc' keyword.

value	Class Corresponds To: The '__value' keyword.

Tagged Values

Tag	Applies To
managedType	Class with stereotype reference, value or enumeration; Interface Corresponds To: The keyword used in declaration of this type; expected values are 'class' or 'struct'.

Other Conventions

- The typedef and anonymous tags from native C++ are not supported
- The Pure property of an operation corresponds to the keyword __abstract

C++/CLI Conventions

These conventions are used for modeling C++/CLI extensions to C++. In order to set the system to generate managed C++/CLI you must modify the C++ version in the C++ Options.

Stereotypes

Stereotype	Applies To
event	Operation Description: Defines an event to provide access to the event handler for this Class.
property	Operation, Attribute Description: This is a property possibly containing both read and write code.
reference	Class Description: Corresponds to the 'ref class' or 'ref struct' keyword.
value	Class Description: Corresponds to the 'value class' or 'value struct' keyword.

Tagged Values

Tag	Applies To
attribute_name	Operation with stereotype property or event Description: The name of the variable behind this property or event.
generic	Operation Description: Defines the generic parameters for this Operation.
genericConstraints	Templated Class or Interface, Operation with tag generic Description: Defines the constraints on the generic parameters for this Operation.
initonly	Attribute Description: Corresponds to the 'initonly' keyword.
literal	Attribute Description: Corresponds to the literal

	keyword.
managedType	Class with stereotype reference, value or enumeration; Interface Description: Corresponds to either the 'class' or 'struct' keyword.

Other Conventions

- The typedef and anonymous tags are not used
- The property get/property set stereotypes are not used
- The Pure property of an operation corresponds to the keyword abstract

Delphi Conventions

Enterprise Architect supports round trip engineering of Delphi, where these conventions are used:

Stereotypes

Stereotype	Applies To
constructor	Operation Corresponds To: A constructor.
destructor	Operation Corresponds To: A destructor.
dispinterface	Class, Interface Corresponds To: A dispatch interface.
enumeration	Class Corresponds To: An enumerated type.
metaclass	Class Corresponds To: A metaclass type.
object	Class Corresponds To: An object type.

operator	Operation Corresponds To: An operator.
property get	Operation Corresponds To: A 'read' property.
property set	Operation Corresponds To: A 'write' property.
struct	Class Corresponds To: A record type.

Tagged Values

Tag	Applies To
attribute_name	Operation with stereotype property get or property set Corresponds To: The name of the variable behind this property.
overload	Operation Corresponds To: The 'overload' keyword.

override	Operation Corresponds To: The 'override' keyword.
packed	Class Corresponds To: The 'packed' keyword.
property	Class Corresponds To: A property; see *Delphi Properties* for more information.
reintroduce	Operation Corresponds To: The 'reintroduce' keyword.

Other Conventions

- The Static property of an attribute or operation corresponds to the 'class' keyword
- The Fixed property of a parameter corresponds to the 'const' keyword
- The value of inout for the Kind property of a parameter corresponds to the 'Var' keyword
- The value of out for the Kind property of a parameter corresponds to the 'Out' keyword

Java Conventions

Enterprise Architect supports round trip engineering of Java - including AspectJ extensions - where these conventions are used.

Stereotypes

Stereotype	Applies To
annotation	Interface Corresponds To: An annotation type.
CompactConstructor	Operation Corresponds to: A compact canonical constructor for the record.
record	Class Corresponds To: A record type.
default	Operation Corresponds To: The 'default' keyword.
enum	Attributes within a Class stereotyped enumeration Corresponds To: An enumerated option,

	distinguished from other attributes that have no stereotype.
enumeration	Class Corresponds To: An enumerated type.
operator	Operation Corresponds To: An operator.
property get	Operation Corresponds To: A 'read' property.
property set	Operation Corresponds To: A 'write' property.
static	Class or Interface Corresponds To: The 'static' keyword.

Tagged Values

Tag	Applies To
annotations	Anything Corresponds To: The annotations on the current code feature.

arguments	Attribute with stereotype enum Corresponds To: The arguments that apply to this enumerated value.
attribute_name	Operation with stereotype property get or property set Corresponds To: The name of the variable behind this property.
dynamic	Class or Interface Corresponds To: The 'dynamic' keyword.
generic	Operation Corresponds To: The generic parameters to this operation.
parameterList	Parameter Corresponds To: A parameter list with the ... syntax.
RecordHeader	<<record>>Class Corresponds To: The record header of the record definition.
throws	Operation Corresponds To: The exceptions that are thrown by this method.

transient	Attribute Corresponds To: The 'transient' keyword.

Other Conventions

- Package statements are generated when the current Package is not a namespace root
- The Const property of an attribute or operation corresponds to the final keyword
- The Transient property of an attribute corresponds to the volatile keyword
- The Fixed property of a parameter corresponds to the final keyword

AspectJ Conventions

These are the conventions used for supporting AspectJ extensions to Java.

Stereotypes

Stereotype	Applies To
advice	Operation Corresponds To: A piece of advice in an AspectJ aspect.
aspect	Class Corresponds To: An AspectJ aspect.
pointcut	Operation Corresponds To: A 'pointcut' in an AspectJ aspect.

Tagged Values

Tag	Applies To

className	Attribute or operation within a Class stereotyped aspect Corresponds To: The Classes this AspectJ intertype member belongs to.

Other Conventions

- The specifications of a pointcut are included in the 'Behavior' field of the method

PHP Conventions

Enterprise Architect supports the round trip engineering of PHP 4 and 5, where these conventions are used.

Stereotypes

Stereotype	Applies To
trait	Class Corresponds To: A 'trait'.
property get	Operation Corresponds To: A 'read' property.
property set	Operation Corresponds To: A 'write' property.

Tagged Values

Tag	Applies To
attribute_nam	Operation with stereotype property get or

e	property set Corresponds To: The name of the variable behind this property.
final	Operations in PHP 5 Corresponds To: The 'final' keyword.

Common Conventions

- An unspecified type is modeled as var
- Methods returning a reference are generated by setting the Return Type to var*
- Reference parameters are generated from parameters with the parameter Kind set to inout or out

PHP 5 Conventions

- The final Class modifier corresponds to the Is Leaf property
- The abstract Class modifier corresponds to the Abstract property
- Parameter type hinting is supported by setting the Type of a parameter

- The value of inout or out for the Kind property of a parameter corresponds to a reference parameter

Python Conventions

Enterprise Architect supports the round trip engineering of Python, where these conventions are used.

Tagged Values

Tag	Applies To
async	Operation Corresponds To: The "async" keyword in function definition.
Decorators	Class, Operation Corresponds To: The decorators applied to this element in the source.

Other Conventions

- Model members with Private Scope correspond to code members with two leading underscores
- Attributes are only generated when the Initial value is not empty
- All types are reverse engineered as var

SystemC Conventions

Enterprise Architect supports round-trip engineering of SystemC, where these conventions are used.

Stereotypes

Stereotype	Applies To
delegate	Method Corresponds To: A delegate.
enumeration	Inner Class Corresponds To: An enum type.
friend	Method Corresponds To: A friend method.
property	Method Corresponds To: A property definition.
sc_ctor	Method Corresponds To: A SystemC constructor.
sc_module	Class Corresponds To: A SystemC module.

sc_port	Attribute Corresponds To: A port.
sc_signal	Attribute Corresponds To: A signal.
struct	Inner Class Corresponds To: A struct or union.

Tagged Values

Tag	Applies To
kind	Attribute (Port) Corresponds To: Port kind (clocked, fifo, master, slave, resolved, vector).
mode	Attribute (Port) Corresponds To: Port mode (in, out, inout).
overrides	Method Corresponds To: The Inheritance list of a method declaration.

throw	Method Corresponds To: The exception specification of a method.

Other Conventions

- SystemC also inherits most of the stereotypes and Tagged Values of C++

SystemC Toolbox Pages

To model a SystemC design, drag these icons onto a diagram from the 'SystemC Constructs' page of the Diagram Toolbox.

Page	Icon
SystemC	Module Action: Defines a SystemC Module. An sc_module -stereotyped Class element.
SystemC Features	Port Action: Defines a SystemC Port.

	An sc_port- stereotyped attribute.

Access

Ribbon	Design > Diagram > Toolbox :  > Specify 'SystemC Constructs' in the 'Find Toolbox Item' dialogs
Keyboard Shortcuts	Ctrl+Shift+3 :  > Specify 'SystemC Constructs' in the 'Find Toolbox Item' dialog
Other	You can display or hide the Diagram Toolbox by clicking on the or icons at the left-hand end of the Caption Bar at the top of the Diagram View.

VB.NET Conventions

Enterprise Architect supports round-trip engineering of Visual Basic.NET, where these conventions are used. Earlier versions of Visual Basic are supported as a different language.

Stereotypes

Stereotype	Applies To
event	Operation Corresponds To: An event declaration.
import	Operation Corresponds To: An operation to be imported from another library.
module	Class Corresponds To: A module.
operator	Operation Corresponds To: An operator overload definition.
partial	Operation

	Corresponds To: The 'partial' keyword on an operation.
property	Operation Corresponds To: A property possibly containing both read and write code.

Tagged Values

Tag	Applies To
Alias	Operation with stereotype import Corresponds To: The alias for this imported operation.
attribute_name	Operation with stereotype property Corresponds To: The name of the variable behind this property.
Charset	Operation with stereotype import Corresponds To: The character set clause for this import - one of the values 'Ansi', 'Unicode' or 'Auto'.
delegate	Operation

	Corresponds To: The 'delegate' keyword.
enumTag	Operation with stereotype property Corresponds To: The datatype that this property is represented as.
Handles	Operation Corresponds To: The 'handles' clause on this operation.
Implements	Operation Corresponds To: The 'implements' clause on this operation.
Lib	Operation with stereotype import Corresponds To: The library this import comes from.
MustOverrid e	Operation Corresponds To: The 'MustOverride' keyword.
Narrowing	Operation with stereotype operator Corresponds To: The 'Narrowing' keyword.
NotOverridea ble	Operation Corresponds To: The 'NotOverrideable'

	keyword.
Overloads	Operation Corresponds To: The 'overloads' keyword.
Overrides	Operation Corresponds To: The 'overrides' keyword.
parameterArray	Parameter Corresponds To: A parameter list using the 'ParamArray' keyword.
partial	Class, Interface Corresponds To: The 'partial' keyword.
readonly	Operation with stereotype property Corresponds To: This property only defining 'read' code.
shadows	Class, Interface, Operation Corresponds To: The 'Shadows' keyword.
Shared	Attribute Corresponds To: The 'Shared' keyword.
Widening	Operation with stereotype operator

	Corresponds To: The 'Widening' keyword.
writeonly	Operation with stereotype property Corresponds To: This property only defining 'write' code.

Other Conventions

- Namespaces are generated for each Package below a namespace root
- The Is Leaf property of a Class corresponds to the NotInheritable keyword
- The Abstract property of a Class corresponds to the MustInherit keyword
- The Static property of an attribute or operation corresponds to the Shared keyword
- The Abstract property of an operation corresponds to the MustOverride keyword
- The value of in for the Kind property of a parameter corresponds to the ByVal keyword
- The value of inout or out for the Kind property of a parameter corresponds to the ByRef keyword

Verilog Conventions

Enterprise Architect supports round-trip engineering of Verilog, where these conventions are used.

Stereotypes

Stereotype	Applies To
asynchronous	Method Corresponds To: A concurrent process.
enumeration	Inner Class Corresponds To: An enum type.
initializer	Method Corresponds To: An initializer process.
module	Class Corresponds To: A module.
part	Attribute Corresponds To: A component instantiation.
port	Attribute

	Corresponds To: A port.
synchronous	Method Corresponds To: A sequential process.

Tagged Values

Tag	Applies To
kind	Attribute (signal) Corresponds To: The signal kind (such as register, bus).
mode	Attribute (Port) Corresponds To: The Port mode (in, out, inout).
Portmap	Attribute (part) Corresponds To: The generic/Port map of the component instantiated.
sensitivity	Method Corresponds To: The sensitivity list of a sequential process.

type	Attribute Corresponds To: The range or type value of an attribute.

Verilog Toolbox Pages

Access: 'Design > Diagram > Toolbox : 'Hamburger' icon > HDL | Verilog Constructs'

Drag these icons onto a diagram to model a Verilog design.

Item	Action
Module	Defines a Verilog Module. A module-stereotyped Class element.
Enumeration	Defines an Enumerated Type. An enumeration element.
Port	Defines a Verilog Port. A port-stereotyped attribute.
Part	Defines a Verilog component instantiation. A part-stereotyped attribute.
Attribute	Defines an attribute.

Procedure	Defines a Verilog process: • Concurrent - An asynchronous-stereotyped method • Sequential - A synchronous-stereotyped method • Initializer - An initializer-stereotyped method

VHDL Conventions

Enterprise Architect supports round-trip engineering of VHDL, where these conventions are used.

Stereotypes

Stereotype	Applies To
architecture	Class Corresponds To: An architecture.
asynchronous	Method Corresponds To: An asynchronous process.
configuration	Method Corresponds To: A configuration.
enumeration	Inner Class Corresponds To: An enumerated type.
entity	Interface Corresponds To: An entity.
part	Attribute

	Corresponds To: A component instantiation.
port	Attribute Corresponds To: A port.
signal	Attribute Corresponds To: A signal declaration.
struct	Inner Class Corresponds To: A record definition.
synchronous	Method Corresponds To: A synchronous process.
typedef	Inner Class Corresponds To: A type or subtype definition.

Tagged Values

Tag	Applies To
isGeneric	Attribute (port) Corresponds To: The 'port' declaration in

	a generic interface.
isSubType	Inner Class (typedef) Corresponds To: A subtype definition.
kind	Attribute (signal) Corresponds To: The signal kind (such as 'register', 'bus').
mode	Attribute (Port) Corresponds To: The Port mode ('in', 'out', 'inout', 'buffer', 'linkage').
portmap	Attribute (part) Corresponds To: The generic/Port map of the component instantiated.
sensitivity	Method (synchronous) Corresponds To: The 'sensitivity' list of a synchronous process.
type	Inner Class (typedef) Corresponds To: The 'type' indication of a 'type' declaration.
typeNameSpace	Attribute (part) Corresponds To: The 'type' namespace of the instantiated component.

VHDL Toolbox Pages

Access

To model a VHDL design, drag icons from the VHDL toolbox pages and drop them on your diagram.

Ribbon	Design > Diagram > Toolbox :  > Specify 'VHDL Constructs' in the 'Find Toolbox Item' dialog
Keyboard Shortcuts	Ctrl+Shift+3 :  > Specify 'VHDL Constructs' in the 'Find Toolbox Item' dialog
Other	You can display or hide the Diagram Toolbox by clicking on the  or  icons at the left-hand end of the Caption Bar at the top of the Diagram View.

VHDL Toolbox Page

Item	Action
Architecture	Defines an architecture to be associated with a VHDL entity. An architecture-stereotyped Class element.
Entity	Defines a VHDL entity to contain the Port definitions. An entity-stereotyped interface element.
Enumeration	Defines an Enumerated Type. An Enumeration element.
Struct	Defines a VHDL record. A struct-stereotyped Class element.
Typedef	Defines a VHDL type or subtype. A typedef-stereotyped Class element.

VHDL Features Toolbox Page

Item	Action
Part	Defines a VHDL component instantiation. A part-stereotyped attribute.
Port	Defines a VHDL Port. A port-stereotyped attribute.
Signal	Defines a VHDL signal. A signal-stereotyped attribute.
Procedure	Defines a VHDL process: • Concurrent - An asynchronous-stereotyped method • Sequential - A synchronous-stereotyped method • Configuration - An configuration-stereotyped method

Visual Basic Conventions

Enterprise Architect supports the round trip engineering of Visual Basic 5 and 6, where these conventions are used.

Visual Basic .NET is supported as a different language.

Stereotypes

Stereotype	Applies To
global	Attribute Corresponds To: The 'Global' keyword.
import	Operation Corresponds To: An operation to be imported from another library.
property get	Operation Corresponds To: A property 'get'.
property set	Operation Corresponds To: A property 'set'.
property let	Operation Corresponds To: A property 'let'.

with events	Attribute Corresponds To: The 'WithEvents' keyword.

Tagged Values

Tag	Applies To
Alias	Operation with stereotype import Corresponds To: The alias for this imported operation.
attribute_name	Operation with stereotype property get, property set or property let Corresponds To: The name of the variable behind this property.
Lib	Operation with stereotype import Corresponds To: The library this import comes from.
New	Attribute Corresponds To: The 'new' keyword.

Other Conventions

- The value of in for the Kind property of a parameter corresponds to the ByVal keyword
- The value of inout or out for the Kind property of a parameter corresponds to the ByRef keyword

Language Options

You can set up various options for how Enterprise Architect handles a particular language when generating and reverse-engineering code. These options are either specific to:

- Your user ID, for all models or
- The model in which they are defined, for all users

Access

Ribbon	Start > Appearance > Preferences > Preferences > Source Code Engineering > <language name> Settings > Model > Options > Source Code Engineering > <language name>
Keyboard Shortcuts	Ctrl+F9 ('Preferences' dialog)

Languages Supported

Language

Action Script
Ada 2012 (in the Unified and Ultimate Editions of Enterprise Architect)
ArcGIS
ANSI C
C#
C++
Delphi
Java
PHP
Python
SystemC
Verilog (Unified and Ultimate Editions)
VHDL (Unified and Ultimate Editions)

Visual Basic
Visual Basic .NET

ActionScript Options - User

If you intend to generate ActionScript code from your model, you can configure the code generation options using the 'ActionScript Specifications' page of the 'Preferences' dialog to:

- Specify the default source directory
- Specify the editor for ActionScript code

Access

Ribbon	Start > Appearance > Preferences > Preferences > Source Code Engineering > ActionScript
Keyboard Shortcuts	Ctrl+F9

Options

Option	Action
Disable	Leave this checkbox unselected to

Language	support ActionScript code generation. Select this checkbox to disable ActionScript code support.
Options for the current user	In the 'Default Source Directory' and 'Editor' fields, click on the ... button and browse for the source directory and external file editor that you will use.

Notes

- These options apply to all models that you access

ActionScript Options - Model

If you intend to generate ActionScript code from your model, you can configure the model-specific code generation options using the 'ActionScript Specifications' page of the 'Manage Project Options' dialog to:

- Specify default ActionScript version to generate (AS2.0 or AS3.0)
- Specify default file extensions
- Specify the Collection Class definitions for Association connectors

Access

Ribbon	Settings > Model > Options > Source Code Engineering > ActionScript

Options

Option	Action
Options for the current	Type in the default ActionScript version and default file extension to apply when

model	generating ActionScript source code.
Collection Classes	Click on this button to open the 'Collection Classes for Association Roles' dialog, through which you specify the Collection Class definitions for Association connectors.

Notes

- These options affect all users of the current model; however, they do not apply to other models

Ada 2012 Options - User

If you intend to generate Ada 2012 code from your model, you can configure the code generation options using the 'Ada' page of the 'Preferences' dialog to:

- Inform the reverse engineering process whether the name of the Tagged Record is the same as the Package name
- Advise the engine of the alternate Tagged Record name to locate
- Specify whether the engine should create a reference type for the Tagged Record (if one is not defined)
- Supply the name of the reference type to be created (default is Ref)
- Specify the reference parameter of a Reference / Access type
- Tell the engine to ignore the name of the reference parameter
- Indicate the name of the reference parameter to locate

Access

Ribbon	Start > Appearance > Preferences > Preferences > Source Code Engineering > Ada

Keyboard Shortcuts	Ctrl+F9

Options

Option	Action
Disable Language	Leave this checkbox unselected to support Ada 2012 code generation. Select this checkbox to disable Ada 2012 code support.
Options for the current user	Specifies the options used for the current user; these options apply to all models that are accessed by the user.

Notes

- Ada 2012 support is available in the Unified and Ultimate Editions of Enterprise Architect

Ada 2012 Options - Model

If you intend to generate Ada 2012 code from your model, you can configure the model-specific code generation options using the 'Ada' page of the 'Manage Project Options' dialog to:

- Specify the default file extension and
- Specify the Collection Class definitions for Association connectors

Access

Ribbon	Settings > Model > Options > Source Code Engineering > Ada

Options

Option	Action
Options for the current model	Type in the default file extension to apply when generating Ada source code.

Collection Classes	Click on this button to open the 'Collection Classes for Association Roles' dialog, through which you specify the Collection Class definitions for Association connectors.

Notes

- These options affect all users of the current model; however, they do not apply to other models
- Ada 2012 support is available in the Unified and Ultimate Editions of Enterprise Architect

ArcGIS Options - User

If you intend to generate ArcGIS code from your model, you can configure the code generation options using the 'ArcGIS' page of the 'Preferences' dialog to:

- Specify default source directory
- Specify the editor for ArcGIS code

ArcGIS must be enabled in the 'MDG Technologies' dialog ('Specialize > Technologies > Manage') in order for the 'ArcGIS' page to be available.

Access

Ribbon	Start >Appearance > Preferences > Preferences > Source Code Engineering > ArcGIS
Keyboard Shortcuts	Ctrl+F9

Options

Option	Action

Disable Language	Leave this checkbox unselected to support ArcGIS code generation. Select this checkbox to disable ArcGIS code support.
Options for the current user	Specifies the options used for the current user; these options apply to all models that are accessed by the user.

ArcGIS Options - Model

If you intend to generate ArcGIS code from your model, you can configure the model-specific code generation options using the 'ArcGIS' page of the 'Manage Project Options' dialog to:

- Specify default file extensions
- Specify the Collection Class definitions for Association connectors

Access

Ribbon	Settings > Model > Options > Source Code Engineering > ArcGIS

Options

Option	Action
Options for the current model	Type in the default file extension to apply when generating ArcGIS source code.

Collection Classes	Click on this button to open the 'Collection Classes for Association Roles' dialog, through which you specify the Collection Class definitions for Association connectors.

Notes

- These options affect all users of the current model; however, they do not apply to other models

C Options - User

If you intend to generate C code from your model, you can configure the code generation options using the 'C Specifications' page of the 'Preferences' dialog.

Access

Ribbon	Start > Application > Preferences > Preferences > Source Code Engineering > C
Keyboard Shortcuts	Ctrl+F9

Options

Option	Action
Disable Language	Leave this checkbox unselected to support C code generation. Select this option to disable C code support.

Options for the current user	In the value fields, specify the options that apply under your own user ID in all models that you access: • The default attribute type to create (fixed as int) • Whether a #define constant is imported as an attribute in imported C code (if 'Object Oriented programming' is set to True on the 'C Specifications' page of the 'Manage Project Options' dialog) • Whether to generate comments for C methods to the declaration, and to reverse engineer comments from the declaration • Whether to generate comments for C methods to the implementation, and to reverse engineer comments from the implementation • Whether to update comments in regenerating code from the model • Whether to update the implementation file in re-generating code from the model • The default source code directory location (click on the ... button) • The default file extensions to read when importing a directory of C code

	• The Code Editor to use (click on the ... button) • The search path for the implementation file relative to the header file path

C Options - Model

If you intend to generate C code from your model, you can configure the model-specific code generation options using the 'C Specifications' page of the 'Manage Project Options' dialog to:

- Specify default file extensions (header and source)
- Define support for Object Oriented programming
- Set the StateMachine engineering options
- Specify the Collection Class definitions for Association connectors

Access

Ribbon	Settings > Model > Options > Source Code Engineering > C

Options

Option	Action
Options for the current	In the value fields, specify these options:

model	• The default header and source file extensions for the code files • Support for Object Oriented programming; if this is True, then set: - The Namespace delimiter character - Whether the first parameter of an operation is a Class reference - The parameter reference style in generated C code - The reference parameter name in generated code - The default Constructor name in generated code - The default Destructor name in generated code
StateMachine Engineering	In the value fields, use the drop-down arrows to set the options to True or False; these options apply to generating code from StateMachine models in the current model only: • 'Use the new StateMachine Template' - set to True to use the code generation templates from Enterprise Architect Release 11 and later, set to False to apply the EASL Legacy templates • Generate Trace Code - set to True to generate Trace code, False to omit it

Collection Classes	Click on this button to open the 'Collection Classes for Association Roles' dialog, through which you specify the Collection Class definitions for Association connectors.

Notes

- These options affect all users of the current model; however, they do not apply to other models

C# Options - User

If you intend to generate C# code from your model, you can configure the code generation options using the 'C# Specifications' page of the 'Preferences' dialog

Access

Ribbon	Start > Application > Preferences > Preferences > Source Code Engineering > C#
Keyboard Shortcuts	Ctrl+F9

Options

Option	Action
Disable Language	Leave this checkbox unselected to support C# code generation. Select this checkbox to disable C# code support.

Options for the current user	In the value fields, specify the options that apply under your own user ID in all models that you access: • The default attribute type to create • Whether Namespaces should be generated when generating C# Classes • Whether to remove new lines (hard carriage returns) from the summary tag when importing XML.NET style comments • Whether to generate a Finalizer method when generating code for a C# Class • Whether to generate a Dispose method when generating code for a C# Class • The default source code directory location (click on the ... button) • The Code Editor to use (click on the ... button)

C# Options - Model

If you intend to generate C# code from your model, you can configure the model-specific code generation options using the 'C# Specifications' page of the 'Manage Project Options' dialog to:

- Specify the default file extension
- Indicate additional Collection Classes - to define custom Collection Classes, which can be simple substitutions (such as CArray<#TYPE#>) or a mix of other strings and substitutions (such as Cmap<CString,LPCTSTR,#TYPE#*,#TYPE#*>); these Collection Classes are defined by default:
  - List<#TYPE#>;Stack<#TYPE#>;Queue<#TYPE#>;
- Set the StateMachine Engineering options
- Specify the Collection Class definitions for Association connectors

Access

Ribbon	Settings > Model > Options > Source Code Engineering > C#

Options

Option	Action
Options for the current model	Type in the default file extension to apply when generating C# source code, and a list of any additional Collection Classes you want to define.
StateMachine Engineering	In the value fields, use the drop-down arrows to set the options to True or False; these options apply to generating code from StateMachine models in the current model only: • 'Use the new StateMachine Template' - set to True to use the code generation templates from Enterprise Architect Release 11 and later, set to False to apply the EASL Legacy templates • 'Generate Trace Code' - set to True to generate Trace code, False to omit it
Collection Classes	Click on this button to open the 'Collection Classes for Association Roles' dialog, through which you specify the Collection Class definitions for Association connectors.

Notes

- These options affect all users of the current model; however, they do not apply to other models

C++ Options - User

If you intend to generate C++ code from your model, you can configure the code generation options using the 'C++ Specifications' page of the 'Preferences' dialog.

Access

Ribbon	Start > Appearance > Preferences > Preferences > Source Code Engineering > C++
Keyboard Shortcuts	Ctrl+F9

Options

Option	Action
Disable Language	Leave this checkbox unselected to support C++ code generation. Select this option to disable C++ code support.

Options for the current user	In the value fields, specify the options that apply under your own user ID in all models that you access: • The default attribute type to create • Whether Namespaces should be generated when generating C++ Classes • What style to apply when generating and processing comments for C++ • Whether to generate comments for C++ methods to the declaration, or reverse engineer comments from the declaration • Whether to generate comments for C++ methods to the implementation, or reverse engineer comments from the implementation • Whether to update comments in re-generating code from the model • Whether to update the implementation file in re-generating code from the model • The default source code directory location (click on the ... button) • The default file extensions to read when importing a directory of C++ code

	• The Code Editor to use (click on the ... button) • The search path for the implementation file relative to the header file path

C++ Options - Model

If you intend to generate C++ code from your model, you can configure the model-specific code generation options using the 'C++ Specifications' page of the 'Manage Project Options' dialog to:

- Indicate the version of C++ to generate; this controls the set of templates used and how properties are created
- Specify the default reference type used when a type is specified by reference
- Specify the default file extensions
- Specify default Get/Set prefixes
- Specify the Collection Class definitions for Association connectors
- Define additional Collection Classes - to define custom Collection Classes, which can be simple substitutions (such as CArray<#TYPE#>) or a mix of other strings and substitutions (such as Cmap<CString,LPCTSTR,#TYPE#*,#TYPE#*>); these Collection Classes are defined by default:
  - CArray<#TYPE#>;CMap<CString,LPCTSTR,#TYPE#*, #TYPE#*>;
- Set the StateMachine Engineering options

Access

Ribbon	Settings > Model > Options > Source Code Engineering > C++

Options

Option	Action
Options for the current model	In the value fields, specify the options that affect all users of the current model: • The version of C++ you are using (which determines which templates to use when generating code) • The default reference type to use when creating properties for C++ attributes by reference • The default header and source file extensions for the code files • The default 'Get' prefix • The default 'Set' prefix • The additional Collection Classes
StateMachine Engineering Options	In the value fields, use the drop-down arrows to set the options to True or False; these options apply to generating code

	from StateMachine models in the current model only: • 'Use the new StateMachine Template' - set to True to use the code generation templates from Enterprise Architect Release 11 and later, set to False to apply the EASL Legacy templates • 'Generate Trace Code' - set to True to generate Trace code, False to omit it
Collection Classes	Click on this button to open the 'Collection Classes for Association Roles' dialog, through which you specify the Collection Class definitions for Association connectors.

Notes

- These options affect all users of the current model; however, they do not apply to other models

Delphi Options - User

If you intend to generate Delphi code from your model, you can configure the code generation options using the 'Delphi Specifications' page of the 'Preferences' dialog to:

- Set the default attribute type
- Indicate a default source directory
- Set the default code editor to use to edit Delphi source code

Access

Ribbon	Start > Appearance > Preferences > Preferences > Source Code Engineering > Delphi
Keyboard Shortcuts	Ctrl+F9

Options

Option	Action

Disable Language	Leave this checkbox unselected to support Delphi code generation. Select this option to disable Delphi code support.
Options for the current user	Specifies the options used for the current user; these options apply to all models that are accessed by the user.

Delphi Options - Model

If you intend to generate Delphi code from your model, you can configure the model-specific code generation options using the 'Delphi Specifications' page of the 'Manage Project Options' dialog to:

- Specify default file extensions (header and source)
- Specify the Collection Class definitions for Association connectors

Access

Ribbon	Settings > Model > Options > Source Code Engineering > Delphi

Options

Option	Action
Options for the current model	Type in the default file extension to apply when generating Delphi source code.

Collection Classes	Click on this button to open the 'Collection Classes for Association Roles' dialog, through which you specify the Collection Class definitions for Association connectors.

Notes

- These options affect all users of the current model; however, they do not apply to other models

Delphi Properties

Enterprise Architect has comprehensive support for Delphi properties. These are implemented as Tagged Values, with a specialized property editor to help create and modify Class properties. By using the 'Feature Visibility' element context menu option, you can display the 'tags' compartment that contains the properties. Imported Delphi Classes with properties have this feature automatically made visible for your convenience.

Manually activate the property editor

- In the selected Class set the code generation language to 'Delphi'
- Right-click on the Class and select 'Delphi Properties' to open the editor

Using the Delphi Properties editor, you can build properties quickly and simply; from here you can:

- Change the name and scope (only Public and Published are currently supported)
- Change the property type (the drop-down list includes all defined Classes in the project)
- Set the Read and Write information (the drop-down lists have all the attributes and operations from the current Class; you can also enter free text)
- Set 'Stored' to True or False

- Set the Implements information
- Set the default value, if one exists

Notes

- When you use the 'Create Property' dialog from the 'Attribute' screen, the system generates a pair of Get and Set functions together with the required property definition as Tagged Values; you can manually edit these Tagged Values if required
- Public properties are displayed with a '+' symbol prefix and published with a '^'
- When creating a property in the 'Create Property Implementation' dialog (accessed through the 'Attributes' dialog), you can set the scope to 'Published' if the property type is Delphi
- Only 'Public' and 'Published' are supported
- If you change the name of a property and forward engineer, a new property is added, but you must manually delete the old one from the source file

Java Options - User

If you intend to generate Java code from your model, you can configure the code generation options using the 'Java Specifications' page of the 'Preferences' dialog.

Access

Ribbon	Start > Appearance > Preferences > Preferences > Source Code Engineering > Java
Keyboard Shortcuts	Ctrl+F9

Options

Option	Action
Disable Language	Leave this checkbox unselected to support Java code generation. Select this checkbox to disable Java code support.

Options for the current user	In the value fields, specify the options that apply under your own user ID in all models that you access; the: • Default attribute type to create (select from the drop-down list) • Default source code directory location (click on the ... button) • Code Editor to use (click on the ... button)

Java Options - Model

If you intend to generate Java code from your model, you can configure the model-specific code generation options using the 'Java Specifications' page of the 'Manage Project Options' dialog to:

- Specify the default file extension
- Specify a default 'Get' prefix
- Specify a default 'Set' prefix
- Set the StateMachine Engineering options
- Specify the Collection Class definitions for Association connectors
- Define additional Collection Classes - to define custom Collection Classes, which can be simple substitutions (such as CArray<#TYPE#>) or a mix of other strings and substitutions (such as Cmap<CString,LPCTSTR,#TYPE#*,#TYPE#*>); these Collection Classes are defined by default:
    - HashSet<#TYPE#>;Map<String,#TYPE#>;

Access

Ribbon	Settings > Model > Options > Source Code Engineering > Java

Options

Option	Action
Options for the current model	In the value fields, specify the options that affect all users of the current model; the: • Default file extension for the code files • The default Get and Set prefixes • The default and additional Collection Classes
StateMachine Engineering	In the value fields, use the drop-down arrows to set the options to True or False; these options apply to generating code from StateMachine models in the current model only: • 'Use the new StateMachine Template' - set to True to use the code generation templates from Enterprise Architect Release 11 and later, set to False to apply the EASL Legacy templates • 'Generate Trace Code' - set to True to generate Trace code, False to omit it
Collection	Click on this button to open the

Classes	'Collection Classes for Association Roles' dialog, through which you specify the Collection Class definitions for Association connectors.

Notes

- These options affect all users of the current model; however, they do not apply to other models

MySQL Options - User

If you intend to generate MySQL code from your model, you can configure the code generation options using the 'MySQL' page of the 'Preferences' dialog to:

- Specify a default attribute type
- Specify a default source directory
- Specify file name extensions for files to import
- Specify an editor for changing code
- Specify a default owner

Access

Ribbon	Start > Application > Preferences > Preferences > Source Code Engineering > MySQL
Keyboard Shortcuts	Ctrl+F9

Options

Option	Action
Disable Language	Leave this checkbox unselected to support MySQL code generation. Select this option to disable MySQL code support.
Options for the current user	Specifies the options used for the current user; these options apply to all models that are accessed by the user.

MySQL Options - Model

If you intend to generate MySQL code from your model, you can configure the model-specific code generation options using the 'MySQL' page of the 'Manage Project Options' dialog to:

- Specify the default file extension
- Specify the Collection Class definitions for Association connectors

Access

Ribbon	Settings > Model > Options > Source Code Engineering > MySQL

Options

Option	Action
Options for the current model	Type in the default file extension to apply when generating MySQL source code.

Collection Classes	Click on this button to open the 'Collection Classes for Association Roles' dialog, through which you specify the Collection Class definitions for Association connectors.

Notes

- These options affect all users of the current model; however, they do not apply to other models

PHP Options - User

If you intend to generate PHP code from your model, you can configure the code generation options using the 'PHP Specifications' page of the 'Preferences' dialog to:

- Define a semi-colon separated list of extensions to look at when doing a directory code import for PHP
- Set a default directory for opening and saving PHP source code
- Specify the default editor to use when editing PHP code

Access

Ribbon	Start > Appearance > Preferences > Preferences > Source Code Engineering > PHP
Keyboard Shortcuts	Ctrl+F9 | Source Code Engineering | PHP

Options

Option	Action

Disable Language	Leave this checkbox unselected to support PHP code generation. Select this option to disable PHP code support.
Options for the current user	Specifies the options used for the current user; these options apply to all models that are accessed by the user.

PHP Options - Model

If you intend to generate PHP code from your model, you can configure the model-specific code generation options using the 'PHP Specifications' page of the 'Manage Project Options' dialog to:

- Specify the default PHP version to generate
- Define the default file extension
- Specify a default 'Get' prefix
- Specify a default 'Set' prefix

Access

Ribbon	Settings > Model > Options > Source Code Engineering > PHP

Options

Option	Action
Options for the current	Type in the default PHP version, the default file extension to apply when generating PHP source code, and the

model	default 'Get' and 'Set' prefixes.

Notes

- These options affect all users of the current model; however, they do not apply to other models

Python Options - User

If you intend to generate Python code from your model, you can configure the code generation options using the 'Python Specifications' page of the 'Preferences' dialog to:

- Specify the default source directory to be used
- Specify the default editor used to write and edit Python code

Access

Ribbon	Start > Appearance > Preferences > Preferences > Source Code Engineering > Python
Keyboard Shortcuts	Ctrl+F9

Options

Option	Action
Disable	Leave this checkbox unselected to

Language	support Python code generation. Select this option to disable Python code support.
Options for the current user	Specifies the options used for the current user; these options apply to all models that are accessed by the user.

Python Options - Model

If you intend to generate Python code from your model, you can configure the model-specific code generation options using the 'Python Specifications' page of the 'Manage Project Options' dialog to:

- Specify the default file extension

Access

Ribbon	Settings > Model > Options > Source Code Engineering > Python

Options

Option	Action
Options for the current model	Type in the default file extension to apply when generating Python source code.

Notes

- These options affect all users of the current model; however, they do not apply to other models

SystemC Options - User

If you intend to generate SystemC code from your model, you can configure the code generation options using the 'SystemC' page of the 'Preferences' dialog to:

- Specify a default source directory
- Specify an editor for changing code

Access

Ribbon	Start > Appearance > Preferences > Preferences > Source Code Engineering > SystemC
Keyboard Shortcuts	Ctrl+F9

Options

Option	Action
Disable Language	Leave this checkbox unselected to support SystemC code generation.

	Select this option to disable SystemC code support.
Options for the current user	Specifies the options used for the current user; these options apply to all models that are accessed by the user.

SystemC Options - Model

If you intend to generate SystemC code from your model, you can configure the model-specific code generation options using the 'SystemC' page of the 'Manage Project Options' dialog to:

- Specify the default file extension
- Specify the Collection Class definitions for Association connectors

Access

Ribbon	Settings > Model > Options > Source Code Engineering > SystemC

Options

Option	Action
Options for the current model	Type in the default file extension to apply when generating SystemC source code.

Collection Classes	Click on this button to open the 'Collection Classes for Association Roles' dialog, through which you specify the Collection Class definitions for Association connectors.

Notes

- These options affect all users of the current model; however, they do not apply to other models

Teradata Options - User

If you intend to generate Teradata code from your model, you can configure the code generation options using the 'Teradata' page of the 'Preferences' dialog to:

- Specify a default attribute type
- Specify a default source directory
- Specify an editor for changing code

Access

Ribbon	Start > Application > Preferences > Preferences > Source Code Engineering > Teradata
Keyboard Shortcuts	Ctrl+F9

Options

Option	Action
Disable	Leave this checkbox unselected to

Language	support Teradata code generation. Select this option to disable Teradata code support.
Options for the current user	Specifies the options used for the current user; these options apply to all models that are accessed by the user.

Teradata Options - Model

If you intend to generate Teradata code from your model, you can configure the model-specific code generation options using the 'Teradata' page of the 'Manage Project Options' dialog to:

- Specify the default file extension
- Specify the Collection Class definitions for Association connectors

Access

Ribbon	Settings > Model > Options > Source Code Engineering > Teradata

Options

Option	Action
Options for the current model	Type in the default file extension to apply when generating Teradata source code.

Collection Classes	Click on this button to open the 'Collection Classes for Association Roles' dialog, through which you specify the Collection Class definitions for Association connectors.

Notes

- These options affect all users of the current model; however, they do not apply to other models

VB.NET Options - User

If you intend to generate VB.NET code from your model, you can configure the code generation options using the 'VB.NET Specifications' page of the 'Preferences' dialog to:

- Specify the default attribute type
- Indicate whether to generate namespaces
- Specify a default source directory
- Specify an editor for changing code

Access

Ribbon	Start > Appearance > Preferences > Preferences > Source Code Engineering > VB.Net
Keyboard Shortcuts	Ctrl+F9

Options

Option	Action

Disable Language	Leave this checkbox unselected to support VB.NET code generation. Select this option to disable VB.NET code support.
Options for the current user	Specifies the options used for the current user; these options apply to all models that are accessed by the user.

VB.NET Options - Model

If you intend to generate VB.NET code from your model, you can configure the model-specific code generation options using the 'VB.Net Specifications' page of the 'Manage Project Options' dialog to:

- Specify the default file extension
- Specify the Collection Class definitions for Association connectors

Access

Ribbon	Settings > Model > Options > Source Code Engineering > VB.Net

Options

Option	Action
Options for the current model	Type in the default file extension to apply when generating VB.Net source code.

Collection Classes	Click on this button to open the 'Collection Classes for Association Roles' dialog, through which you specify the Collection Class definitions for Association connectors.

Notes

- These options affect all users of the current model; however, they do not apply to other models

Verilog Options - User

If you intend to generate Verilog code from your model, you can configure the code generation options using the 'Verilog' page of the 'Preferences' dialog to:

- Specify a default source directory
- Specify an editor for changing code

Access

Ribbon	Start > Appearance > Preferences > Preferences > Source Code Engineering > Verilog
Keyboard Shortcuts	Ctrl+F9

Options

Option	Action
Disable Language	Leave this checkbox unselected to support Verilog code generation.

	Select this option to disable Verilog code support.
Options for the current user	Specifies the options used for the current user; these options apply to all models that are accessed by the user.

Verilog Options - Model

If you intend to generate Verilog code from your model, you can configure the model-specific code generation options using the 'Verilog' page of the 'Manage Project Options' dialog to:

- Specify the default file extension
- Specify the Collection Class definitions for Association connectors

Access

Ribbon	Settings > Model > Options > Source Code Engineering > Verilog

Options

Option	Action
Options for the current model	Type in the default file extension to apply when generating Verilog source code.

Collection Classes	Click on this button to open the 'Collection Classes for Association Roles' dialog, through which you specify the Collection Class definitions for Association connectors.

Notes

- These options affect all users of the current model; however, they do not apply to other models

VHDL Options - User

If you intend to generate VHDL code from your model, you can configure the code generation options using the 'VHDL' page of the 'Preferences' dialog to:

- Specify a default source directory
- Specify an editor for changing code

Access

Ribbon	Start > Appearance > Preferences > Preferences > Source Code Engineering > VHDL
Keyboard Shortcuts	Ctrl+F9

Options

Option	Action
Disable Language	Leave this checkbox unselected to support VHDL code generation.

	Select this option to disable VHDL code support.
Options for the current user	Specifies the options used for the current user; these options apply to all models that are accessed by the user.

VHDL Options - Model

If you intend to generate VHDL code from your model, you can configure the model-specific code generation options using the 'VHDL' page of the 'Manage Project Options' dialog to:

- Specify the default file extension
- Specify the Collection Class definitions for Association connectors

Access

Ribbon	Settings > Model > Options > Source Code Engineering > VHDL

Options

Option	Action
Options for the current model	Type in the default file extension to apply when generating VHDL source code.

Collection Classes	Click on this button to open the 'Collection Classes for Association Roles' dialog, through which you specify the Collection Class definitions for Association connectors.

Notes

- These options affect all users of the current model; however, they do not apply to other models

Visual Basic Options - User

If you intend to generate Visual Basic code from your model, you can configure the code generation options using the 'VB Specifications' page of the 'Preferences' dialog to:

- Specify the default attribute type
- Define the default source directory
- Define the file extensions to search for code files to import
- Define the default editor to use for editing source code

Access

Ribbon	Start > Appearance > Preferences > Preferences > Source Code Engineering > Visual Basic
Keyboard Shortcuts	Ctrl+F9

Options

Option	Action
Disable Language	Leave this checkbox unselected to support Visual Basic code generation. Select this option to disable Visual Basic code support.
Options for the current user	Specifies the options used for the current use; these options apply to all models that are accessed by the user.

Visual Basic Options - Model

If you intend to generate Visual Basic code from your model, you can configure the model-specific code generation options using the 'VB Specifications' page of the 'Manage Project Options' dialog to:

- Specify the default Visual Basic version to generate
- Indicate the default file extension when reading/writing
- Indicate the Microsoft Transaction Server (MTS) transaction mode for MTS objects
- Specify if a Class uses Multi use (True or False)
- Specify if a Class uses the Persistable property
- Indicate data binding and data source behaviors
- Set the global namespace
- Set the Exposed attribute
- Indicate if the Creatable attribute is True or False
- Specify the Collection Class definitions for Association connectors

Access

Ribbon	Settings > Model > Options > Source Code Engineering > Visual Basic

Options

Option	Action
Options for the current model	Type in the default file extension to apply when generating Visual Basic source code, and click on the drop-down arrow in each of the other fields and select the appropriate value.
Collection Classes	Click on this button to open the 'Collection Classes for Association Roles' dialog, through which you specify the Collection Class definitions for Association connectors.

Notes

- These options affect all users of the current model; however, they do not apply to other models

MDG Technology Language Options

If you have loaded an MDG Technology that specifies a code module into your *Sparx Systems > EA > MDG Technologies* folder, the language is included in the 'Source Code Engineering' list on the 'Preferences' dialog. The language is only listed on the 'Preferences' dialog if an MDG Technology file actually uses it in your model.

Access

Ribbon	Start > Appearance > Preferences > Preferences > Source Code Engineering > MDG
Keyboard Shortcuts	Ctrl+F9

Options

Field	Action
Default	Default extension for generated source

Extension	files; shown if the option is in the technology. This is saved per project.
Import File Extensions	Default folder to import source files from; shown if the technology supports namespaces. This is saved once for all projects.
Generate Namespaces	Indicates if namespaces are generated or not.
Default Source Directory	The default directory to save generated source files. This is always shown.
Editor	Indicates the editor that is used to edit source files.
Att Type	Indicates the default attribute type.

Notes

- These options are set in the technology inside the <CodeOptions> tag of a code module, as shown:
  <CodeOption

name="DefaultExtension">.rb</CodeOption>

Reset Options

Enterprise Architect stores some of the options for a Class when it is first created. Some are global; for example, $LinkClass is stored when you first create the Class, so in existing Classes the global change in the 'Preferences' dialog will not automatically be picked up. You must modify the options for the existing Class.

Modify options for a single Class

Step	Action
1	Click on the Class to change, and select the 'Develop > Source Code > Generate > Generate Single Element' ribbon option. The 'Generate Code' dialog displays.
2	Click on the Advanced button. The 'Object Options' dialog displays.
3	Click on the 'Attributes/Operations' option.
4	Change the options, and click on the Close button to apply the changes.

Modify options for all Classes within a Package

Step	Action
1	Click on the Package in the Browser window, and select the 'Develop > Preferences > Options > Reset Source Language' ribbon option. The 'Manage Code Generation' dialog displays.
2	In the 'Where language is:' field, click on the drop-down arrow and select the language that you want to change from.
3	In the 'Convert to:' field, click on the drop-down arrow and select the language that you want to change to.
4	Select the checkbox against each option to apply to the changed Class elements in the Package: • Clear Filenames of the files to generate code to • Reset Default options on each Class • Process Child Packages under the selected Package

5	Click on the OK button to apply the changes.

Set Collection Classes

Using Enterprise Architect, you can define Collection Classes for generating code from Association connectors where the target role has a multiplicity setting greater than 1.

Tasks

Task	Detail
Defining Collection Classes	On the 'Source Code Engineering' section of the 'Manage Project Options' dialog (select the 'Settings > Model > Options > Source Code Engineering' ribbon option), on each language page click on the Collection Classes button. The 'Collection Classes for Association Roles' dialog displays. On this dialog, you can define: • The default Collection Class for 1..* roles • The ordered Collection Class to use for 1..* roles • The qualified Collection Class to use for 1..* roles

Defining Collection Classes for a specific Class	Class-specific Collection Classes can be defined by clicking the Collection Classes button in the Class 'Properties' dialog of the element.
Code Generation Precedence	When Enterprise Architect generates code for a connector that has a multiplicity role >1: 1. If the Qualifier is set, use the qualified collection: - for the Class if set - else use the code language qualified collection 2. If the 'Order' option is set, use the ordered collection: - for the Class if set - else use the code language ordered collection 3. Else use the default collection: - for the Class if set - else use the code language default collection
Using Markers	You can include the marker #TYPE# in the collection name; Enterprise Architect replaces this with the name of the Class being collected at source generation time (for example, Vector<#TYPE#> would

	become Vector<foo>). Conversely, when reverse engineering, an Association connector is also created if a matching entry (for example, foo if foo is found in the model) is defined as a Collection Class.
Additional Collection Classes	Additional Collection Classes can be defined within the model-specific language options pages for C#, C++ and Java.
Member Type	On the 'Role(s)' tab of the Association 'Properties' dialog (accessible from the right-click context menu of any Association) there is a 'Member Type' field for each of the Source and Target Roles. If you set this, the value you enter overrides all the listed options.

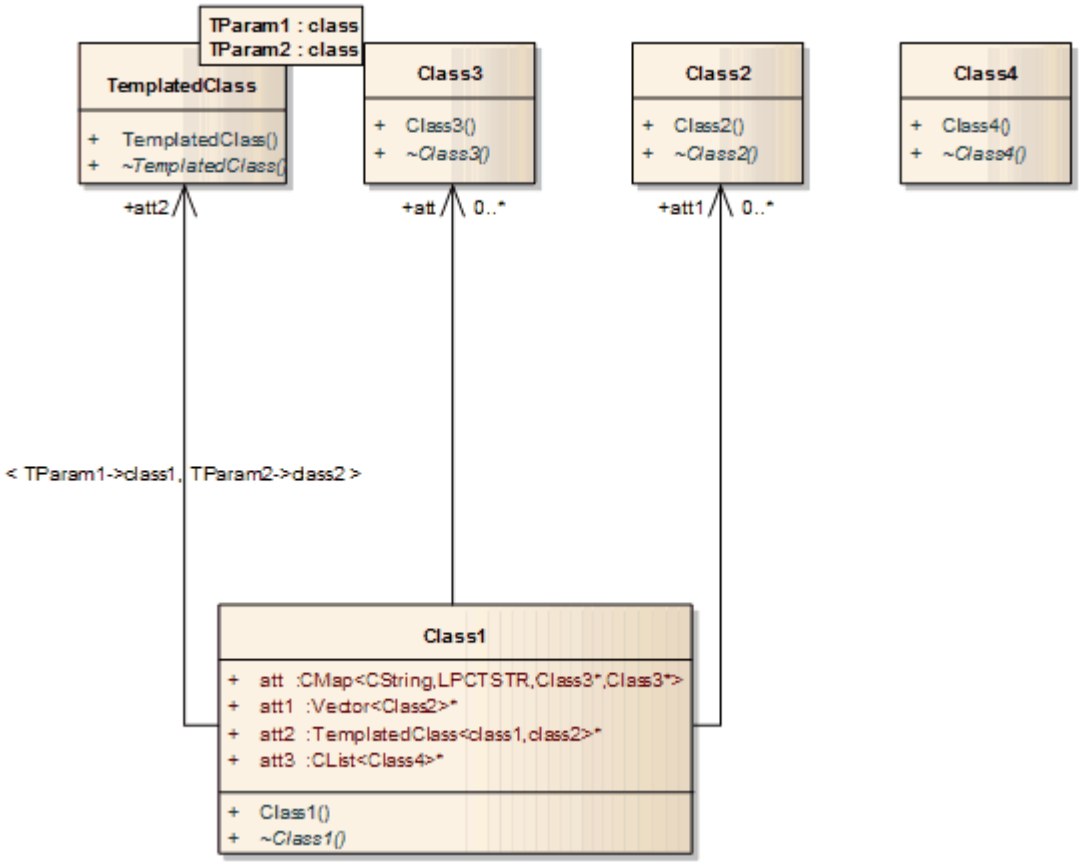
Example Use of Collection Classes

Consider this source code:

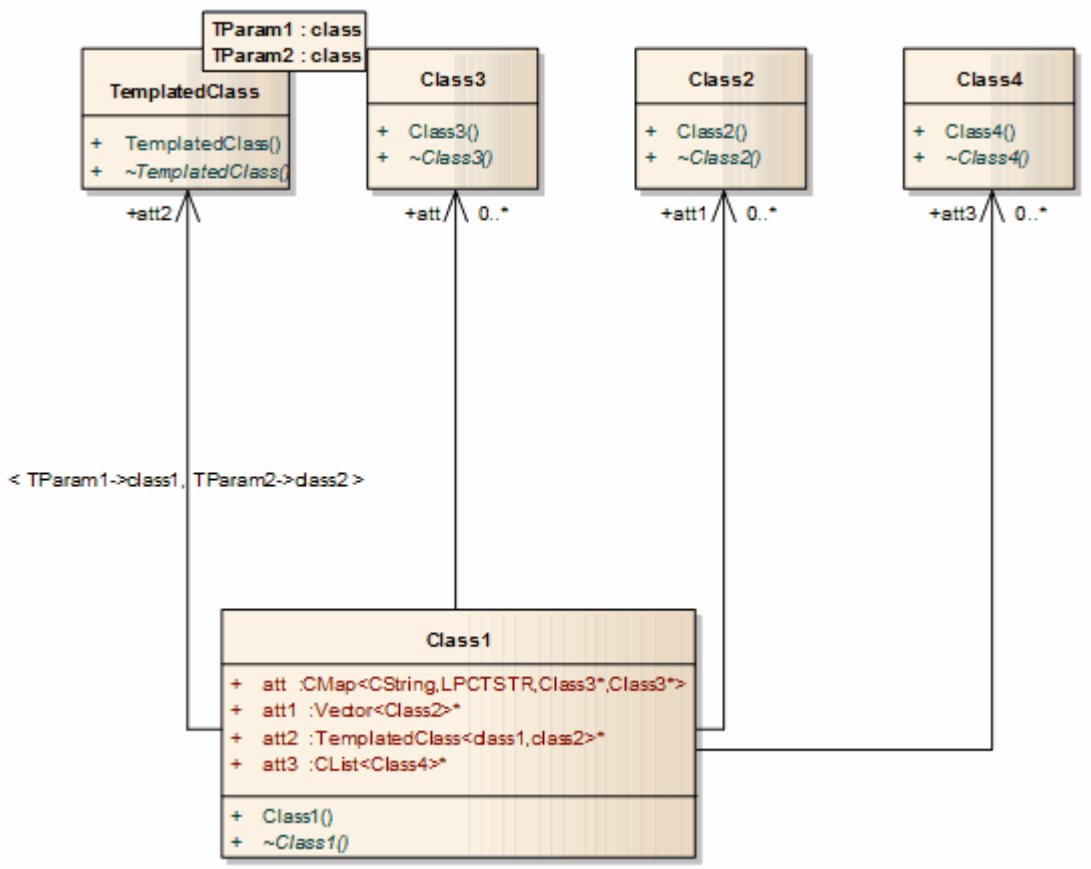
```
class Class1
{
public:
    Class1();
    virtual ~Class1();
    CMap<CString,LPCTSTR,Class3*,Class3*> att;
    Vector<Class2> *att1;
    TemplatedClass<class1,class2> *att2;
    CList<Class4> *att3;
};
class Class2
{
public:
    Class2();
    virtual ~Class2();
};
class Class3
{
public:
    Class3();
```

```
    virtual ~Class3();
};
class Class4
{
public:
    Class4();
    virtual ~Class4();
};
template<class TParam1, class TParam2>
class TemplatedClass
{
public:
    TemplatedClass() {
    }
    virtual ~TemplatedClass() {
    }
};
```

If this code is imported into the system with default import options, this diagram is generated:



If, however, you enter the value 'CList<#Type#>' in the 'Additional Collection Classes' field in the model-specific language options page (C#, Java, C++), an Association connector is also created to Class 4:


TParam1 : class
TParam2 : class
TemplatedClass
+ TemplatedClass()
+ ~TemplatedClass()
Class3
+ Class3()
+ ~Class3()
Class2
+ Class2()
+ ~Class2()
Class4
+ Class4()
+ ~Class4()
+att2
+att
0..*
+att1
0..*
+att3
0..*
< TParam1->class1, TParam2->class2 >
Class1
+ att :CMap<CString,LPCTSTR,Class3*,Class3*>
+ att1 :Vector<Class2>*
+ att2 :TemplatedClass<class1,class2>*
+ att3 :CList<Class4>*
+ Class1()
+ ~Class1()

Local Paths

When a team of developers are working on the same Enterprise Architect model, each developer might store their version of the source code in their local file system, but not always at the same location as their fellow developers. To manage this scenario in Enterprise Architect, you can define local paths for each user, on the 'Local Paths' dialog.

You can use local paths in generating code and reverse engineering, and in Version Control, developing XML schemas and generating document and web reports.

Local paths might take a little time to set up, but if you want to work collaboratively on source and model concurrently, the effort is well worth while.

For example, if:

- Developer A stores her .java files in a C:\Java\Source directory, while developer B stores his in D:\Source, and
- Both developers want to generate and reverse engineer into the same Enterprise Architect model located on a shared (or replicated) network drive

Developer A might define a local path of:

JAVA_SOURCE = "C:\Java\Source"

All Classes generated and stored in the Enterprise Architect project are stored as:

%JAVA_SOURCE%\<xxx.java>

Developer B defines a local path as:

JAVA_SOURCE ="D:\Source"

Now, Enterprise Architect stores all java files in these directories as:

%JAVA_SOURCE%\<filename>

On each developer's machine, the filename is expanded to the correct local version.

Access

Ribbon	Develop > Source Code > Options > Configure Local Paths

Local Paths Dialog

Using the 'Local Paths' dialog, you can set up local paths for a single user on a particular machine. For a description of the use of local paths, see the *Local Paths* topic.

Access

Ribbon	Develop > Source Code > Options > Configure Local Paths

Options

Option	Action
Path	Type in or browser for the path of the local directory in the file system (for example, d:\java\source).
ID	Type in the shared ID that is substituted for the Local Path (for example, JAVA_SRC).

Type	Click on the drop-down arrow and select the type of path to apply to (for example, Java).
Relative Paths	Lists the paths currently defined for the model, defaulting to most recent at the top. If you want to change the sequence of paths in the list, click on a path and use the  buttons to move the path up or down one position in the list.
Apply Path	Click on a path in the 'Relative Paths' list and click on this button to update any existing full path names in the model to the shared relative path name. For example: d:\java\source\main.java might become %JAVA_SRC%\main.java
Expand Path	Click on a path in the 'Relative Paths' list and click on this button to remove the relative path and substitute the full path name (the opposite effect of the Apply Path button).
New	Click on this button to clear the data fields so that you can define another local path.

Save	When you have defined a local path, click on this button to save it and add it to the 'Relative Paths' list.
Delete	Click on a path in the 'Relative Paths' list and click on this button to remove the path from the list altogether.
Close	Click on this button to close the dialog, saving any changes to the list.

Notes

- You can also set up a hyperlink (for an Enterprise Architect command) on a diagram to access the 'Local Paths' dialog, to switch, update or expand your current local path
- If the act of expanding or applying a path for a linked file will create a duplicate record, the process will skip that record and display a message at the end of the process

Language Macros

When reverse engineering a language such as C++, you might find preprocessor directives scattered throughout the code. This can make code management easier, but can hamper parsing of the underlying C++ language.

To help remedy this, you can include any number of macro definitions, which are ignored during the parsing phase of the reverse engineering. It is still preferable, if you have the facility, to preprocess the code using the appropriate compiler first; this way, complex macro definitions and defines are expanded out and can be readily parsed. If you don't have this facility, then this option provides a convenient substitute.

Access

Ribbon	Settings > Reference Data > Settings > Preprocessor Macros or Develop > Source Code > Options > Configure > Define Preprocessor Macros

Define a macro

Step	Action
1	Select the 'Preprocessor Macros' menu option. The 'Language Macros' dialog displays.
2	Click on the Add New button.
3	Enter details for your macro.
4	Click on the OK button.

Macros Embedded Within Declarations

Macros are sometimes used within the declaration of Classes and operations, as in these examples:

```
class __declspec Foo
{
int __declspec Bar(int p);
};
```

If declspec is defined as a C++ macro, as outlined, the imported Class and operation contain a Tagged Value called DeclMacro1 with value __declspec (subsequent macros would be defined as DeclMacro2, DeclMacro3 and so on).

During forward engineering, these Tagged Values are used

to regenerate the macros in code.

Define Complex Macros

It is sometimes useful to define rules for complex macros that can span multiple lines; Enterprise Architect ignores the entire code section defined by the rule.

Such macros can be defined in Enterprise Architect as in these two examples; both types can be combined in one definition.

Block Macros

BEGIN_INTERFACE_PART ^
END_INTERFACE_PART

The ^ symbol represents the body of the macro - this enables skipping from one macro to another; the spaces surrounding the ^ symbol are required.

Function Macros

RTTI_EMULATION()

Enterprise Architect skips over the token including everything inside the parentheses.

Function Macros can also include the function body:

RTTI_EMULATION() {}

In this case, Enterprise Architect skips over the token including everything inside the parentheses and inside the braces. Note that if the Function Macro includes the

function body, it cannot be combined with a Block Macro.

Notes

- You can transport these language macro (or preprocessor macro) definitions between models, using the 'Settings > Model > Transfer > Export Reference Data' and 'Import Reference Data' options; the macros are exported as a Macro List

Developing Programming Languages

You can make use of a range of established programming languages in Enterprise Architect, but if these are not suitable to your needs you can develop your own. You would then apply it to your models through an MDG Technology that you might develop just for this purpose, or for broader purposes. After developing the language, you could also write MDA Transformation templates to convert a Platform Independent Model or a model in another language into a model for your new language, or vice-versa.

Access

Ribbon	Develop > Source Code > Options > Edit Code Templates
Keyboard Shortcuts	Ctrl+Shift+P

Develop a Programming Language

Ste	Description

p	
1	In the Code Template Editor, click on the New Language button and, on the 'Programming Languages Datatypes' dialog, click on the Add Product button. Enter your new programming language name and define the datatypes for it. You cannot access the new language in the Code Template Editor until at least one datatype has been added to the language.
2	After you have defined all the datatypes you need, click on the Close button, select the language in the 'Language' field of the Code Template Editor, and start to edit or create the code templates for the new language. The code templates define how the system should perform: • Forward code engineering of your models in the new language • Behavioral Code generation (if this is appropriate)
3	If you prefer, you can also define source code options for your new language. These are additional settings for the language that are not provided by the data types or code templates, and that help define how the system handles that language when generating and reverse-engineering code. The code options are made available to your models

	only through an MDG Technology.
4	Defining a grammar for your language is an optional step that provides two primary benefits: • Reverse engineering of existing code into your model • Synchronization during code generation so that changes made to the file since it was last generated are not lost. To access the grammar editor select the 'Develop > Source Code > Grammar Editor' ribbon option.
5	If you intend MDA transformations to be made to (or from) your new programming language, you can also edit and create transformation templates for it. The process of creating transformation templates is very similar to that for creating code templates.
6	Having created the datatypes, code templates, code options, grammar and transformation templates for your new language, you can incorporate and distribute them in an MDG Technology.

Code Template Framework

When you use Enterprise Architect to generate code from a model, or transform the model, the system refers to the Code Template Framework (CTF) for the parameters that define how it should:

- Forward engineer a UML model
- Generate Behavioral Code
- Perform a Model Driven Architecture (MDA) Transformation
- Generate DDL in database modeling

A range of standard templates is available for the direct generation of code and for transformation; if you do not want to use the standard CTF configurations, you can customize them to meet your needs.

CTF Templates

Template Type	Detail
Code Templates	When you forward engineer a Class model, the code templates define how the skeletal code is to be generated for a given programming language. The templates for a language are

	automatically associated with the language. The templates are written as plain text with a syntax that shares some aspects of both mark-up languages and scripting languages.
Model Transformation Templates	Model Transformation Templates provide a fully configurable method of defining how Model Driven Architecture (MDA) Transformations convert model elements and model fragments from one domain to another. This process is two-tiered. It creates an intermediary language (which can be viewed for debugging) which is then processed to create the objects.
Behavioral Code Generation Templates	Enterprise Architect supports user-definable code generation of the UML Behavioral models. This applies the standard Code Template Framework but includes specific Enterprise Architect Simulation Library (EASL) code generation macros.
DDL Templates	DDL Templates are very similar to Code generation templates, but they have been extended to support DDL generation with

	their own set of base templates, macros, function macros and template options.

Code Template Customization

Enterprise Architect helps you to generate source code from UML models for a wide range of programming languages. Standard templates (mappings) are provided out-of-the-box but you can customize the way that code is generated by using the practical and flexible Code Template Framework (CTF). This sophisticated framework allows you to customize every detail of the way code is generated, including the facility to create new templates for languages not supported in the base product. For example, JavaScript is not one of the supported languages but a series of templates can be written quickly to generate JavaScript from UML models. In these cases existing templates act as a useful starting point and reference for new languages.

The code template framework also provides the mechanism for generation of behavioral models and is used for the transformation templates.

Features

Feature	Detail
Default Templates	Default Code Templates are built into Enterprise Architect for forward engineering supported languages.

Code Template Editor	A Code Template Editor is provided for creating and maintaining user-defined Code Templates.
Customizing Code Templates	Descriptions of the template syntax and the macros and functions you can use to control the effects of the templates.
Synchronize Code	A subset of the default Code Templates to synchronize code.

Code and Transform Templates

Code templates and transform (Model Transformation) templates define how the system should generate or transform code in one or other of the programming languages that Enterprise Architect supports. Each language has a wide range of base templates, each of which defines how a particular code structure is generated. You can use these base templates as they are, or you can customize and add to the templates to better support your use of the standard languages, or of other languages that you might define to the system. You review, update and create templates through the Code Template editor or Transformation Template editor.

The order in which the base templates are listed in the two editors relates to the hierarchical order of the objects and their parts that are to be processed. Calls are made from certain base templates to others, and you can add further calls to both base templates and to your own custom templates. By default, the File template is the starting point of a code generation process through the templates; a File consists of Classes that can contain Attributes and Operations.

Access

	Develop >Source Code > Options > Edit

Ribbon	Code Templates Design > Package > Transform > Transform Templates
Keyboard Shortcuts	Ctrl+Shift+P (Code Generation Templates) Ctrl+Alt+H (MDA Transformation Templates)

Application of Templates

Action	Detail
Calling Templates	Within any template, you can call other templates using %TemplateName%. The enclosing percent (%) signs indicate a macro. You would use this for a single call to the ClassBody template, %ClassBody%, as shown: % list = "TemplateName" @separator= "\n" @indent= " " % The %list macro performs an iterative pass on all the objects in the scope of the current template and calls the

	TemplateName for each of them:
	% list = "ClassBody" @separator= "\n" @indent= " " %
	After generation or transformation, each macro is substituted to produce the generated output; for a language such as C++, the result of processing this template might be:
	/**
	* This is an example Class note generated using code templates
	* @author Sparx Systems
	*/
	class ClassA: public ClassB
	{
	...
	}

Execution of Code Templates	Each template might act only on a particular element type; for example, the ClassNotes template only acts on UML Class and Interface elements. The element from which code is currently being generated is said to be in scope; if the element in scope is stereotyped, the system searches for a template that has been defined for that stereotype. If a specialized template is found, it is

	executed; otherwise the default implementation of the base template is used. Templates are processed sequentially, line by line, replacing each macro with its underlying text value from the model.
Transfer Templates Between Projects	If you edit a base Code Generation or Transformation template, or create a customized template, you can copy them from one project to another as Reference Data.

Base Templates

The Code Template Framework consists of a number of base templates. Each base template transforms particular aspects of the UML to corresponding parts of object-oriented languages.

The base templates form a hierarchy, which varies slightly across different programming languages. In a typical template hierarchy relevant to a language such as C# or Java (which do not have header files) the templates can be modeled as Classes, but usually are just plain text. This hierarchy would be slightly more complicated for languages such as C++ and Delphi, which have separate implementation templates.

Each of the base templates must be specialized to be of use in code engineering; in particular, each template is specialized for the supported languages (or 'products'). For example, there is a ClassBody template defined for C++, another for C#, another for Java, and so on; by specializing the templates, you can tailor the code generated for the corresponding UML entity.

Once the base templates are specialized for a given language, they can be further specialized based on:

- A Class's stereotype, or
- A feature's stereotype (where the feature can be an operation or attribute)

This type of specialization enables, for example, a C# operation that is stereotyped as «property» to have a

different Operation Body template from an ordinary operation; the Operation Body template can then be specialized further, based on the Class stereotype.

Base templates used in the CTF

Template	Description
Attribute	A top-level template to generate member variables from UML attributes.
Attribute Declaration	Used by the Attribute template to generate a member variable declaration.
Attribute Notes	Used by the Attribute template to generate member variable notes.
Class	A top-level template for generating Classes from UML Classes.
Class Base	Used by the Class template to generate a base Class name in the inheritance list of a derived Class, where the base Class doesn't exist in the model.
Class Body	Used by the Class template to generate the body of a Class.

Class Declaration	Used by the Class template to generate the declaration of a Class.
Class Interface	Used by the Class template to generate an interface name in the inheritance list of a derived Class, where the interface doesn't exist in the model.
Class Notes	Used by the Class template to generate the Class notes.
File	A top-level template for generating the source file. For languages such as C++, this corresponds to the header file.
Import Section	Used in the File template to generate external dependencies.
Linked Attribute	A top-level template for generating attributes derived from UML Associations.
Linked Attribute Notes	Used by the Linked Attribute template to generate the attribute notes.
Linked Attribute	Used by the Linked Attribute template to generate the attribute declaration.

Declaration	
Linked Class Base	Used by the Class template to generate a base Class name in the inheritance list of a derived Class, for a Class element in the model that is a parent of the current Class.
Linked Class Interface	Used by the Class template to generate an Interface name in the inheritance list of a derived Class, for an Interface element in the model that is a parent of the current Class.
Namespace	A top-level template for generating namespaces from UML Packages (although not all languages have namespaces, this template can be used to generate an equivalent construct, such as Packages in Java).
Namespace Body	Used by the Namespace template to generate the body of a namespace.
Namespace Declaration	Used by the Namespace template to generate the namespace declaration.
Operation	A top-level template for generating operations from a UML Class's

	operations.
Operation Body	Used by the Operation template to generate the body of a UML operation.
Operation Declaration	Used by the Operation template to generate the operation declaration.
Operation Notes	Used by the Operation template to generate documentation for an operation.
Parameter	Used by the Operation Declaration template to generate parameters.

Templates for generating code for languages with separate interface and implementation sections

Template	Description
Class Impl	A top-level template for generating the implementation of a Class.
Class Body Impl	Used by the Class Impl template to generate the implementation of Class

	members.
File Impl	A top-level template for generating the implementation file.
File Notes Impl	Used by the File Impl template to generate notes in the source file.
Import Section Impl	Used by the File Impl template to generate external dependencies.
Operation Impl	A top-level template for generating operations from a UML Class's operations.
Operation Body Impl	Used by the Operation template to generate the body of a UML operation.
Operation Declaration Impl	Used by the Operation template to generate the operation declaration.
Operation Notes Impl	Used by the Operation template to generate documentation for an operation.

Export Code Generation and Transformation Templates

It is possible to export Code Generation and Transformation templates from your model to a .xml file. You can then import that file - and hence the templates - into other models, as reference data. You can export customized templates, which includes those that you or other users have created and updated, and base (standard) templates that have been tailored. You do not need to export base templates that have not been changed, as these are available in every installation of Enterprise Architect.

Access

Ribbon	Settings > Model > Transfer > Export Reference Data

Export a Code Generation template or Transformation template

Ste	Action

p	
1	On the 'Export Reference Data' dialog, in the 'Name' list, select the templates to export. The list includes any standard Code Generation or Transformation templates that have been changed, and any customized templates that you have created or changed. You can select one or more templates to be exported to a single XML file, by pressing Ctrl or Shift as you click on the template names.
2	Click on the Export button.
3	When prompted to do so, enter a valid file name with a .xml extension.
4	Click on the Save button and on the OK button. This exports the template(s) to the file; you can use any text or XML viewer to examine the file.

Import Code Generation and Transformation Templates

If you have exported Code Generation and/or Transformation templates from an Enterprise Architect model, you can import them into other Enterprise Architect models as reference data.

Access

Ribbon	Settings > Model > Transfer > Import Reference Data

Import Code Generation and/or Transformation Templates

Step	Action
1	On the 'Import Reference Data' dialog, click on the Select File button and browse to the .xml file containing the required Code Generation or

	Transformation templates.
2	Select the name of one or more template datasets and click on the Import button.

Synchronize Code

Enterprise Architect uses code templates during the forward synchronization of these programming languages:

- ActionScript
- C
- C++
- C#
- Delphi
- Java
- PHP
- Python
- VB
- VB.Net

Three types of change can occur in the source when it is synchronized with the UML model:

- Existing sections are synchronized: for example, the return type in an operation declaration is updated
- New sections are added to existing features: for example, Notes are added to a Class declaration where there were previously none
- New features and elements are added: for example, a new operation is added to a Class

Each of these changes has a different effect on the CTF and must be handled differently by Enterprise Architect, as described in these topics:

- *Synchronize Existing Sections*
- *Add New Sections to Existing Features*
- *Add New Features and Elements*

Code sections that can be synchronized

Only a subset of the CTF base templates is used during synchronization. This subset corresponds to the distinct sections that Enterprise Architect recognizes in the source code.

Code Template	Code Section
Class Notes	Comments preceding the Class declaration.
Class Declaration	Up to and including the Class parents.
Attribute Notes	Comments preceding an Attribute declaration.
Attribute Declaration	Up to and including the terminating character.
Operation Notes	Comments preceding an operation declaration.

Operation Notes Impl	As for Operation Notes.
Operation Declaration	Up to and including the terminating character.
Operation Declaration Impl	Up to and including the terminating character.
Operation Body	Everything between and including the braces.
Operation Body Impl	As for Operation Body.

Synchronize Existing Sections

When an existing section in the source code differs from the result generated by the corresponding template, that section is replaced.

Consider, for example, this C++ Class declaration:

(asm) class A: public B

Now assume that you add an inheritance relationship from Class A to Class C; the entire Class declaration would be replaced with something resembling this:

(asm) class A: public B, public C

Add New Sections

These sections can be added to existing features in the source code, as new sections:

- Class Notes
- Attribute Notes
- Operation Notes
- Operation Notes Impl
- Operation Body
- Operation Body Impl

Assume that, in this example, Class A had no note when you originally generated the code:

```
(asm) class A: public B, public C
```

If you now specify a note in the model for Class A, Enterprise Architect attempts to add the new note from the model during synchronization, by executing the Class Notes template.

To make room for the new section to be inserted, you can specify how much white space to append to the section via synchronization macros.

Add New Features and Elements

These features and elements can be added to the source code during synchronization:

- Attributes
- Inner Classes
- Operations

They are added by executing the relevant templates for each new element or feature in the model.

Enterprise Architect attempts to preserve the appropriate indenting of new features in the code, by finding the indents specified in list macros of the Class; for languages that make use of namespaces, the 'synchNamespaceBodyIndent' macro is available.

Classes defined within a (non-global) namespace are indented according to the value set for this macro, during synchronization.

The value is ignored:

- For Classes defined within a Package set up as a root namespace, or
- If the 'Generate Namespaces' option is set to False in the appropriate language page (C#, C++ or VB.Net) on the 'Preferences' dialog ('Start > Appearance > Preferences > Preferences > Source Code Engineering > <language>')

The Code Template Editor

The Code Template Editor provides the facilities of the Common Code Editor, including Intelli-sense for the various macros. For more information on Intelli-sense and the Common Code Editor, see the *Editing Source Code* topic.

Access

Ribbon	Develop > Source Code > Options > Edit Code Templates
Keyboard Shortcuts	Ctrl+Shift+P

Options

Option	Action
Language	Select the programming language.
New Language	Display the 'Programming Languages Datatypes' dialog, which enables you to

	include programming languages other than those supported for Enterprise Architect, for which to create or edit code templates.
Template	Display the contents of the active template, and open the editor for modifying templates.
Templates	List the base code templates; the active template is highlighted. The 'Modified' field indicates whether you have changed the default template for the current language.
Stereotype Overrides	List the stereotyped templates, for the active base template. The 'Modified' field indicates whether you have modified a default stereotyped template.
Add New Custom Template	Invoke a dialog for creating a custom stereotyped template.
Add New Stereotyped Override	Invoke a dialog for adding a stereotyped template, for the currently selected base template.

Get Default Template	Update the editor display with the default version of the active template.
Save	Overwrite the active templates with the contents of the editor.
Delete	If you have overridden the active template, the override is deleted and replaced by the corresponding default code template.

Notes

- User-modified and user-defined Code Templates can be imported and exported as reference data (see the *Sharing Reference Data* topic); the templates defined for each language are indicated in the 'Export Reference Data' dialog by the language name with the suffix _Code_Templates - if no templates exist for a language, there is no entry for the language in the dialog

Create New Custom Template

The Create New Custom Template dialog provides the ability to create a custom template for the current Programming or Database Management System (DBMS) Language, depending on what information is being edited with the Code Template Editor.

When this dialog is loaded you will be prompted to enter a value for Template Type and Template Name. In order to save a new template both Type and Name are required.

Options

Option	Action
Template Type	Choose the type of Template for the new Custom Template.
Template Name	Enter a Name for the new Custom Template.
OK	Save the details of the new Custom Template.
Cancel	Close the Create New Custom Template dialog and loose any unsaved changes.

Note:

All templates of type "<none>" are treated as functions, therefore Enterprise Architect will automatically remove all space characters entered into the Name.

Code Template Syntax

Code Templates are written using Enterprise Architect's Code Template Editor. The Code Template Editor supports syntax highlighting of the Code Template Framework language.

Syntax Elements

Elements	Detail
Basic Constructs	Templates can contain: • Literal Text • Variables • Macros • Calls to other templates
Comments	If you want to add comments to the templates, use the command: $COMMENT="text" where "text" is the text of the comment; this must be enclosed in quotes. The command is case-sensitive, and must be typed in upper case.

Literal Text

All text within a given template that is not part of a macro or a variable definition/reference, is considered literal text. With the exception of blank lines, which are ignored, literal text is directly substituted from the template into the generated code.

Consider this excerpt from the Java Class Declaration template:

```
$bases = "Base"
class % className % $bases
```

On the final line, the word 'class ', including the subsequent space, would be treated as literal text and thus for a Class named 'foo' would return the output:

```
class fooBase
```

A blank line following the variable $bases would have no effect on the output.

Inserting System Characters:

The %, $, " and \ characters have special meaning in the template syntax and cannot always be used as literal text. If these characters must be generated from within the templates, they can be safely reproduced using these direct substitution macros:

Macro	Action

%dl%	Produce a literal $ character.
%pc%	Produce a literal % character.
%qt%	Produce a literal " character.
%sl%	Produce a literal \ character

Notes

String conjunction operators (“+”, “+=”) are not required but can be used

Variables

Template variables provide a convenient way of storing and retrieving data within a template. This section explains how variables are defined and referenced.

Variable Definitions

Variable definitions take the basic form:

$<name> = <value>

where <name> can be any alpha-numeric sequence and <value> is derived from a macro or another variable.

A simple example definition would be:

$foo = %className%

Variables can be defined using values from:

- Substitution, function or list macros
- String literals, enclosed within double quotation marks
- Variable references

Definition Rules

These rules apply to variable definitions:

- Variables have global scope within the template in which they are defined and are not accessible to other templates

- Each variable must be defined at the start of a line, without any intervening white space
- Variables are denoted by prefixing the name with $, as in $foo
- Variables do not have to be declared, prior to being defined
- Variables must be defined using either the assignment operator (=), or the addition-assignment operator (+=)
- Multiple terms can be combined in a single definition using the addition operator (+)

Examples

Using a substitution macro:

```
$foo = %opTag:"bar"%
```

Using a literal string:

```
$foo = "bar"
```

Using another variable:

```
$foo = $bar
```

Using a list macro:

```
$ops = %list="Operation" @separator="\n\n" @indent="\t"%
```

Using the addition-assignment operator (+=):

```
$body += %list="Operation" @separator="\n\n" @indent="\t"%
```

That definition is equivalent to:

```
$body = $body + %list="Operation" @separator="\n\n" @indent="\t"%
```

Using multiple terms:

```
$templateArgs = %list="ClassParameter" @separator=", "%
$template ="template<" + $templateArgs + ">"
```

Variable References

Variable values can be retrieved by using a reference of the form:

```
$<name>
```

where <name> can be a previously defined variable.

Variable references can be used:

- As part of a macro, such as the argument to a function macro
- As a term in a variable definition
- As a direct substitution of the variable value into the output

It is legal to reference a variable before it is defined. In this case, the variable is assumed to contain an empty string value: ""

Variable References - Example 1

Using variables as part of a macro. This is an excerpt from the default C++ ClassNotes template.

```
$wrapLen = %genOptWrapComment%
$style = %genOptCPPCommentStyle%     (Define variables to store the style and wrap length options)
%if $style == "XML.NET"%     (Reference to $style as part of a condition)
%XML_COMMENT($wrapLen)%
%else%
%CSTYLE_COMMENT($wrapLen)%     (Reference to $wrapLen as an argument to function macro)
%endIf%
```

Variable References - Example 2

Using variable references as part of a variable definition.

```
$foo = "foo"          (Define our variables)
$bar = "bar"
$foobar = $foo + $bar     ($foobar now contains the value foobar)
```

Variable References - Example 3

Substituting variable values into the output.

$bases=%classInherits% (Store the result of the ClassInherits template in $bases)

Class %className%$bases (Now output the value of $bases after the Class name)

Macros

Macros provide access to element fields within the UML model and are also used to structure the generated output. All macros are enclosed within percent (%) signs, as shown:

%<macroname>%

In general, macros (including the % delimiters) are substituted for literal text in the output. For example, consider this item from the Class Declaration template:

... class %className% ...

The field substitution macro, %className%, would result in the current Class name being substituted in the output. So if the Class being generated was named Foo, the output would be:

... class Foo ...

The CTF contains a number of types of macro:

- Template Substitution Macros
- Field Substitution Macros
- Substitution Examples
- Attribute Field Substitution Macros
- Class Field Substitution Macros
- Code Generation Option Field Substitution Macros
- Connector Field Substitution Macros
- Constraint Field Substitution Macros
- Effort Field Substitution Macros

- File Field Substitution Macros
- File Import Field Substitution Macros
- Link Field Substitution Macros
- Linked File Field Substitution Macros
- Metric Field Substitution Macros
- Operation Field Substitution Macros
- Package Field Substitution Macros
- Parameter Field Substitution Macros
- Problem Field Substitution Macros
- Requirement Field Substitution Macros
- Resource Field Substitution Macros
- Risk Field Substitution Macros
- Scenario Field Substitution Macros
- Tagged Value Substitution Macros
- Template Parameter Substitution Macros
- Test Field Substitution Macros
- Function Macros
- Control Macros
- List Macro
- Branching Macros
- Synchronization Macros
- The Processing Instruction (PI) Macro
- EASL Code Generation Macros

Template Substitution Macros

Template substitution macros correspond to Base templates, and result in the execution of the named template. By convention, template macros are named according to Pascal casing.

Structure: %<TemplateName>%

where <TemplateName> can be one of the templates listed in this topic.

When a template is referenced from within another template, it is generated with respect to the elements currently in scope. The specific template is selected based on the stereotypes of the elements in scope.

As noted previously, there is an implicit hierarchy among the various templates. Some care should be taken in order to preserve a sensible hierarchy of template references. For example, it does not make sense to use the %ClassInherits% macro within any of the Attribute or Operation templates. Conversely, the Operation and Attribute templates are designed for use within the ClassBody template.

Template substitution macros in the CTF

- Attribute
- AttributeDeclaration
- AttributeDeclarationImpl
- AttributeNotes

- Class
- ClassBase
- ClassBody
- ClassBodyImpl
- ClassDeclaration
- ClassDeclarationImpl
- ClassImpl
- ClassInherits
- ClassInterface
- ClassNotes
- ClassParameter
- File
- FileImpl
- ImportSection
- ImportSectionImpl
- InnerClass
- InnerClassImpl
- LinkedAttribute
- LinkedAttributeDeclaration
- LinkedAttributeNotes
- LinkedClassBase
- LinkedClassInterface
- Namespace
- NamespaceBody
- NamespaceDeclaration

- NamespaceImpl
- Operation
- OperationBody
- OperationBodyImpl
- OperationDeclaration
- OperationDeclarationImpl
- OperationImpl
- OperationNotes
- Parameter

Field Substitution Macros

The field substitution macros provide access to data in your model. In particular, they are used to access data fields from:

- Packages
- Classes
- Attributes
- Operations, and
- Parameters

Field substitution macros are named according to Camel casing. By convention, the macro is prefixed with an abbreviated form of the corresponding model element. For example, attribute-related macros begin with att, as in the %attName% macro, to access the name of the attribute in scope.

Macros that represent checkboxes return a value of T if the box is selected. Otherwise the value is empty.

This table lists a small number of project field substitution macros. Type-specific macros are listed in the subtopics of this *Field Substitution Macros* section.

Project Macros

Macro Name	Description

eaDateTime	The current time with format: DD-MMM-YYYY HH:MM:SS AM/PM.
eaGUID	A unique GUID for this generation.
eaVersion	Program Version (located in the 'About Enterprise Architect' dialog by selecting 'Start > Help > Help > About EA').

Substitution Examples

Field substitution macros can be used in one of two ways:

- Direct Substitution or
- Conditional Substitution

Direct Substitution

This form directly substitutes the corresponding value of the element in scope into the output.

Structure: %<macroName>%

Where <macroName> can be any of the macros listed in the Field Substitution Macros tables.

Examples

- %className%
- %opName%
- %attName%

Conditional Substitution

This form of the macro enables alternative substitutions to

be made depending on the macro's value.

Structure: %<macroName> (== "<text>") ? <subTrue> (: <subFalse>) %

Where:

- () denotes that values between the parentheses are optional
- <text> is a string representing a possible value for the macro
- <subTrue> and <subFalse> can be a combination of quoted strings and the keyword value; where the value is used, it is replaced with the macro's value in the output

Examples

- %classAbstract=="T" ? "pure" :""%
- %opStereotype=="operator" ? "operator" :""%
- %paramDefault != "" ? " = " value : ""%

These three examples output nothing if the condition fails. In this case the False condition can be omitted, resulting in this usage:

- %classAbstract=="T" ? "pure"%
- %opStereotype=="operator" ? "operator"%
- %paramDefault != "" ? " = "value%

The third example of both blocks shows a comparison checking for a non-empty value or existence. This test can

also be omitted.

- %paramDefault ? " = " value : ""%
- %paramDefault ? " = " value%

All of these examples containing paramDefault are equivalent. If the parameter in scope had a default value of 10, the output from each of them would normally be:

= 10

Notes

- In a conditional substitution macro, any white space following <macroName> is ignored; if white space is required in the output, it should be included within the quoted substitution strings

Attribute Field Substitution Macros

This table lists each of the attribute field substitution macros.

Field substitution macros are named according to Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected. Otherwise the value is empty.

Attribute Macros

Macro Name	Description
attAlias	'Attributes' dialog: Alias.
attAllowDuplicates	'Attributes Detail' dialog: 'Allow Duplicates' checkbox.
attClassifierGUID	The unique GUID for the classifier of the current attribute.
attCollection	'Attributes Detail' dialog: 'Attribute is a Collection' checkbox.
attConst	'Attributes' dialog: 'Const' checkbox.
attContainerType	'Attributes Detail' dialog: Container Type.

attContainment	'Attributes' dialog: Containment.
attDerived	'Attributes' dialog: 'Derived' checkbox.
attGUID	The unique GUID for the current attribute.
attInitial	'Attributes' dialog: Initial.
attIsEnumLiteral	'Attributes' dialog: 'Is Literal' checkbox.
attIsID	'Attributes Detail' dialog: 'isID' checkbox.
attLength	'Column' dialog: Length.
attLowerBound	'Attributes Detail' dialog: Lower Bound.
attName	'Attributes' dialog: Name.
attNotes	'Attributes' dialog: Notes.
attOrderedMultiplicity	'Attributes Detail' dialog: 'Ordered Multiplicity' checkbox.

attProperty	'Attributes' dialog: 'Property' checkbox.
attQualType	The attribute type qualified by the namespace path (if generating namespaces) and the classifier path (dot delimited). If the attribute classifier has not been set, is equivalent to the attType macro.
attScope	'Attributes' dialog: Scope.
attStatic	'Attributes' dialog: 'Static' checkbox.
attStereotype	'Attributes' dialog: Stereotype.
attType	'Attributes' dialog: Type.
attUpperBound	'Attributes Detail' dialog: Upper Bound.
attVolatile	'Attributes Detail' dialog: 'Transient' checkbox.

Class Field Substitution Macros

This table provides a list of methods for accessing each available Class property in the Code Generation and Transformation templates.

Field substitution macros are named according to Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected. Otherwise the value is empty.

Class Macros

Macro Name	Description
elemType	The element type: Interface or Class.
classAbstract	Class 'Properties' dialog: 'Abstract' checkbox ('Details' tab).
classAlias	Class 'Properties' dialog: 'Alias' field.
classArguments	Class 'Detail' dialog: C++ Templates: Arguments.
classAuthor	Class 'Properties' dialog: 'Author' field.
classBaseName	'Type Hierarchy' dialog: Class Name (for use where no connector exists between

	child and base Classes).
classBaseScope	The scope of the inheritance as reverse engineered. (For use where no connector exists between child and base Classes.)
classBaseVirtual	The virtual property of the inheritance as reverse engineered. (For use where no connector exists between child and base Classes.)
classComplexity	Class 'Properties' dialog: 'Complexity' field.
classCreated	The date and time the Class was created.
classGUID	The unique GUID for the current Class.
classHasConstructor	Looks at the list of methods in the current object and, depending on the conventions of the current language, returns T if one is a default constructor. Typically used with the genOptGenConstructor macro.
classHasCopyConstructor	Looks at the list of methods in the current object and, depending on the conventions of the current language, returns T if one is a copy constructor. Typically used with the genOptGenCopyConstructor macro.

classHasDestructor	Looks at the list of methods in the current object and, depending on the conventions of the current language, returns T if one is a destructor. Typically used with the genOptGenDestructor macro.
classHasParent	True, if the Class in scope has one or more base Classes.
classHasStereotype	True, if the Class in scope has a stereotype that matches a stereotype name (which you can optionally specify as fully qualified). It therefore checks all stereotypes that a Class has and returns 'T' if any of them is the specified stereotype or a specialization of it. For example: • %classHasStereotype:"block"% will return 'T' for any block-stereotyped Class from any SysML version, including associationBlock • %classHasStereotype:"SysML1.4::block"% will specifically match the SysML 1.4 versions Compare this with classStereotype, later.
classImports	'Code Gen' dialog: Imports.

classIsActive	Class 'Advanced' dialog: 'Is Active' checkbox.
classIsAssociationClass	True, if the Association is an AssociationClass connector.
classIsInstantiated	True, if the Class is an instantiated template Class.
classIsLeaf	Class 'Advanced' dialog: 'Is Leaf' checkbox.
classIsRoot	Class 'Advanced' dialog: 'Is Root' checkbox.
classIsSpecification	Class 'Advanced' dialog: 'Is Specification' checkbox.
classKeywords	Class 'Properties' dialog: 'Keywords' field.
classLanguage	Class 'Properties' dialog: 'Language' field.
classMacros	A space separated list of macros defined for the Class.
classModified	The date and time the Class was last modified.

classMultiplicity	Class 'Advanced' dialog: Multiplicity.
className	Class 'Properties' dialog: 'Name' field.
classNotes	Class 'Properties' dialog: 'Note' field.
classParamDefault	Class 'Detail' dialog.
classParamName	Class 'Detail' dialog.
classParamType	Class 'Detail' dialog.
classPersistence	Class 'Properties' dialog: 'Persistence' field ('Details' tab)
classPhase	Class 'Properties' dialog: 'Phase' field.
classQualName	The Class name prefixed by its outer Classes. Class names are separated by double colons (::).
classScope	Class 'Properties' dialog: 'Scope' field.
classStereoty	Class 'Properties' dialog: 'Stereotype'

pe	field. Retrieves the name of the first stereotype applied to the Class. When used in a comparison, it checks whether that first stereotype exactly matches a string. For example: %classStereotype=="enumeration" ? "enum" : "class"% Compare this with classHasStereotype, earlier.
classStatus	Class 'Properties' dialog: 'Status' field.
classVersion	Class 'Properties' dialog: 'Version' field.

Code Generation Option Field Substitution Macros

Code generation option field substitution macros operate on the source code generation options defined in the 'Source Code Engineering' pages of either the:

- 'Preferences' dialog ('Start > Appearance > Preferences > Preferences > Source Code Engineering') for user-specific options, or
- 'Manage Project Options' dialog ('Settings > Model > Options') for model-specific options

For more information on the division of the options, see the *Source Code Engineering Options* topic.

Field substitution macros are named according to Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected. Otherwise the value is empty. This table lists each of the code generation option field substitution macros.

Code Generation Option Macros

Macro Name	Description
genOptActionScriptVersion	ActionScript Specifications page: Default Version.

genOptCDefaultAttributeType	C Specifications page: Default Attribute Type.
genOptCGenMethodNotesInBody	C Specifications page: Method Notes In Implementation.
genOptCGenMethodNotesInHeader	C Specifications page: Method Notes In Header.
genOptCSynchNotes	C Specifications page: Synchronize Notes in Generation.
genOptCSynchCFile	C Specifications page: Synchronise Implementation file in Generation.
genOptCDefaultSourceDirectory	C Specifications page: Default Source Directory.
genOptCNamespaceDelimiter	C Specifications page: Namespace Delimiter.
genOptCOperationRefPara	C Specifications page: Reference as Operation Parameter.

m	
genOptCOperationRefParamStyle	C Specifications page: Reference Parameter Style.
genOptCOperationRefParamName	C Specifications page: Reference Parameter Name.
genOptCConstructorName	C Specifications page: Default Constructor Name.
genOptCDestructorName	C Specifications page: Default Destructor Name.
genOptCPPCommentStyle	C++ Specifications page: Comment Style.
genOptCPPDefaultAttributeType	C++ Specifications page: Default Attribute Type.
genOptCPPDefaultReferenceType	C++ Specifications page: Default Reference Type.
genOptCPPDefaultSource	C++ Specifications page: Default Source Directory.

Directory	
genOptCPPGenMethodNotesInHeader	C++ Specifications page: 'Method Notes In Header' checkbox.
genOptCPPGenMethodNotesInBody	C++ Specifications page: Method Notes In Body checkbox.
genOptCPPGetPrefix	C++ Specifications page: Get Prefix.
genOptCPPHeaderExtension	C++ Specifications page: Header Extension.
genOptCPPSetPrefix	C++ Specifications page: Set Prefix.
genOptCPPSourceExtension	C++ Specifications page: Source Extension.
genOptCPPSynchNotes	C++ Specifications page: Synchronize Notes.
genOptCPPSynchCPPFile	C++ Specifications page: Synchronize CPP File.

genOptCSDefaultAttributeType	C# Specifications page: Default Attribute Type.
genOptCSSourceExtension	C# Specifications page: Default file extension.
genOptCSGenDispose	C# Specifications page: Generate Dispose.
genOptCSGenFinalizer	C# Specifications page: Generate Finalizer.
genOptCSGenNamespace	C# Specifications page: Generate Namespace.
genOptCSDefaultSourceDirectory	C# Specifications page: Default Source Directory.
genOptDefaultAssocAttName	Source Code Engineering page: Default name for associated attribute.
genOptDefaultConstructorScope	Object Lifetimes page: Default Constructor Visibility.

genOptDefaultCopyConstructorScope	Object Lifetimes page: Default Copy Constructor Visibility.
genOptDefaultDatabase	Code Editors page: Default Database.
genOptDefaultDestructorScope	Object Lifetimes page: Default Destructor Constructor Visibility.
genOptGenCapitalisedProperties	'Source Code Engineering' page: 'Capitalize Attribute Names for Properties' checkbox.
genOptGenComments	'Source Code Engineering' page: 'Comments - Generate' checkbox.
genOptGenConstructor	Object Lifetimes page: 'Generate Constructor' checkbox.
genOptGenConstructorInline	Object Lifetimes page: 'Constructor Inline' checkbox.
genOptGenCopyConstructor	Object Lifetimes page: 'Generate Copy Constructor' checkbox.

genOptGenCopyConstructorInline	Object Lifetimes page: 'Copy Constructor Inline' checkbox.
genOptGenDestructor	Object Lifetimes page: 'Generate Destructor' checkbox.
genOptGenDestructorInline	Object Lifetimes page: 'Destructor Inline' checkbox.
genOptGenDestructorVirtual	Object Lifetimes page: 'Virtual Destructor' checkbox.
genOptGenImplementedInterfaceOps	'Code Generation' page: 'Generate methods for implemented interfaces' checkbox.
genOptGenPrefixBoolProperties	'Source Code Engineering' page: 'Use 'Is' for Boolean property Get()' checkbox.
genOptGenRoleNames	'Source Code Engineering' page: 'Autogenerate role names when creating code' checkbox.
genOptGenU	'Source Code Engineering' page: 'Do not

nspecAssocDir	generate members where Association direction is unspecified' checkbox.
genOptJavaDefaultAttributeType	Java Specifications page: Default attribute type.
genOptJavaGetPrefix	Java Specifications page: Get Prefix.
genOptJavaDefaultSourceDirectory	Java Specifications page: Default Source Directory.
genOptJavaSetPrefix	Java Specifications page: Set Prefix.
genOptJavaSourceExtension	Java Specifications page: Source code extension.
genOptPHPDefaultSourceDirectory	PHP Specifications page: Default Source Directory.
genOptPHPGetPrefix	PHP Specifications page: Get Prefix.
genOptPHPS	PHP Specifications page: Set Prefix.

etPrefix	
genOptPHPSourceExtension	PHP Specifications page: Default file extension.
genOptPHPVersion	PHP Specifications page: PHP Version.
genOptPropertyPrefix	'Source Code Engineering' page: Remove prefixes when generating Get/Set properties.
genOptVBMultiUse	VB Specifications page: 'Multiuse' checkbox.
genOptVBPersistable	VB Specifications page: 'Persistable' checkbox.
genOptVBDataBindingBehavior	VB Specifications page: 'Data binding behavior' checkbox.
genOptVBDataSourceBehavior	VB Specifications page: 'Data source behavior' checkbox.
genOptVBGlobal	VB Specifications page: 'Global namespace' checkbox.

genOptVBCreatable	VB Specifications page: 'Creatable' checkbox.
genOptVBExposed	VB Specifications page: 'Exposed' checkbox.
genOptVBMTS	VB Specifications page: MTS Transaction Mode.
genOptVBNetGenNamespace	VB.Net Specifications page: Generate Namespace.
genOptVBVersion	VB Specifications page: Default Version.
genOptWrapComment	'Source Code Engineering' page: Wrap length for comment lines.

Connector Field Substitution Macros

This table lists each of the connector field substitution macros.

Field substitution macros are named according to Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected. Otherwise the value is empty.

Connector Macros

Macro Name	Description
connectorAlias	Connector 'Properties' dialog: 'Alias' field.
connectorAssociationClassElemGUID	The GUID of the connector's Association Class element.
connectorAssociationClassElemName	The name of the connector's Association Class element.
connectorDestAccess	Connector 'Properties' dialog, 'Target Role' tab: Access.
connectorDes	Connector 'Properties' dialog, 'Target

tAggregation	Role' tab: Aggregation.
connectorDestAlias	Connector 'Properties' dialog, 'Target Role' tab: Alias.
connectorDestAllowDuplicates	Connector 'Properties' dialog, 'Target Role' tab: 'Allow Duplicates' checkbox.
connectorDestChangeable	Connector 'Properties' dialog, 'Target Role' tab: Changeable.
connectorDestConstraint	Connector 'Properties' dialog, 'Target Role' tab: Constraint(s).
connectorDestContainment	Connector 'Properties' dialog, 'Target Role' tab: Containment.
connectorDestDerived	Connector 'Properties' dialog, 'Target Role' tab: 'Derived' checkbox.
connectorDestDerivedUnion	Connector 'Properties' dialog, 'Target Role' tab: 'DerivedUnion' checkbox.
connectorDestElem*	A set of macros that access a property of the element at the target end of a connector. The * (asterisk) is a wildcard that corresponds to any Class substitution

	macro in the Class macro list. For example: • connectorDestElemAlias (classAlias) • connectorDestElemAuthor (classAuthor)
connectorDestElemType	The element type of the connector destination element. (Separate from the connectorDestElem* macros because there is no classType substitution macro.)
connectorDestFeature*	A set of macros that access a property of the feature at the target end of a connector. The * (asterisk) is a wildcard that corresponds to any attribute or operation substitution macro in the Attribute macro or Operation macro list, depending on the connectorDestFeatureType. For example: • connectorDestFeatureReturnClassifierGUID - an operation's return classifier GUID • connectorDestFeatureContainment - an attribute's containment
connectorDestFeatureType	The type of the connector destination feature.

	• connectorDestFeatureType="Attribute" or "Operation"
connectorDestMemberType	Connector 'Properties' dialog, 'Target Role' tab: Member Type.
connectorDestMultiplicity	Connector 'Properties' dialog, 'Target Role' tab: Multiplicity.
connectorDestNavigability	Connector 'Properties' dialog, 'Target Role' tab: Navigability.
connectorDestNotes	Connector 'Properties' dialog, 'Target Role' tab: Role Notes.
connectorDestOrdered	Connector 'Properties' dialog, 'Target Role' tab: 'Ordered' checkbox.
connectorDestOwned	Connector 'Properties' dialog, 'Target Role' tab: 'Owned' checkbox.
connectorDestQualifier	Connector 'Properties' dialog, 'Target Role' tab: Qualifier(s).
connectorDestRole	Connector 'Properties' dialog, 'Target Role' tab: Role.
connectorDes	Connector 'Properties' dialog, 'Target

tScope	Role' tab: Target Scope.
connectorDestStereotype	Connector 'Properties' dialog, 'Target Role' tab: Stereotype.
connectorDirection	Connector Properties: Direction.
connectorEffect	'Transition Constraints' dialog: 'Effect' field.
connectorGuard	'Object Flow' and 'Transition Constraints' dialogs: 'Guard' field.
connectorGUID	The unique GUID for the current connector.
connectorIsAssociationClass	True, if the connector is an AssociationClass connector.
connectorName	Connector Properties: Name.
connectorNotes	Connector Properties: Notes.
connectorSourceAccess	Connector 'Properties' dialog, 'Source Role' tab: Access.

connectorSourceAggregation	Connector 'Properties' dialog, 'Source Role' tab: Aggregation.
connectorSourceAlias	Connector 'Properties' dialog, 'Source Role' tab: Alias.
connectorSourceAllowDuplicates	Connector 'Properties' dialog, 'Source Role' tab: Allow Duplicates checkbox.
connectorSourceChangeable	Connector 'Properties' dialog, 'Source Role' tab: Changeable.
connectorSourceConstraint	Connector 'Properties' dialog, 'Source Role' tab: Constraint(s).
connectorSourceContainment	Connector 'Properties' dialog, 'Source Role' tab: Containment.
connectorSourceDerived	Connector 'Properties' dialog, 'Source Role' tab: 'Derived' checkbox.
connectorSourceDerivedUnion	Connector 'Properties' dialog, 'Source Role' tab: 'DerivedUnion' checkbox.

connectorSourceElem*	A set of macros that access a property of the element at the source end of a connector. The * (asterisk) is a wildcard that corresponds to any Class substitution macro in the Class macro list. For example: • connectorSourceElemAlias (classAlias) • connectorSourceElemAuthor (classAuthor)
connectorSourceElemType	The element type of the connector source element. (Separate from the connectorSourceElem* macros because there is no classType substitution macro.)
connectorSourceFeature*	A set of macros that access a property of the feature at the source end of a connector. The * (asterisk) is a wildcard that corresponds to any attribute or operation substitution macro in the Attribute macro or Operation macro list, depending on the connectorSourceFeatureType. For example: • connectorSourceFeatureCode - Operation's Code • connectorSourceFeatureInitial - Attribute's Initial

connectorSourceFeatureType	The type of the connector source feature. • connectorSourceFeatureType="Attribute" or "Operation"
connectorSourceMemberType	Connector 'Properties' dialog, 'Source Role' tab: Member Type.
connectorSourceMultiplicity	Connector 'Properties' dialog, 'Source Role' tab: Multiplicity.
connectorSourceNavigability	Connector 'Properties' dialog, 'Source Role' tab: Navigability.
connectorSourceNotes	Connector 'Properties' dialog, 'Source Role' tab: Role Notes.
connectorSourceOrdered	Connector 'Properties' dialog, 'Source Role' tab: 'Ordered' checkbox.
connectorSourceOwned	Connector 'Properties' dialog, 'Source Role' tab: 'Owned' checkbox.
connectorSourceQualifier	Connector 'Properties' dialog, 'Source Role' tab: Qualifier(s).

connectorSourceRole	Connector 'Properties' dialog, 'Source Role' tab: Role.
connectorSourceScope	Connector 'Properties' dialog, 'Source Role' tab: Target Scope.
connectorSourceStereotype	Connector 'Properties' dialog, 'Source Role' tab: Stereotype.
connectorStereotype	Connector 'Properties' dialog: 'Stereotype' field.
connectorTrigger	'Transition Constraints' dialog: 'Trigger' field.
connectorType	The connector type; f or example, Association or Generalization.
connectorWeight	'Object Flow Constraints' dialog: 'Weight' field.

Constraint Field Substitution Macros

This table lists each of the 'Constraint' field substitution macros.

Field substitution macros are named according to Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected. Otherwise the value is empty.

Constraint Macros

Macro Name	Description
constraintName	'Class' dialog, 'Constraints' tab: Name.
constraintNotes	'Class' dialog, 'Constraints' tab: Notes.
constraintStatus	'Class' dialog, 'Constraints' tab: Status.
constraintType	'Class' dialog, 'Constraints' tab: Type.
constraintWeight	'Class' dialog, 'Constraints' tab: ordering (hand up/down) keys.

Effort Field Substitution Macros

This table lists each of the 'Effort' field substitution macros.

Field substitution macros are named according to Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected. Otherwise the value is empty.

Effort Macros

Macro Name	Description
effortName	Effort window: Effort.
effortNotes	Effort window: Notes (unlabelled).
effortTime	Effort window: Time.
effortType	Effort window: Type.

File Field Substitution Macros

This table lists each of the file field substitution macros.

Field substitution macros are named according to Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected. Otherwise the value is empty.

File Macros

Macro Name	Description
fileExtension	The file type extension of the file being generated.
fileName	The name of the file being generated.
fileNameImpl	The filename of the implementation file for this generation, if applicable.
fileHeaders	'Code Gen' dialog: Headers.
fileImports	'Code Gen' dialog: Imports. For supported languages this also includes dependencies derived from these types of relationship: • Aggregation • Association

	• Attribute classifier • Method return type • Method parameter classifier • Generalization • Realization (to interface) • Template Binding (C++) • Dependency
filePath	The full path of the file being generated.
filePathImpl	The full path of the implementation file for this generation, if applicable.

File Import Field Substitution Macros

This table lists each of the file import field substitution macros.

Field substitution macros are named according to Camel casing. Macros that represent checkboxes return a value of T if the box is selected. Otherwise the value is empty.

File Import Macros

Macro Name	Description
importClassName	The name of the Class being imported.
importFileName	The filename of the Class being imported.
importFilePath	The full path of the Class being imported.
importFromAggregation	T if the Class has an Aggregation connector to a Class in this file, F otherwise.
importFromAssociation	T if the Class has an Association connector to a Class in this file, F

	otherwise.
importFromAtt	T if an attribute of a Class in the current file is of the type of this Class, F otherwise.
importFromDependency	T if the Class has a Dependency connector to a Class in this file, F otherwise.
importFromGeneralization	T if the Class has a Generalization connector to a Class in this file, F otherwise.
importFromMeth	T if a method return type of a Class in the current file is the type of this Class, F otherwise.
importFromParam	T if a method parameter of a Class in the current file is of the type of this Class; otherwise F.
importFromPropertyType	T if the Class has a property (Part/Port) typing to another Class, F otherwise.
importFromRealization	T if the Class has a Realization connector to a Class in this file, F otherwise.
importFromT	T if the Class has a TemplateBinding

emplateBinding	connector to a Class in this file, F otherwise.
importInFile	T if the Class is in the current file, F otherwise.
importPackagePath	The Package path with a '.' separator of the Class being imported.
ImportRelativeFilePath	The relative file path of the Class being imported from the file path of the file being generated.

Link Field Substitution Macros

If you want to provide access to data concerning connectors in the model, particularly Associations and Generalizations, you can use the 'Link field substitution' macros. The macro names are in Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected; otherwise the value is empty.

Link Macros

Macro Name	Description/Result
linkAttAccess	Association 'Properties' dialog, Target Role: 'Access' field.
linkAttAggregation	Association 'Properties' dialog, Source or Target Role: Aggregation.
linkAttCollectionClass	The collection appropriate for the linked attribute in scope.
linkAttContainment	Association 'Properties' dialog, Target Role: Containment.
linkAttName	'Association Properties' dialog: Target.

linkAttNotes	Association 'Properties' dialog, Target Role: Role Notes.
linkAttOwnedByAssociation	True, if the 'Owned' checkbox on the 'Role(s)' page of the Association 'Properties' dialog is not selected.
linkAttOwnedByClass	True, if the 'Owned' checkbox on the 'Role(s)' page of the Association 'Properties' dialog is selected.
linkAttQualName	The Association target qualified by the namespace path (if generating namespaces) and the classifier path (dot delimited).
linkAttRole	Association 'Properties' dialog, Target Role: Role.
linkAttRoleAlias	'Association Properties Target Role' dialog: Alias
linkAttStereotype	Association 'Properties' dialog, Target Role: Stereotype.
linkAttTargetScope	Association 'Properties' dialog, Target Role: Target Scope.
linkCard	Link 'Properties' dialog, Target Role:

	Multiplicity.
linkGUID	The unique GUID for the current connector.
linkIsAssociationClass	True, if the Association is an AssociationClass connector.
linkIsBound	Returns T if any TemplateBindings are specified on the connector.
linkParamSubs	Returns a comma-separated list of the arguments specified.
linkParentName	Generalization 'Properties' dialog: 'Target' field.
linkParentQualName	The Generalization target qualified by the namespace path (if generating namespaces) and the classifier path (dot delimited).
linkStereotype	The stereotype of the current connector.
linkVirtualInheritance	Generalization 'Properties' dialog: 'Virtual Inheritance' field.

Linked File Field Substitution Macros

This table lists each of the 'Linked File' field substitution macros.

Field substitution macros are named according to Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected. Otherwise the value is empty.

Linked File Macros

Macro Name	Description
linkedFileLastWrite	Class 'Properties' dialog: 'Files' tab, 'Last Write' field.
linkedFileNotes	Class 'Properties' dialog: 'Files' tab, 'Notes' field.
linkedFilePath	Class 'Properties' dialog: 'Files' tab, 'File Path' field.
linkedFileSize	Class 'Properties' dialog: 'Files' tab, 'Size' field.
linkedFileType	Class 'Properties' dialog: 'Files' tab, 'Type' field.

Metric Field Substitution Macros

This table lists each of the Metric field substitution macros.

Field substitution macros are named according to Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected. Otherwise the value is empty.

Metric Macros

Macro Name	Description
metricName	Metrics screen: 'Metric' field.
metricNotes	Metrics screen: (Notes) field.
metricType	Metrics screen: 'Type' field.
metricWeight	Metrics screen: 'Weight' field.

Operation Field Substitution Macros

The 'Operation field substitution' macros provide access to data concerning operations in the model. The macro names are in Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected; otherwise the value is empty.

Operation field substitution macros

Macro Name	Description/Result
opAbstract	'Operation' dialog: 'Virtual' checkbox.
opAlias	'Operation' dialog: Alias.
opBehavior	'Operation Behavior' dialog: Behavior.
opCode	'Operation Behavior' dialog: Behavior Code.
opConcurrency	'Operation' dialog: Concurrency.
opConst	'Operation' dialog: 'Const' checkbox.
opGUID	The unique GUID for the current

	operation.
opHasSelfRefParam	Scans the list of parameters in the current Operation, returning 'T' if one type is the Class reference (this could be ClassA* or ClassA&, depending on the value of the genOptCOperationRefParamStyle code generation option field substitution macro).
opImplMacros	A space-separated list of macros defined in the implementation of this operation.
opIsQuery	'Operation' dialog: 'IsQuery' checkbox.
opMacros	A space-separated list of macros defined in the declaration for this operation.
opName	'Operation' dialog: Name.
opNotes	'Operation' dialog: Notes.
opPure	'Operation' dialog: 'Pure' checkbox.
opReturnArray	'Operation' dialog: 'Return Array' checkbox.
opReturnClassifierGUID	The unique GUID for the classifier of the current operation.

opReturnQualType	The operation return type qualified by the namespace path (if generating namespaces) and the classifier path (dot delimited). If the return type classifier has not been set, it is equivalent to the opReturnType macro.
opReturnType	'Operation' dialog: Return Type.
opScope	'Operation' dialog: Scope.
opStatic	'Operation' dialog: 'Static' checkbox.
opStereotype	'Operation' dialog: Stereotype.
opSynchronized	'Operation' dialog: 'Synchronized' checkbox.

Package Field Substitution Macros

This table lists the Package Field Substitution macros.

Field Substitution macros are named according to Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected. Otherwise the value is empty.

Package Macros

Macro Name	Description
packageAbstract	Package dialog: Abstract.
packageAlias	Package dialog: Alias.
packageAuthor	Package dialog: Author.
packageComplexity	Package dialog: Complexity.
packageGUID	The unique GUID for the current Package.
packageKeywords	Package dialog: Keywords.

packageLanguage	Package dialog: Language.
packageName	Package dialog: Name.
packagePath	The string representing the hierarchy of Packages, for the Class in scope. Each Package name is separated by a dot (.).
packagePhase	Package dialog: Phase.
packageScope	Package dialog: Scope.
packageStatus	Package dialog: Status.
packageStereotype	Package dialog: Stereotype.
packageVersion	Package dialog: Version.

Parameter Field Substitution Macros

This table lists each of the Parameter field substitution macros.

Field substitution macros are named according to Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected. Otherwise the value is empty.

Parameter Macros

Macro Name	Description
paramClassifierGUID	The unique GUID for the classifier of the current parameter.
paramDefault	Operation 'Parameters' dialog: 'Default' field.
paramFixed	Operation 'Parameters' dialog: 'Fixed' checkbox.
paramGUID	The unique GUID for the current parameter.
paramIsEnum	True, if the parameter uses the enum keyword (C++).

paramKind	Operation 'Parameters' dialog: 'Kind' field.
paramName	Operation 'Parameters' dialog: 'Name' field.
paramNotes	Operation 'Parameters' dialog: 'Notes' field.
paramQualType	The parameter type qualified by the namespace path (if generating namespaces) and the classifier path (dot delimited). If the parameter classifier has not been set, is equivalent to the paramType macro.
paramType	Operation 'Parameters' dialog: 'Type' field.

Problem Field Substitution Macros

This table lists each of the Problem field substitution macros.

Field substitution macros are named according to Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected. Otherwise the value is empty.

Problem Macros

Macro Name	Description
problemCompletedBy	'Maintenance' dialog, 'Element Issues' tab: Completed by.
problemCompletedDate	'Maintenance' dialog, 'Element Issues' tab: Completed.
problemHistory	'Maintenance' dialog, 'Element Issues' tab: History.
problemName	'Maintenance' dialog, 'Element Issues' tab: Name.
problemNotes	'Maintenance' dialog, 'Element Issues' tab: Description.

problemPriority	'Maintenance' dialog, 'Element Issues' tab: Priority.
problemRaisedBy	'Maintenance' dialog, 'Element Issues' tab: Raised by.
problemRaisedDate	'Maintenance' dialog, 'Element Issues' tab: Raised.
problemStatus	'Maintenance' dialog, 'Element Issues' tab: Status.
problemVersion	'Maintenance' dialog, 'Element Issues' tab: Version.

Requirement Field Substitution Macros

This table lists each of the Requirement field substitution macros with a description of the result.

Field substitution macros are named according to Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected. Otherwise the value is empty.

Requirement Macros

Macro Name	Description
requirementDifficulty	'Properties' dialog: 'Require' tab: Difficulty.
requirementLastUpdated	'Properties' dialog: 'Require' tab: Last Update.
requirementName	'Properties' dialog: 'Require' tab: Short Description.
requirementNotes	'Properties' dialog: 'Require' tab: Notes.
requirementP	'Properties' dialog: 'Require' tab: Priority.

riority	
requirementStatus	'Properties' dialog: 'Require' tab: Status.
requirementType	'Properties' dialog: 'Require' tab: Type.

Resource Field Substitution Macros

This table lists each of the Resource field substitution macros.

Field substitution macros are named according to Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected. Otherwise the value is empty.

Resource Macros

Macro Name	Description
resourceAllocatedTime	Resource Allocation window: Allocated Time.
resourceEndDate	Resource Allocation window: End Date.
resourceExpectedTime	Resource Allocation window: Expected Time.
resourceExpendedTime	Resource Allocation window: Time Expended.
resourceHistory	Resource Allocation window: History.

resourceName	Resource Allocation window: Resource.
resourceNotes	Resource Allocation window: Description.
resourcePercentCompleted	Resource Allocation window: Completed(%).
resourceRole	Resource Allocation window: Role.
resourceStartDate	Resource Allocation window: Start Date.

Risk Field Substitution Macros

This table lists each of the Risk field substitution macros.

Field substitution macros are named according to Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected. Otherwise the value is empty.

Risk Macros

Macro Name	Description
riskName	Risks window: Risk.
riskNotes	Risks window: (Notes).
riskType	Risks window: Type.
riskWeight	Risks window: Weight.

Scenario Field Substitution Macros

This table lists each of the Scenario field substitution macros with a description of the result.

Field substitution macros are named according to Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected. Otherwise the value is empty.

Scenario Macros

Macro Name	Description
scenarioGUID	The unique ID for a scenario. Identifies the scenario unambiguously within a model.
scenarioName	'Properties' dialog, 'Scenario' tab: Scenario.
scenarioNotes	'Properties' dialog, 'Scenario' tab: (Notes).
scenarioType	'Properties' dialog, 'Scenario' tab: Type.

Tagged Value Substitution Macros

Tagged Value macros are a special form of field substitution macros, which provide access to element tags and the corresponding Tagged Values. They can be used in one of two ways:

- Direct Substitution
- Conditional Substitution

Direct Substitution

This form of the macro directly substitutes the value of the named tag into the output.

Structure: %<macroName>:"<tagName>"%

<macroName> can be one of:

- attTag
- classTag
- connectorDestElemTag
- connectorDestTag
- connectorSourceElemTag
- connectorSourceTag
- connectorTag
- linkAttTag
- linkTag
- opTag

- packageTag
- paramTag

This corresponds to the tags for attributes, Classes, operations, Packages, parameters, connectors with both ends, elements at both ends of connectors and connectors including the attribute end.

<tagName> is a string representing the specific tag name.

Example

%opTag:"attribute"%

Conditional Substitution

This form of the macro mimics the conditional substitution defined for field substitution macros.

Structure: %<macroName>:"<tagName>" (== "<test>") ? <subTrue> (: <subFalse>) %

Note:

- <macroName> and <tagName> are as defined here
- (<text>) denotes that <text> is optional
- <test> is a string representing a possible value for the macro
- <subTrue> and <subFalse> can be a combination of

quoted strings and the keyword value; where the value is used, it gets replaced with the macro's value in the output

Examples

%opTag:"opInline" ? "inline" : ""%

%opTag:"opInline" ? "inline"%

%classTag:"unsafe" == "true" ? "unsafe" : ""%

%classTag:"unsafe" == "true" ? "unsafe"%

Tagged Value macros use the same naming convention as field substitution macros.

Template Parameter Substitution Macros

If you want to provide access in a transformation template to data concerning the transformation of a Template Binding connector's binding parameter substitution in the model, you can use the Template Parameter substitution macros. The macro names are in Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected; otherwise the value is empty.

Template Parameter substitution macros

Macro Name	Description
parameterSubstitutionFormal	'Template Binding Properties' dialog, 'Binding Parameter' tab, 'Parameter Substitution(s)' panel: Formal Template Parameter name.
parameterSubstitutionActual	'Template Binding Properties' dialog, 'Binding Parameter' tab, 'Parameter Substitution(s)' panel: Actual parameter name/expression.
parameterSubstitutionActu	'Template Binding Properties' dialog, 'Binding Parameter' tab, 'Parameter

alClassifier	Substitution(s)' panel: Actual parameter classifier.

Test Field Substitution Macros

This table lists each of the Test field substitution macros with a description of the result.

Field substitution macros are named according to Camel casing. Macros that represent checkboxes return a value of 'T' if the box is selected. Otherwise the value is empty.

Test Macros

Macro Name	Description
testAcceptanceCriteria	Testing dialog: Acceptance Criteria.
testCheckedBy	Test Cases window: Checked By.
testDateRun	Test Cases window: Last Run.
testClass	Test Cases window: Test Class (the type of test defined: Unit, Integration, System, Acceptance, Inspection, Scenario)
testInput	Testing dialog: Input.
testName	Test Cases window: Test.

testNotes	Test Cases window: Description.
testResults	Testing dialog: Results.
testRunBy	Test Cases window: Run By. (Values are derived from the Project Author definitions in the 'People' dialog - 'Settings > Reference Data > Model Types > People > Project Authors'.)
testStatus	Test Cases window: Status.
testType	Test Cases window: Type.

Function Macros

Function macros are a convenient way of manipulating and formatting various element data items. Each function macro returns a result string. There are two primary ways to use the results of function macros:

- Direct substitution of the returned string into the output, such as: %TO_LOWER(attName)%
- Storing the returned string as part of a variable definition such as: $name = %TO_LOWER(attName)%

Function macros can take parameters, which can be passed to the macros as:

- String literals, enclosed within double quotation marks
- Direct substitution macros without the enclosing percent signs
- Variable references
- Numeric literals

Multiple parameters are passed using a comma-separated list.

Function macros are named according to the All-Caps style, as in:

%CONVERT_SCOPE(opScope)%

The available function macros are described here. Parameters are denoted by square brackets, as in:

FUNCTION_NAME([param]).

CONVERT_SCOPE([umlScope])

For use with supported languages, to convert [umlScope] to the appropriate scope keyword for the language being generated. This table shows the conversion of [umlScope] with respect to the given language.

Language	Conversions
C++	Package ==> public Public ==> public Private ==> private Protected ==> protected
C#	Package ==> internal Public ==> public Private ==> private Protected ==> protected
Delphi	Package ==> protected Public ==> public Private ==> private Protected ==> protected
Java	Package ==> {blank} Public ==> public Private ==> private

	Protected ==> protected
PHP	Package ==> public Public ==> public Private ==> private Protected ==> protected
VB	Package ==> Protected Public ==> Public Private ==> Private Protected ==> Protected
VB .Net	Package ==> Friend Public ==> Public Private ==> Private Protected ==> Protected

COLLECTION_CLASS([language])

Gives the appropriate collection Class for the language specified for the current linked attribute.

CSTYLE_COMMENT([wrap_length])

Converts the notes for the element currently in scope to plain C-style comments, using /* and */.

DELPHI_PROPERTIES([scope], [separator], [indent])

Generates a Delphi property.

DELPHI_COMMENT([wrap_length])

Converts the notes for the element currently in scope to Delphi comments.

EXEC_ADD_IN(, [function_name],, ...,)

Invokes an Enterprise Architect Add-In function, which can return a result string.

[addin_name] and [function_name] specify the names of the Add-In and function to be invoked.

Parameters to the Add-In function can be specified via parameters [prm_1] to [prm_n].

```
$result = %EXEC_ADD_IN("MyAddin",
```

"ProcessOperation", classGUID, opGUID)%

Any function that is to be called by the EXEC_ADD_IN macro must have two parameters: an EA.Repository object, and a Variant array that contains any additional parameters from the EXEC_ADD_IN call. Return type should be Variant.

```
Public Function ProcessOperation(Repository As EA.Repository, args As Variant) As Variant
```

FIND([src], [subString])

Position of the first instance of [subString] in [src]; -1 if none.

GET_ALIGNMENT()

Returns a string where all of the text on the current line of output is converted into spaces and tabs.

JAVADOC_COMMENT([wrap_length])

Converts the notes for the element currently in scope to javadoc -style comments.

LEFT([src], [count])

The first [count] characters of [src].

LENGTH([src])

Length of [src]. Returns a string.

MATH_ADD(x,y) MATH_MULT(x,y) and MATH_SUB(x,y)

In a code template or DDL template, these three macros perform, respectively, the mathematical functions of:

- Addition (x+y)
- Multiplication (x*y) and
- Subtraction (x-y)

The arguments x and y can be integers or variables, or a combination of the two. Consider these examples, as used in a 'Class' template for C++ code generation:

- $a = %MATH_ADD(3,4)%
- $b = %MATH_SUB(10,3)%
- $c = %MATH_MULT(2,3)%
- $d = %MATH_ADD($a,$b)%

- $e = %MATH_SUB($b,$c)%
- $f = %MATH_MULT($a,$b)%
- $g = %MATH_MULT($a,10)%
- $h = %MATH_MULT(10,$b)%

These compute, in the same sequence, to:

- a = 3 + 4 = $a
- b = 10 - 3 = $b
- c = 2 * 3 = $c
- d = a + b = $d
- e = b - c = $e
- f = a * b = $f
- g = a * 10 = $g
- h = 10 * b = $h

When the code is generated, the .h file (for C++) contains these corresponding strings:

- a = 3 + 4 = 7
- b = 10 - 3 = 7
- c = 2 * 3 = 6
- d = a + b = 14
- e = b - c = 1
- f = a * b = 49
- g = a * 10 = 70
- h = 10 * b = 70

MID([src], [start]) MID([src], [start], [count])

Substring of [src] starting at [start] and including [count] characters. Where [count] is omitted the rest of the string is included.

PI([option], [value], {[option], [value]})

Sets the PI for the current template to [value]. Valid values for [value] are:

- "\n"
- "\t "
- “ “
- “”

<option> controls when the new PI takes effect. Valid values for <option> are:

- I, Immediate: the new PI is generated before the next non-empty template line
- N, Next: the new PI is generated after the next non-empty template line

Multiple pairs of options are allowed in one call. An example of the situation where this would used is where one keyword is always on a new line, as illustrated here:

%PI=" "%

%classAbstract ? "abstract"%

%if classTag:"macro" != ""%
%PI("I", "\n", "N", " ")%
%classTag:"macro"%
%endIf%
class
%className%
For more details, see *The Processing Instruction (PI) Macro*.

PROCESS_END_OBJECT([template_name])

Enables the Classes that are one Class further away from the base Class, to be transformed into objects (such as attributes, operations, Packages, parameters and columns) of the base Class. [template_name] refers to the working template that temporarily stores the data.

REMOVE_DUPLICATES([source], [separator])

Where [source] is a [separator] separated list; this removes any duplicate or empty strings.

REPLACE([string], [old], [new])

Replaces all occurrences of [old] with [new] in the given string <string>.

RESOLVE_OP_NAME()

Resolves clashes in interface names where two method-from interfaces have the same name.

RESOLVE_QUALIFIED_TYPE()
RESOLVE_QUALIFIED_TYPE([separator])
RESOLVE_QUALIFIED_TYPE([separator], [default])

Generates a qualified type for the current attribute, linked attribute, linked parent, operation, or parameter. Enables the specification of a separator other than. and a default value for when some value is required.

RIGHT([src], [count])

The last [count] characters of [src].

TO_LOWER([string])

Converts [string] to lower case.

TO_UPPER([string])

Converts [string] to upper case.

TRIM([string]) TRIM([string], [trimChars])

Removes trailing and leading white spaces from [string]. If [trimChars] is specified, all leading and trailing characters in the set of <trimChars> are removed.

TRIM_LEFT([string]) TRIM_LEFT([string], [trimChars])

Removes the specified leading characters from <string>.

TRIM_RIGHT([string]) TRIM_RIGHT([string],

[trimChars])

Removes the specified trailing characters from <string>.

VB_COMMENT([wrap_length])

Converts the notes for the element currently in scope to Visual Basic style comments.

WRAP_COMMENT([comment], [wrap_length], [indent], [start_string])

Wraps the text [comment] at width [wrap_length] putting [indent] and [start_string] at the beginning of each line.

```
$behavior = %WRAP_COMMENT(opBehavior, "40", "
", "//")%
```

<wrap_length> must still be passed as a string, even though WRAP_COMMENT treats this parameter as an integer.

WRAP_LINES([text], [wrap_length], [start_string] {, [end_string] })

Wraps [text] as designated to be [wrap_length], adding

[start_string] to the beginning of every line and [end_string] to the end of the line if it is specified.

XML_COMMENT([wrap_length])

Converts the notes for the element currently in scope to XML-style comments.

Control Macros

Control macros are used to control the processing and formatting of the templates. The basic types of control macro include:

- The list macro, for generating multiple element features, such as attributes and operations
- The branching macros, which form if-then-else constructs to conditionally execute parts of a template
- The PI macro for formatting new lines in the output, which takes effect from the next non-empty line
- A PI function macro that enables setting PI to a variable and adds the ability to set the PI that is generated before the next line
- The synchronization macros

In general, control macros are named according to Camel casing.

List Macro

If you need to loop or iterate through a set of Objects that are contained within or are under the current object, you can do so using the *%list* macro. This macro performs an iterative pass on all the objects in the scope of the current template, and calls another template to process each one.

The basic structure is:

%list=<TemplateName> @separator=<string> @indent=<string> (<conditions>) %

where <string> is a double-quoted literal string and <TemplateName> can be one of these template names:

- Attribute
- AttributeImpl
- Class
- ClassBase
- ClassImpl
- ClassInitializer
- ClassInterface
- Constraint
- Custom Template (custom templates enable you to define your own templates)
- Effort
- InnerClass
- InnerClassImpl

- LinkedFile
- Metric
- Namespace
- Operation
- OperationImpl
- Parameter
- Problem
- Requirement
- Resource
- Risk
- Scenario
- Test

<conditions> is optional and looks the same as the conditions for 'if' and 'elseIf' statements.

Example

In a Class transform, the Class might contain multiple attributes; this example calls the Attribute transform and outputs the result of processing the transform for each attribute of the Class in scope. The resultant list separates its items with a single new line and indents them two spaces respectively. If the Class in scope had any stereotyped attributes, they would be generated using the appropriately specialized template.

%list="Attribute" @separator="\n" @indent=" "%

The separator attribute, denoted by @separator, specifies the space that should be used between the list items, excluding the last item in the list.

The indent attribute, denoted by @indent, specifies the space by which each line in the generated output should be indented.

Special Cases

There are some special cases to consider when using the %list macro:

- If the Attribute template is used as an argument to the %list macro, this also generates attributes derived from Associations by executing the appropriate LinkedAttribute template
- If the ClassBase template is used as an argument to the %list macro, this also generates Class bases derived from links in the model by executing the appropriate LinkedClassBase template
- If the ClassInterface template is used as an argument to the %list macro, this also generates Class bases derived from links in the model by executing the appropriate LinkedClassInterface template
- If InnerClass or InnerClassImpl is used as an argument to the %list macro, these Classes are generated using the Class and ClassImpl templates respectively; these arguments direct that the templates should be processed

based on the inner Classes of the Class in scope

Branching Macros

Branching macros provide if-then-else constructs. The CTF supports a limited form of branching through these macros:

- if
- elseIf
- else
- endIf
- endTemplate (which exits the current template)

The basic structure of the if and elseIf macros is:

%if <test> <operator> <test>%

where <operator> can be one of:

- ==
- !=
- < (mathematics comparison, less than)
- > (mathematics comparison, greater than)
- <= (mathematics comparison, less than or equal to)
- >= (mathematics comparison, greater than or equal to)

and <test> can be one of:

- a string literal, enclosed within double quotation marks
- a direct substitution macro, without the enclosing percent signs
- a variable reference

Note that if you are using one of the mathematics comparison operators, <test> must be a decimal number in

string format.

Branches can be nested, and multiple conditions can be specified using one of:

- and, or
- or

When specifying multiple conditions, 'and' and 'or' have the same order of precedence, and conditions are processed left to right.

If conditional statements on strings are case sensitive, 'a String' does not equal 'A STRING'. Hence in some situations it is better to set the variable $str=TO_LOWER(variable) or TO_UPPER(variable) and then compare to a specific case.

Macros are not supported in the conditional statements. It is best to assign the results of a macro (string) to a variable, and then use the variable in the comparison.

```
$fldType = % TO_LOWER ($parameter1)%
$COMMENT = "Use the first 4 characters for Date and Time field types"
$fldType4 = % LEFT ($fldType, 4)%
%if $fldType4 == "date"%
Datetime
%endif%
```

This takes a parameter of value “Datetime”, “DATETIME” or “Date”, and returns “Datetime”.

The endif or endTemplate macros must be used to signify the end of a branch. In addition, the endTemplate macro

causes the template to return immediately, if the corresponding branch is being executed.

Example 1

```
%if elemType == "Interface"%
;
%else%
%OperationBody%
%endIf%
```

In this case:

- If the elemType is "Interface" a semi-colon is returned
- If the elemType is not "Interface", a template called Operation Body is called

Example 2

```
$bases="ClassBase"
$interfaces=""%
%if $bases !="" and $interfaces !=""%
: $bases, $interfaces
%elseIf $bases !=""%
: $bases
%elseIf $interfaces !=""%
```

: $interfaces

%endIf%

In this case the text returned is ':ClassBase'.

Conditions using Boolean Value

When setting up branching using conditions that involve a system checkbox (Boolean fields), such as Attribute.Static (attStatic) the conditional statement would be written as:

```
%if attStatic == "T"%
```

For example:

```
% if attCollection == "T" or attOrderedMultiplicity == "T" %
% endTemplate %
```

Synchronization Macros

The synchronization macros are used to provide formatting hints to Enterprise Architect when inserting new sections into the source code, during forward synchronization. The values for synchronization macros must be set in the File templates.

The structure for setting synchronization macros is:

%<name>=<value>%

where <name> can be one of the macros listed here and <value> is a literal string enclosed by double quotes.

Synchronization Macros

Macro Name	Description
synchNewClassNotesSpace	Space to append to a new Class note. Default value: \n.
synchNewAttributeNotesSpace	Space to append to a new attribute note. Default value: \n.
synchNewOperationNotesSpace	Space to append to a new operation note. Default value: \n.

synchNewOperationBodySpace	Space to append to a new operation body. Default value: \n.
synchNamespaceBodyIndent	Indent applied to Classes within non-global namespaces. Default value: \t.

The Processing Instruction (PI) Macro

The PI (Processing Instruction) macro provides a means of defining the separator text to be inserted between the code pieces (which represent entities) that are generated using a template.

The structure for setting the Processing Instruction is:

%PI=<value>%

In this structure, <value> is a literal string enclosed by double quotes, with these options:

- "\n" - New line (the default)
- " " - Space
- "\t" - Tab
- "" - Null

By default, the PI is set to generate a new line (\n) for each non-empty substitution, which behavior can be changed by resetting the PI macro. For instance, a Class's Attribute declaration in simple VB code would be generated to a single line statement (with no new lines). These properties are derived from the Class-Attribute properties in the model to generate, for example:

Private Const PrintFormat As String = "Portrait"

The template for generating this starts with the PI being set to a space rather than a new line:

% PI = " " %

% CONVERT_SCOPE (attScope)%

% endIf %

% if attConst == "T" %

Const

% endIf %

On transforming this, attscope returns the VB keyword 'Private' and attConst returns 'Const' on the same line spaced by a single space (fitting the earlier VB Class.Attribute definition example).

Alternatively, when generating a Class you might want the Class declaration, the notes and Class body all separated by double lines. In this case the %PI is set to '/n/n' to return double line spacing:

% PI = "\n\n" %

% ClassDeclaration %

% ClassNotes %

% ClassBody %

PI Characteristics

- Blank lines have no effect on the output
- Any line that has a macro that produces an empty result does not result in a PI separator (space/new line)
- The last entry does not return a PI; for example, %Classbody% does not have a double line added after the body

Code Generation Macros for Executable StateMachines

The templates listed here are available through the Code Template Editor (the 'Develop > Source Code > Options > Edit Code Templates' ribbon option); select 'STM_C++_Structured' in the 'Language' field.

The templates are structured as shown:

StmContextStateMachineEnum
    StmStateMachineEnum

StmContextStateEnum
    StmAllStateEnum

StmContextTransitionEnum
    StmTransitionEnum

StmContextEntryEnum
    StmAllEntryEnum

StmContextStateMachineStringToEnum
    StmStateMachineStringToEnum

StmContextStateEnumToString
    StmStateEnumToString

StmContextTransitionEnumToString
    StmTransitionEnumToString

StmContextStateNameToGuid
    StmStateNameToGuid

StmContextTransitionNameToGuid
    StmTransitionNameToGuid

StmContextDefinition
    StmStateMachineEnum
    StmAllStateEnum
    StmTransitionEnum
    StmAllEntryEnum
    StmAllRegionVariableInitialize
    StmStateWithDeferredEvent
        StmDeferredEvent
    StmTransitionProcMapping
    StmTransitionProc
        StmTransitionExit
        StmTransitionEntry
        StmTargetOutgoingTransition

StmTargetParentSubmachineState

StmStateProcMapping

StmStateProc

StmStateEntry

StmOutgoingTransition

StmConnectionPointReferenceEntry

StmParameterizedInitial

StmSubMachineInitial

StmRegionInitial

StmRegionDeactive

StmStateExitProc

StmStateTransition

StmStateEvent

StmStateTriggeredTransition

StmStateCompletionTransition

StmStateIncomingTransition

StmStateOutgoingTransition

StmSubmachineStateExitEvent

StmVertexOutgoingTransition

StmConnectionPointReferenceExitEvent

StmStateExitEvent

StmVertexOutgoingTransition

StmAllRegionVariable

StmStateMachineStringToEnum

StmStateMachineRun

StmStateInitialData
StmStateMachineEntry
StmOutgoingTransition
StmStateMachineRunInitial
StmStateMachineInitial
StmStateMachineRuns

StmContextManager

StmSimulationManager
StmContextInstanceDeclaration
StmContextInstance
StmContextVariableRunstate
StmContextInstanceAssociation
StmContextInstanceClear

StmEventProxy
StmSignalEnum
StmContextJoinEventEnum
StmJoinEventEnum
StmEventEnum
StmSignalDefinition
StmSignalAttributeAssignment
StmSignalAttribute
StmSignalInitialize

StmEventStringToEnum
StmEventEnumToString
StmEventNameToGuid

StmConsoleManager
StmContextInstanceDeclaration
StmContextInstance
StmContextVariableRunstate
StmContextInstanceAssociation
StmContextInstanceClear

StmStateMachineStrongToEnum

StmInitialForTransition

StmVertextOutgoingTransition

StmSendEvent

StmBroadcastEvent

StmContext$_{Ref}$

Signal & Event

Macro name	Description
stmEventEnum	The name of the Event with the prefix 'ENUM_', all upper case.
StmEventGuid	The GUID of the Event.
stmEventName	The name of the Event with spaces and asterisks removed.
stmEventVariable	The name of the Event with the prefix 'm_' in lower case.
stmIsSignalEvent	Is 'T' if the element is a SignalEvent.
stmSignalEnum	The name of the Signal with the prefix 'ENUM_', all upper case.
stmSignalFirstEvent	The name of the Event with the prefix 'ENUM_', all upper case.
stmSignalGuid	The GUID of the Signal.
stmSignalNa	The name of the Signal with spaces and

me	asterisks removed.
stmSignalVariable	The name of the Signal with the prefix 'm_' in lower case.
stmTriggerName	Transition Properties: The name of the Trigger.
stmTriggerSpecification	Transition Properties: The specification of the Trigger.
stmTriggerType	Transition Properties: The type of the Trigger.

Context

Macro name	Description
stmContextName	The name of the Class with spaces and asterisks removed.
stmContextQualName	The qualified name of the Class for which code is being generated.
stmContextV	

ariableName	
stmContextFileName	The output file name for the Class for which code is being generated.

Writing Object Runstate to StateMachine Initialization

Macro name	Description
stmContextVariableRunstateName	
stmContextVariableRunstateValue	
stmContextHasStatemachine	Is 'T' if the current context has one or more StateMachines.
stmHasHistoryPattern	Is 'T' if the StateMachine has a History Pattern.

stmHasTerminatePattern	Is 'T' if the StateMachine has a Terminate Pattern.
stmHasDeferredEventPattern	Is 'T' if the StateMachine has a Deferred Event Pattern.
stmHasSubmachinePattern	Is 'T' if the StateMachine has a Submachine Pattern.
stmHasOrthogonalPattern	Is 'T' if the StateMachine has an Orthogonal Pattern.

StateMachine

Macro name	Description
stmStatemachineName	The name of the StateMachine with asterisks and spaces removed.
stmStatemachineEnum	The name of the StateMachine plus 'ENUM_' plus the name of the StateMachine in upper case.
stmStatemachineGuid	The GUID of the StateMachine element.

stmStateCount	The number of State elements in the StateMachine.
stmSubmachineInitialCount	The number of Initial elements in the Sub Machine State element.
stmStatemachineHasSubmachineState	Is 'T' if the StateMachine has at least one SubMachine State.
stmStatemachineInitialCount	The number of Initial elements in the StateMachine.

Region

Macro name	Description
stmRegionEnum	The name of the State Region plus 'ENUM_' plus the name of the State Region in upper case.
stmRegionFQName	The fully qualified name of the State Region.

stmRegionName	The name of the State Region with spaces and asterisks removed.
stmRegionVariable	The name of the State Region with the prefix 'm_' in lower case.
stmRegionFQVariable	The fully qualified name of the State Region with the prefix 'm_' in lower case.
stmRegionGuid	The GUID of the Region.
stmRegionInitial	
stmRegionIsOwnedByStateMachine	Is 'T' if the Region is owned by a StateMachine.

Transition

Macro name	Description
stmTransitionEnum	The name of the Transition with the prefix 'ENUM_', plus the name of the

	Transition in upper case.
stmTransitionGuid	The GUID of the Transition.
stmTransitionName	The name of the Transition with spaces and asterisks removed.
stmTransitionSourceGuid	The GUID of the Source element in the Transition.
stmTransitionTargetGuid	The GUID of the Target element in the Transition.
stmTransitionVariable	The name of the Transition with the prefix 'm_' in lower case.
stmTransitionSourceVariable	
stmTransitionTargetVariable	
stmTransitionFQVariable	
stmSourceVe	The name of the Transition's source

rtexEnum	vertex plus '_ENUM' plus the name of the Transition's source vertex in upper case.
stmTargetVertexEnum	The name of the Transition's target vertex plus '_ENUM' plus the name of the Transition's target vertex in upper case.
stmSourceIsInitial	Is 'T' if the Transition's source is an Initial.
stmSourceIsState	Is 'T' if the Transition's source is a State.
stmSourceIsEntryPoint	Is 'T' if the Transition's source is an Entry Point.
stmSourceIsExitPoint	Is 'T' if the Transition's source is an Exit Point.
stmSourceIsFork	Is 'T' if the Transition's source is a Fork.
stmSourceIsJoin	Is 'T' if the Transition's source is a Join element.
stmTargetIsFinalState	Is 'T' if the Transition's target is a Final State element.
stmTargetIsE	Is 'T' if the Transition's target is an Exit

xitPoint	Point element.
stmTargetIsState	Is 'T' if the Transition's target is a State element.
stmTargetIsChoice	Is 'T' if the Transition's target is a Choice element.
stmTargetIsJunction	Is 'T' if the Transition's target is a Junction element.
stmTargetIsEntryPoint	Is 'T' if the Transition's target is an Entry Point element.
stmTargetIsConnectionPointReference	Is 'T' if the Transition's target is a Connection Point Reference element.
stmTargetIsFork	Is 'T' if the Transition's target is a Fork element.
stmTargetIsJoin	Is 'T' if the Transition's target is a Join element.
stmTransitionEffect	The Effect of the Transition.
stmTransitionGuard	The Guard of the Transition.

stmTransitionKind	The type or kind of the Transition.
stmTargetInitialTransition	
stmTargetIsSubmachineState	Is 'T' if the Transition's target is a Submachine State.
stmSourceStateEnum	The name of the Transition's source state with the prefix '_ENUM' in upper case.
stmTargetStateEnum	The name of the Transition's target state, with the prefix '_ENUM' in upper case.
stmTargetVertexFQName	The fully qualified name of the Transition's target vertex.
stmTargetIsDeepHistory	Is 'T' if the Transition's target is a Deep History State.
stmTargetIsShallowHistory	Is 'T' if the Transition's target is a Shallow History State.
stmTargetIsTerminate	Is 'T' if the Transition's target is a Terminate element.

stmParentIsStateMachine	Is 'T' if the vertex is an Entry Point or Exit Point, or if the container is a StateMachine.
stmSourceParentStateEnum	
stmTargetParentStateEnum	
stmTargetSubmachineEnum	
stmTargetRegionIndex	
stmIsSelfTransition	Is 'T' if the Transition's source is the same as its target.
stmHistoryOwningRegionInitialTransition	
stmDefaultHi	

storyTransition	

Vertex and State

Macro name	Description
stmVertexName	The name of the Vertex.
stmStateName	The name of the State.
stmVertexGuid	The GUID of the Vertex.
stmVertexFQName	The fully qualified name of the Vertex.
stmStateFQName	The fully qualified name of the State.
stmVertexType	The type of the vertex; one of 'State', 'FinalState', 'Pseudostate', 'ConnectionPointReference' or ' ' (empty).

stmPseudostateKind	The kind of the Pseudostate; one of 'initial', 'deepHistory', 'shallowHistory', 'join', 'fork', 'junction', 'choice', 'entryPoint', 'exitPoint' or 'terminate'.
stmPseudostateName	The name of the Pseudostate.
stmPseudostateVariable	The name of the Pseudostate with the prefix 'm_' in lower case.
stmPseudostateStateMachineName	The name of the Pseudostate StateMachine.
stmPseudostateStateMachineVariable	The name of the Pseudostate StateMachine with the prefix 'm_' in lower case.
stmVertexVariable	The name of the Vertex with the prefix 'm_' in lower case.
stmVertexEnum	The name of the Vertex plus '_ENUM' plus the name of the Vertex in upper case.
stmStateEnum	The name of the State plus '_ENUM' plus the name of the State in upper case.

stmConnectionPointReferenceStateName	The name of the Connection Point Reference.
stmConnectionPointReferenceStateVariable	The name of the Connection Point Reference with the prefix 'm_' in lower case.
stmConnectionPointReferenceEntryCount	
stmParameterizedInitialCount	
stmInitialCountForTransition	
stmStateVariable	The name of the State with the prefix 'm_' in lower case.
stmStateEntryBehavior	The behavior defined for an 'entry' Action operation for a State (the text on the 'Behavior' tab for the 'entry' Action

	operation on the Features window for the element).
stmStateEntryCode	The initial code defined for an 'entry' Action operation for a State (the text for the 'entry' Action operation on the Behavior's 'Code' tab).
stmStateDoBehavior	The behavior defined for a 'do' Action operation for a State (the text on the 'Behavior' tab for the 'do' Action operation on the Features window for the element).
stmStateDoCode	The initial code defined for a 'do' Action operation for a State (the text for the 'do' Action operation on the Behavior's 'Code' tab).
stmStateExitBehavior	The behavior defined for an 'exit' Action operation for a State (the text on the 'Behavior' tab for the 'exit' Action operation on the Features window for the element).
stmStateExitCode	The initial code defined for an 'exit' Action operation for a State (the text for the 'exit' Action operation on the Behavior's 'Code' tab).

stmStateSubmachineName	The name of the Submachine.
stmStateSubmachineVariable	The name of the Submachine with the prefix 'm_' in lower case.
stmStateIsFinal	Is 'T' if the State is a FinalState.
stmStateIsSubmachineState	Is 'T' if the State is a Submachine State ('Properties' page | Advanced | 'isSubmachineState' property).
stmSubMachineEnum	The name of the Submachine followed by '_ENUM' plus the name of Submachine in upper case.
stmStateHasChildrenToJoin	
stmStateIsTransitionTarget	
stmThisIsSource	

stmThisIsSourceState	
stmStateParentIsSubmachine	Is 'T' if the State's container is a StateMachine.
stmStateContainerMatchTransitionContainer	
stmVertexRegionIndex	
stmStateRegionCount	The number of regions in the State.
stmStateInitialCount	The number of Initial elements in the StateMachine.
stmVertexContainerVariable	
stmVertexParentEnum	

stmStateHasUnGuardedCompletionTransition	
stmStateEventHasUnGuardedTransition	
stmInitialTransition	

Instance Association

Macro name	Description
stmSourceInstanceName	
stmTargetInstanceName	
stmSourceRoleName	
stmTargetRol	

eName	

EASL Code Generation Macros

Enterprise Architect provides a number of Enterprise Architect Simulation Library (EASL) code generation macros to generate code from behavioral models. These are:

- EASL_INIT
- EASL_GET
- EASLList and
- EASL_END

EASL_INIT

The EASL_INIT macro is used to initialize an EASL behavior model. The behavior model code generation is dependent on this model.

Aspect	Description
Syntax	%EASL_INIT(<<GUID>>)% where: • <<GUID>> is the GUID of the Object (usually a Class element) that is the owner of the behavior model

EASL_GET

The EASL_GET macro is used to retrieve a property or a collection of an EASL object. The EASL objects and the properties and collections for each object are identified in the *EASL Collections* and *EASL Properties* topics.

Aspect	Description
Syntax	$result = %EASL_GET(<<Property>>, <<Owner ID>>, <<Name>>)% where: • <<Property>> is one of "Property", "Collection", "At", "Count", or "IndexOf" • <<OwnerID>> is the ID of the owner object for which the property/collection is to be retrieved • <<Name>> is the name of the property or Collection being accessed • $result is the returned value; this is “” if not a valid property If <<Property>> is: • "At", then <<OwnerID>> is the ID of a collection and <<Name>> is the index into the collection for which the item is to be retrieved • "Count", then <<Owner ID>> is the ID of a collection and <<Name>> is not used; it will retrieve the item number in

	the collection • "IndexOf", then <<Owner ID>> is the ID of a collection and <<Name>> is the ID of the item in the collection; it will retrieve the index (string format) of the item within the collection
Example	$sPropName = %EASL_GET("Property", $context, "Name")%

EASLList

The EASLList macro is used to render each object in an EASL collection using the appropriate template.

Aspect	Description
Syntax	$result = %EASLList=<<TemplateName>> @separator=<<Separator>> @indent=<<indent>> @owner=<<OwnedID>> @collection=<<CollectionName>> @option1=<<OPTION1>> @option2=<<OPTION2>>.....

	@optionN=<<OPTIONN>>% where: • <<TemplateName>> is the name of any behavioral model template or custom template • <<Separator>> is a list separator (such as “\n”) • <<indent>> is any indentation to be applied to the result • <<OwnedID>> is the ID of the object that contains the required collection • <<CollectionName>> is the name of the required collection • <<OPTION1>...<<OPTION99>> are miscellaneous options that might be passed on the template; each option is given as an additional input parameter to the template • $result is the resultant value; this is “” if not a valid collection
Example	$sStates = %EASLList="State" @separator="\n" @indent="\t" @owner=$StateMachineGUID @collection="States" @option=$sOption%

EASL_END

The EASL_END macro is used to release the EASL behavior model.

Aspect	Description
Syntax	%EASL_END%

Behavioral Model Templates

- Action
- Action Assignment
- Action Break
- Action Call
- Action Create
- Action Destroy
- Action If
- Action Loop
- Action Opaque
- Action Parallel
- Action RaiseEvent
- Action RaiseException

- Action Switch
- Behavior
- Behavior Body
- Behavior Declaration
- Behavior Parameter
- Call Argument
- Decision Action
- Decision Condition
- Decision Logic
- Decision Table
- Guard
- Property Declaration
- Property Notes
- Property Object
- State
- State CallBack
- State Enumerate
- State EnumeratedName
- StateMachine
- StateMachine HistoryVar
- Transition
- Transition Effect
- Trigger

EASL Collections

This topic lists the EASL collections for each of the EASL objects, as retrieved by the EASL Code Generation Macros code generation macro.

Action

Collection Name	Description
Arguments	The Action's arguments.
SubActions	The sub-actions of the Action.

Behavior

Collection Name	Description
Actions	The Behavior's Actions.
Nodes	The Behavior's nodes.

Parameters	The Behavior's parameters.
Variables	The Behavior's variables.

Classifier

Collection Name	Description
AllStateMachines	All StateMachines for the Classifier.
AsynchProperties	The asynchronous properties of the Classifier.
AsynchTriggers	The asynchronous triggers of the Classifier.
Behaviors	The behaviors of the Classifier.
Properties	The properties of the Classifier.
TimedProperties	The timed properties of the Classifier.
TimedTrigge	

rs	The timed triggers of the Classifier.
Triggers	All triggers of the Classifier.

Construct

Collection Name	Description
AllChildren	The Construct's children.
ClientDependencies	The client dependencies on the Construct.
StereoTypes	The stereotypes of the Construct.
SupplierDependencies	The supplier dependencies on the Construct.

Node

Collection	Description

Name	
IncomingEdges	The Node's incoming edges.
OutgoingEdges	The Node's outgoing edges.
SubNodes	The sub-nodes of the Node.

State

Collection Name	Description
DoBehaviors	The State's Do behaviors.
EntryBehaviors	The State's Entry behaviors.
ExitBehaviors	The State's Exit behaviors.

StateMachine

Collection Name	Description
AllFinalStates	The StateMachine's final States.
AllStates	All States within the StateMachine, including those within Submachine States.
DerivedTransitions	The StateMachine's derived Transitions with the associated valid effect.
States	The States within the StateMachine.
Transitions	The transitions within the StateMachine.
Vertices	The StateMachine's vertices.

Transition

Collection Name	Description

Effects	The Transition's effects.
Guards	The Transition's guards.
Triggers	The Transition's triggers.

Trigger

Collection Name	Description
TriggeredTransitions	The triggered transitions associated with the Trigger.

Vertex

Collection Name	Description
DerivedOutgoingTransitions	The Vertex's derived outgoing transitions after traversing the pseudo-nodes.

IncomingTransitions	The Vertex's incoming transitions.
OutgoingTransitions	The Vertex's outgoing transitions.

EASL Properties

This topic lists the EASL properties for each of the EASL objects, as retrieved by the EASL Code Generation Macros code generation macro.

Action

Property Name	Description
Behavior	The Action's associated behavior (Call Behavior Action or Call Operation Action).
Body	The Action's body.
Context	The Action's context.
Guard	The Action's guard.
IsFinal	A check on whether the action is a final Action.
IsGuarded	A check on whether the action is a guarded Action.

IsInitial	A check on whether the action is an initial Action.
Kind	The Action's kind.
Next	The Action's next action.
Node	The Action's associated node in the graph.

Argument

Property Name	Description
Parameter	The ID of the Argument's associated parameter.
Value	The default value of the argument.

Behavior

Property	

Name	Description
InitialAction	The Behavior's initial action.
isReadOnly	The isReadOnly of the Behavior.
isSingleExecution	The isSingleExecution of the Behavior.
Kind	The kind of Behavior.
ReturnType	The return type of the Behavior.
Specification	The specification of the Behavior.

CallEvent

Property Name	Description
Operation	The operation of the CallEvent.

ChangeEvent

Property Name	Description
ChangeExpression	The change expression of the ChangeEvent.

Classifier

Property Name	Description
HasBehaviors	A check on whether the Classifier has behavioral models (Activity and Interaction).
Language	The Classifier's language.
StateMachine	The StateMachine of the Classifier.

Condition

Property Name	Description
Expression	The Condition's expression.
Lower	The Condition's lower value.
Upper	The Condition's upper value.

Construct

Property Name	Description
GetTaggedValue	The Property's Tagged Value.
IsStereotypeApplied	A check on whether a particular stereotype is applied to the Property.
Notes	Notes on the Property.
UMLType	The UML type of the Property.
Visibility	The visibility of the Property.

Edge

Property Name	Description
From	The ID of the node from which the Edge arises.
To	The ID of the node at which the Edge is targeted.

EventObject

Property Name	Description
EventKind	The event kind of the Event Object.

Instance

Property Name	Description
Classifier	The classifier of the Instance.
Value	The value of the Instance.

Parameter

Property Name	Description
Direction	The direction of the Parameter.
Type	The type of the Parameter.
Value	The value of the parameter.

Primitive

Property	Description

Name	
FQName	The FQ name of the Primitive.
ID	The ID of the Primitive.
Name	The name of the Primitive.
ObjectType	The object type of the Primitive.
Parent	The IDParent of the Primitive.

PropertyObject

Property Name	Description
BoundSize	The bound size of the PropertyObject (if it is a collection).
ClassifierStereoType	The stereotype of the PropertyObject's classifier.
IsAsynchProp	A check on whether the PropertyObject is an asynchronous property.

IsCollection	A check on whether the PropertyObject is a collection.
IsOrdered	A check on whether the PropertyObject is ordered (if it is a collection).
IsTimedProp	A check on whether the PropertyObject is a timed property.
Kind	The PropertyObject's kind.
LowerValue	The PropertyObject's lower value (if it is a collection).
Type	The PropertyObject's type.
UpperValue	The PropertyObject's upper value (if it is a collection).
Value	The PropertyObject's value.

SignalEvent

Property Name	Description

Signal	The signal of the SignalEvent.

State

Property Name	Description
HasSubMachine	A check on whether the State is a Submachine state.
IsFinalState	A check on whether the State is a final state.
SubMachine	Get the ID of the Submachine contained by the State (if applicable).

StateMachine

Property Name	Description
HasSubMach	A check on whether the StateMachine has

ineState	a Submachine state.
InitialState	The StateMachine's initial state.
SubMachine State	The StateMachine's Submachine State.

TimeEvent

Property Name	Description
When	The 'when' property of the TimeEvent.

Transition

Property Name	Description
HasEffect	A check on whether the transition has a valid effect.
	A check on whether the transition is a

IsDerived	derived transition.
IsTranscend	A check on whether the transition transcends from one StateMachine (Submachine State) to another.
IsTriggered	A check on whether the transition is triggered.
Source	The Transition's source.
Target	The Transition's target.

Trigger

Property Name	Description
AsynchDestinationState	The asynchronous destination state of the Trigger (if it is an asynchronous trigger).
DependentProperty	The ID of the property associated with the Trigger.
Event	The Trigger's event.

Name	The Trigger's name.
Type	The Trigger's type.

Vertex

Property Name	Description
IsHistory	A check on whether the vertex is a history state.
IsPseudoState	A check on whether the vertex is a pseudo state.
PseudoStateKind	The Vertex's pseudostate kind.

Call Templates From Templates

Using function calls with parameters, you can call templates from other templates, whether standard templates or user-defined templates created within your project. Also, called templates can return a value, and can be called recursively.

Examples

A call statement returning a parameter to a variable:

```
$sSource = %StateEnumeratedName($Source)%
```

A call statement to a template that has parameters:

```
%RuleTask($GUID, $index)%
```

Using the $parameter statement in the called template:

```
$GUID = $parameter1
$index = $parameter2
```

Templates support recursive calls, such as this recursive call on the template RuleTask:

```
$GUID = $parameter1
$index = $parameter2
% PI = "" %
$nul = "Initialize condition and action object"
$count = %BR_GET("RuletCount")%
% if $count == "" or $count == $index %
```

```
%ComputeRulet($GUID)%
\n
% endTemplate %
%Rulet($index)%
\n
$index = %MATH_ADD($index, "1")%
%RuleTask($GUID, $index)%
```

The Code Template Editor in MDG Development

These topics describe how you use the Code Template Editor window to create custom templates:

- Create Custom Templates
- Customize Base Templates
- Add New Stereotyped Templates

The Code Template Editor provides the facilities of the Common Code Editor, including Intelli-sense for the code generation template macros. For more information on Intelli-sense and the Common Code Editor, see the *Editing Source Code* topic.

Create Custom Templates

Enterprise Architect provides a wide range of templates that define how code elements are generated. If these are not sufficient for your purposes - for example, if you want to generate code in a language not currently supported by Enterprise Architect - you can create completely new custom templates. You can also add stereotype overrides to your custom templates; for example, you might list all of your parameters and their notes in your method notes.

Access

Ribbon	Develop > Source Code > Options > Edit Code Templates Design > Package > Transform > Transform Templates
Keyboard Shortcuts	Ctrl+Shift+P (code generation templates) Ctrl+Alt+H (MDA transformation templates)

Create custom templates using the Code

Templates Editor

Step	Description
1	In the 'Language' field, click on the drop-down arrow and select the appropriate programming language.
2	Click on the Add New Custom Template button. The 'Create New Custom Template' dialog displays.
3	In the 'Template Type' field, click on the drop-down arrow and select the appropriate modeling object. The '<None>' option requires special treatment; it enables the definition of a function macro that doesn't actually apply to any of the types, but must be called as a function to define variables $parameter1, $parameter2 and so on for each value passed in.
4	In the 'Template Name' field, type an appropriate name. Click on the OK button.
5	On the 'Code Templates Editor' tab, the new template is included in the 'Templates' list, with the value 'Yes' in the 'Modified' field. The template is called <Template

	Type>__<Template Name>. Note the double underscore character between the template type and template name.
6	Select the template from the Templates list and edit the contents in the Template field to meet your requirements.
7	Click on the Save button. This stores the new template, which is now available from the list of templates for use. You can also add a stereotype override to the template, if necessary.

Notes

- For a custom language, you must define the File template so that it can call the Import Section, Namespace and Class templates, and any other templates that you decide are applicable

Customize Base Templates

Enterprise Architect provides a wide range of templates that define how code elements are generated. If you want to change the way a code element is generated, you can customize the appropriate existing system-provided templates. Your changes might be to the effect of the template itself, or to its calls to other templates. You can also add stereotype overrides to your customized templates; for example, you might list all of your parameters and their notes in your method notes.

When you customize a system-provided (base) template, you effectively create a copy of the template that is used in preference to the original. All subsequent changes are to that copy, and the original base template is hidden. If you subsequently delete the copy it can no longer override the original, which is then brought into use again.

Access

Ribbon	Develop > Source Code > Options > Edit Code Templates
Keyboard Shortcuts	Ctrl+Shift+P

Customize a base template

Step	Description
1	On the Code Template Editor, in the 'Language' field, click on the drop-down arrow and select the programming language for which you want to customize the base templates.
2	In the Templates list, click on the base template to edit.
3	Update the template.
4	Click on the Save button to store your changes.
5	Repeat steps 2 to 4 for each of the relevant base templates you want to customize.
6	If you prefer, add one or more stereotype overrides to any of the templates.

Add New Stereotyped Templates

Sometimes it is useful to define a specific code generation template for use with elements of a given stereotype. This enables different code to be generated for elements, depending on their stereotype. Enterprise Architect provides some default templates, which have been specialized for commonly used stereotypes in supported languages. For example, the 'Operation Body' template for C# has been specialized for the property stereotype, so that it automatically generates its constituent 'get' and 'set' methods. You can override the default stereotyped templates as described in the *Override Default Templates* topic. Additionally, you can define templates for your own stereotypes, as described here.

Access

Ribbon	Develop > Source Code > Options > Edit Code Templates
Keyboard Shortcuts	Ctrl+Shift+P

Add a new stereotyped template using the Code Template Editor

Step	Description
1	Select the appropriate language, from the Language list.
2	Select one of the base templates, from the Templates list.
3	Click on the 'Add New Stereotyped Override' button. The 'New Template Override' dialog displays.
4	Select the required Feature and/or Class stereotype. Click on the OK button.
5	The new stereotyped template override displays in Stereotype Overrides list, marked as modified.
6	Make the required modifications in the Code Templates Editor.
7	Click on the Save button to store the new stereotyped template in the project file. Enterprise Architect can now use the stereotyped

	template, when generating code for elements of that stereotype.

Notes

- Class and feature stereotypes can be combined to provide a further level of specialization for features; for example, if properties should be generated differently when the Class has a stereotype MyStereotype, then both property and MyStereotype should be specified in the New Template Override dialog

Override Default Templates

Enterprise Architect has a set of built-in or default code generation templates. The Code Templates Editor enables you to modify these default templates, hence customizing the way in which Enterprise Architect generates code. You can choose to modify any or all of the base templates to achieve your required coding style.

Any templates that you have overridden are stored in the project file. When generating code, Enterprise Architect first checks whether a template has been modified and if so, uses that template. Otherwise the appropriate default template is used.

Access

Ribbon	Develop > Source Code > Options > Edit Code Templates
Keyboard Shortcuts	Ctrl+Shift+P

Reference

Override a default code generation template using the Code Templates Editor.

When generating code, Enterprise Architect now uses the overriding template instead of the default template.

Field/Button	Description
Language	Select the appropriate language from the list.
Templates	Select one of the base templates from the list.
Stereotype Overrides	If the base template has stereotyped overrides, you can select one of these from the list.
<Other fields>	Make any other modifications required.
Save	Click on this button to store the modified version of the template to the project file. The template is marked as modified.

Grammar Framework

Enterprise Architect provides reverse engineering support for a number of popular programming languages. However, if the language you are using is not supported, you can write your own grammar for it, using the in-built Grammar Editor. You can then incorporate the grammar into an MDG Technology to provide both reverse engineering and code synchronization support for your target language.

The framework for writing a grammar and importing it into Enterprise Architect is the direct complement to the Code Template Framework. While code templates are for converting a model to a textual form, grammars are required to convert text to a model. Both are required to synchronize changes into your source files.

An example language source file and an example Grammar for that language are provided in the Code Samples directory, which you can access from your installation directory (the default location is C:\Program Files\Sparx Systems\EA). Two other grammar files are also provided, illustrating specific aspects of developing Grammars.

Components

Component	Description
Grammar	Grammars define how a text is to be

Syntax	broken up into a structure, which is necessary when you are converting code into a UML representation. At the simplest level, a grammar is instructions for breaking up an input to form a structure. Enterprise Architect uses a variation of Backus–Naur Form (nBNF) to include processing instructions, the execution of which returns structured information from the parsed results in the form of an Abstract Syntax Tree (AST), which is used to generate a UML representation.
Grammar Editor	The Grammar Editor is an in-built editor that you can use to open, edit, validate and save grammar files.
Grammar Debugging	You can debug the grammar files you create using two facilities: • The Parser, which generates the AST for the Grammar • The Profiler, which also parses the Grammar and generates the AST but which exposes the Profiling pathway to show exactly what happened at each step of the process

Grammar Syntax

Grammars define how a text is to be broken up into a structure, which is exactly what is needed when you are converting code into a UML representation. At the simplest level, a grammar is just instructions for breaking up an input to form a structure. Enterprise Architect uses a variation of Backus–Naur Form (BNF) to express a grammar in a way that allows it to convert the text to a UML representation. What the grammar from Enterprise Architect offers over a pure BNF is the addition of processing instructions, which allow structured information to be returned from the parsed results in the form of an Abstract Syntax Tree (AST). At the completion of the AST, Enterprise Architect will process it to produce a UML model.

Syntax

Syntax	Detail
Comments	Comments have the same form as in many programming languages. // You can comment to the end of a line by adding two /s. /* You can comment multiple lines by adding a / followed by a *.

	The comment is ended when you add a * followed by a /. */
Instructions	Instructions specify the key details of how the grammar works. They are generally included at the top of the grammar, and resemble function calls in most programming languages.
Rules	Rules make up the body of a grammar. A rule can have one or more definitions separated by pipe delimiters (|). For a rule to pass, any single complete definition must pass. Rules are terminated with the semi-colon character (;).
Definitions	A definition is one of the paths a rule can take. Each definition is made up of one or more terms.
Definition Lists	A definition list corresponds to one or more sets of terms. These will be evaluated in order until one succeeds. If none succeed then the containing rule fails. Each pair of definitions is separated by a | character. This is a simple rule with three definitions: <greeting> ::= "hello" | "hi" | ["good"]

	"morning";
Terms	A term can be a reference to a rule, a specific value, a range of values, a sub-rule or a command.
Commands	Like instructions, commands resemble function calls. They serve two main purposes: • To process tokens in a specific way or • To provide a result to the caller

Grammar Instructions

Instructions specify the key details of how the grammar works. They are generally included at the top of the grammar, and resemble function calls in most programming languages.

Instructions

Instruction	Description
caseSensitive()	One of these two instructions is expected to specify if token matching needs to be case sensitive or not. For example, languages in the BASIC family are case insensitive while languages in the C family are case sensitive.
caseInsensitive()	
delimiters(DelimiterRule: Expression)	The delimiters instruction tells the lexical analyzer which rule to use for delimiter discovery. Delimiters are used during keyword analysis, and can be defined as the characters that can be used immediately before or after language

	keywords.
lex(TokenRule: Expression)	The lex instruction tells the lexical analyzer the name of the root rule to use for its analysis.
parse(RootRule: Expression) parse(RootRule: Expression, SkipRule: Expression)	The parse instruction tells the parser the name of the root rule to use for its processing. The optional second argument specifies a skip (or escape) rule, which is generally used to handle comments.

Grammar Rules

Rules are run to break up text into structure. A rule is made up of one or more definitions, each of which is made up of one or more terms.

Types of Rule

Rule	Description
Named rules	A name, followed by a definition list. For example: <rule> ::= <term1> <term2> | "-" <term1>;
Inline Rules	Inside a definition, a rule defined within parentheses. These act in exactly the same way as if they were a named rule being called by a term. For example: <rule> ::= (<inline>);
Optional Rules	Inside a definition, a rule defined within square brackets. This rule succeeds even if the contents fail. For example: <rule> ::= [<inline>];
Repeating	Inside a definition, a term followed by a

Rules	plus sign. This rule matches the inner rule once or more than once. For example: <rule> ::= <inline>+; rule ::= (<term1> <term2>)+;
Optional Repeating Rules	Inside a definition, a rule followed by a star. This rule matches the inner rule zero or more times, meaning it succeeds even if the inner rule never succeeds. For example: <rule> ::= <inline>*; rule ::= (<term1> <term2>)*;

Grammar Terms

Terms identify where tokens are consumed.

Types of Term

Type	Description
Concrete terms	Quoted strings. For example, "class"
Unicode characters	A lexer-only term, having the prefix of U+0x followed by a hexadecimal number. For example: U+0x1234
Ranges	A lexer-only term, matching any character between the two characters specified. For example, "a".."z" or U+0x1234..U+2345
References	The name of another rule, in angled brackets. The token will match if that rule succeeds. For example, <anotherRule>

Commands	A call to a specific command.

Grammar Commands

Commands, like Instructions, resemble function calls. They serve two main purposes:

- To process tokens in a specific way or
- To provide a result to the caller

Commands

Command	Description
attribute(Name: String, Value: Expression)	Creates an attribute on the current AST node. The attribute will be created with the Name specified in the grammar source, and will be given the value of all tokens consumed as a part of executing the Value expression. This command produces the AST node attributes that Enterprise Architect operates on in code engineering.
attributeEx(Name: String) attributeEx(Name: String, Value: String)	Creates an attribute on the current AST node without consuming any tokens. The attribute will be created with the same name as is specified in the grammar source, and with either an empty value or the value specified by the optional Value

	argument. This command produces the AST node attributes that Enterprise Architect operates on in code engineering.
node(Name: String, Target: Expression)	Creates an AST node under the current AST node (the nodes that Enterprise Architect operates on in code engineering). The node will be created with the Name specified in the grammar source.
token(Target: Expression)	Creates a token during lexical analysis for processing during parsing. The value of the token will be the value of all characters consumed as a result of executing the Target expression.
keywords()	Matches any literal string used as a grammar term; that is, if you enter an explicit string that you are searching for, it becomes a key word.
skip(Target: Expression) skip(Target: Expression, Escape: Expression)	Consumes input data (characters when lexing, and tokens when parsing) until the 'Target' expression is matched. The optional 'Escape' expression can be used to handle instances such as escaped quotes within strings.

skipBalanced (Origin: Expression, Target: Expression) skipBalanced (Origin: Expression, Target: Expression, Escape: Expression)	Consumes input data (characters or tokens) until the 'Target' expression is matched and the nesting level reaches zero. If the 'Origin' expression is matched during this process, the nesting level is increased. If the 'Target' expression is matched, the nesting level is decreased. When the nesting level reaches zero, the command exits with success. An optional 'Escape' expression can be provided.
skipEOF()	Consumes all remaining data (characters or tokens) until the end of the file.
fail()	Causes the parser to fail the current rule, including any remaining definitions.
warning()	Inserts a warning into the resulting AST.
except(Target : Expression, Exception: Expression)	Consumes input data that matches the Target expression, but fail on data that matches the Exception expression. This operates somewhat similar to, but exactly the opposite of, the skip command.
preProcess(T	Evaluates an expression and uses that

arget: Expression)	pre-processed data in multiple definitions. This is most useful within expression parsing, where the same left hand side expression will be evaluated against a number of operators. This command reduces the work the parser must do to make this happen.

AST Nodes

In defining a grammar, you would use AST nodes and AST node attributes that can be recognized in code engineering in Enterprise Architect, in the AST results that are returned by the attribute, attributeEx and node commands. The nodes and attributes are identified in these tables. Any others will be ignored in code engineering.

FILE Node

The FILE node represents a file. It isn't mapped to anything, but contains all the required information.

Multiplicity / Nodes	Description
0..* / PACKAGE	See *PACKAGE Node*.
0..* / CLASS	See *CLASS Node*.
0..* / IMPORT	The node to represent the imported namespace/Package or equivalent. The 'NAME' attribute of the node will be the name of imported namespace/Package or equivalent.

0..* / COMMENT	Field labels as part of a skip rule will be at the root level; the code generator looks for comments of this sort by position relative to the node.
0..1 / INSERT_POSITION	This gives the position where new Classes, Packages and method implementations can be inserted into the file. If it is not found, the code generator will automatically insert new items immediately after the last one is found in code.

PACKAGE node

The PACKAGE node corresponds to a namespace or equivalent in the file. When importing with 'package per namespace', Enterprise Architect will create a Package directly under the import for this and place all Classes within it. When not importing namespaces, Enterprise Architect will look for Classes under this point, but it will do nothing with this node.

Additionally, if you are generating with namespaces enabled (see the *Code Options* topics for generic languages) a generated Class will not match a Class in code unless they are under the same Package structure.

Contained in nodes: FILE

Multiplicity / Nodes	Description
1 / NAME	See *NAME Node*.
0..* / CLASS	See *CLASS Node*.
0..* / PACKAGE	The child Package node.
0..1 / OPEN_POSITION	Gives the position where the Package body opens. This can also be used as an insert position.
0..1 / INSERT_POSITION	Gives the position where new Classes and Packages can be inserted into the file. If it is not found, the code generator will automatically insert new items immediately after the last one is found in code.
0..1 / SUPPRESS	Prevents indenting when inserting into this Package.

CLASS/INTERFACE Node

The CLASS (or INTERFACE) node is the most important in code generation. It is brought in as Class (or Interface) Objects.

See *Class DECLARATION* and *Class BODY*.

Contained in Nodes: FILE, PACKAGE, Class BODY

CLASS Declaration

Contained in Nodes: CLASS/INTERFACE

Multiplicity / Nodes	Description
1 / NAME	See *NAME Node*.
0..* / PARENT	See *PARENT Node*.
0..* / TAG	See *TAG Node*.
0..1 / DESCRIPTION	See *DESCRIPTION Node*.
1 / NAME	The name of the Class. If there is a node

	NAME, that will overwrite this attribute.
0..1 / SCOPE	The UML Scope of the Class - Public, Private, Protected or Package.
0..1 / ABSTRACT	If present, indicates that this is an abstract Class.
0..1 / VERSION	The version of the Class.
0..1 / STEREOTYPE	The stereotype that Enterprise Architect should assign to the Class. This does not support multiple stereotypes.
0..1 / ISLEAF	If present, indicates that this is a leaf/final/sealed Class which cannot be inherited by any sub-Class.
0..1 / MULTIPLICITY	If present, represents the multiplicity of the Class.
0..1 / LANGUAGE	Generally, you do not need to set this.
0..1 / NOTE	Generally not used as it is addressed by the comments above the Class.

0..1 / ALIAS	If present, represents the Alias of any identifier, such as a Namespace, Class or variable.
0..* / MACRO	Adds a numbered Tagged Value that Enterprise Architect can use to round trip macros.

Class BODY Node

Contained in Nodes: CLASS/INTERFACE

Multiplicity / Nodes	Description
0..* / METHOD	See *METHOD Node*.
0..* / ATTRIBUTE	See *ATTRIBUTE Node*.
0..* / FIELD	See *FIELD Node*.
0..* / CLASS	See *CLASS Node*.

0..* / SCOPE	See *SCOPE Node*.
0..* / PROPERTY	This node represents the Property definition within the Class Body.
0..* / TAG	See *TAG Node*.
0..* / PARENT	See *PARENT Node*.
0..1 / OPEN_POSITION	Gives the position where the Class body opens. This can also be used as an insert position.
0..1 / INSERT_POSITION	Gives the position where new Class members can be inserted into the file. If it is not found, the code generator will automatically insert new items immediately after the last one is found in code.

SCOPE Node

This is an optional feature for languages resembling C++ that have Blocks that specify the scope of elements. The

language needs to have a name specified that is used for the scope of all elements in the Block. In all other respects it behaves identically to the Class BODY node.

Contained in Nodes: Class BODY

Multiplicity / Nodes	Description
1 / NAME	Used as the scope for all methods and attributes contained within the scope.

METHOD Node

Contained in Nodes: Class BODY, SCOPE

Multiplicity / Nodes	Description
1 / Method DECLARATION	See *Method DECLARATION Node*.

Method DECLARATION Node

Contained in Nodes: METHOD

Multiplicity / Nodes	Description
0..1 / TYPE	See *TYPE Node*.
0..* / PARAMETER	See *PARAMETER Node*.
0..* / TAG	See *TAG NODE*.
0..1 / DESCRIPTION	See *DESCRIPTION Node*.
0..1 / MULTI PARAMETER	Supports Delphi's parameter list style of declaration. This is the equivalent of FIELD.
1 / NAME	The name of the method.
0..1 / TYPE	The return type of the method.
0..1 / SCOPE	The UML Scope of the method - Public, Private, Protected or Package.
0..1 /	If present, indicates that the method is

ABSTRACT	Abstract.
0..1 / STEREOTYPE	The stereotype that Enterprise Architect should assign to the Method. This does not support multiple stereotypes.
0..1 / STATIC	If present, indicates that the method is static.
0..1 / CONST or CONSTANT	If present, indicates that the method is constant.
0..1 / PURE	If present, indicates that the method is a Pure method.
0..1 / ISQUERY	If present, indicates that the method is query/read only.
0..1 / ARRAY	If present, indicates that the method type (return type) is an array.
0..1 / SYNCHRONIZED	If present, indicates that the method is a synchronized method.
0..* / MACRO	The Macro specified in the method declaration.

0..1 / CSHARPIMPLEMENTS	Specifies special behavior for C#.
0..1 / BEHAVIOR	Provides support for Aspect J, using behavior.
0..1 / SHOWBEHAVIOR	Provides support for Aspect J, using behavior, and shows the reverse-engineered behavior on the diagram.

ATTRIBUTE Node

Contained in Nodes: Class BODY, SCOPE

Multiplicity / Nodes	Description
1 / TYPE	See *TYPE Node*.
0..* / TAG	See *TAG Node*.
0..1 / DESCRIPTION	See *DESCRIPTION Node*.

1 / NAME	The name of the Attribute.
0..1 / TYPE	The type of the Attribute.
0..1 / SCOPE	The UML Scope of the Attribute - Public, Private, Protected or Package.
0..1 / DEFAULT	The default value of the Attribute.
0..1 / CONTAINER or ARRAY	If present, indicates the container for the Attribute.
0..1 / CONTAINMENT	Reference or value.
0..1 / STEREOTYPE	The stereotype that Enterprise Architect should assign to the Attribute. This does not support multiple stereotypes.
0..1 / STATIC	If present, indicates that it is a static Attribute.
0..1 / CONST or CONSTANT	If present, indicates that it is a constant Attribute.

0..1 / ORDERED	If present, indicates that the Attribute (value) is ordered.
0..1 / LOWBOUND	If present, represents the lower boundary of the Attribute value.
0..1 / HIGHBOUND	If present, represents the higher boundary of the Attribute value.
0..1 / TRANSIENT or VOLATILE	If present, indicates that the Attribute is Transient or Volatile.

FIELD Node

A field corresponds to multiple attribute declarations in one. Anything not defined in the Declarators but defined in the field itself will be set for each declarator. Everything supported in an attribute is supported in the field. If no declarators are found then this works in the same way as an attribute.

Contained in Nodes: Class BODY, SCOPE

Multiplicity /	Description

Nodes	
0..* / DECLARATOR	See *ATTRIBUTE Node*.

PARAMETER Node

Contained in Nodes: Method DECLARATION, TEMPLATE

Multiplicity / Nodes	Description
1 / TYPE	See *TYPE Node*.
0..* / TAG	See *TAG Node*.
0..1 / DESCRIPTION	See *DESCRIPTION Node*.
0..1 / NAME	The name of the parameter.
0..1 / TYPE	The type of the parameter.

0..1 / KIND	Expected to be in, inout, out or return.
0..1 / DEFAULT	The default value of the parameter.
0..1 / FIXED	If present, indicates that the parameter is fixed/constant.
0..1 / ARRAY	If present, indicates that the parameter type is an array.

NAME Node

Contained in Nodes: PACKAGE, Class DECLARATION

Multiplicity / Nodes	Description
1 / NAME	The name portion.
0..* / QUALIFIER	The qualifier portion.
0..* / NAMEPART	An alternative to using NAME and QUALIFIER. A string of values, all except the last one taken as qualifiers. The last one is taken as the Name.

TYPE Node

Contained in Nodes: Method DECLARATION, ATTRIBUTE, PARAMETER

Multiplicity / Nodes	Description
0..1 / TEMPLATE	The entire text of the template is the name of the type. Only used if NAME is undefined. See *TEMPLATE Node*.
1 / NAME	The name portion.
0..* / QUALIFIER	The qualifier portion.
0..* / NAMEPART	An alternative to using NAME and QUALIFIER. A string of values, all except the last one taken as qualifiers. The last one is taken as the Name.

TEMPLATE Node

Contained in Nodes: TYPE

Multiplicity / Nodes	Description
0..* / PARAMETER	See *PARAMETER Node*.
1 / NAME	

PARENT Node

Contained in Nodes: Class DECLARATION

Multiplicity / Nodes	Description
0..1 / TYPE	Has the value Parent, Implements or VirtualP.
1 / NAME	The name portion of the Parent.
0..* / QUALIFIER	The qualifier portion of the Parent.

0..* / NAMEPART	An alternative to using NAME and QUALIFIER. A string of values, all except the last one taken as qualifiers. The last one is taken as the Name.
0..1 / INSTANTIATION	If present, indicates the instantiation of a template parameter.

TAG Node

Contained in Nodes: Class DECLARATION, Method DECLARATION, ATTRIBUTE, PARAMETER

Multiplicity / Nodes	Description
1 / NAME	The name of the Tagged Value (the Tag).
0..* / VALUE	The value of the Tagged Value.
0..1 / MEMO	If present, indicates that the type of the Tagged Value is <memo>.
0..1 /	If present, indicates that the type of the

NOMEMO	Tagged Value is not <memo>.
0..1 / GROUP	If present, indicates that the value is a Tagged Value group.

DESCRIPTION Node

Contained in Nodes: Class DECLARATION, Method DECLARATION, ATTRIBUTE, PARAMETER

Multiplicity / Nodes	Description
0..* / VALUE	The text that Enterprise Architect should assign to the Note.

Editing Grammars

If you need to write and edit a grammar for code imported in a new programming language, you can do so using the built-in Grammar Editor.

Access

Ribbon	Develop > Source Code > Grammar Editor

Create and Edit Grammar

Field/Button	Action
Open Grammar	Display a browser through which you can locate and open the file containing the grammar you want to edit.
Recent	Recently used grammars can be quickly accessed using this combo box.
Save	Save the current file.

Save As	Saves a copy of the current file
Validate Grammar	The grammar validation will run a series of tests on the current grammar to ensure its validity. Errors and warnings will be displayed informing you of both errors that will make the grammar unusable, and conditions where you might get unexpected results.
Help	Display this Help topic.

Context Menu Options

Field/Button	Action
Open File	Display a browser through which you can locate and open the file containing the grammar you want to edit.
Validate	The grammar validation will run a series of tests on the current grammar to ensure its validity. Errors and warnings will be displayed informing you of both errors that will make the grammar unusable, and

	conditions where you might get unexpected results.
Language	The Grammar Editor defaults to normal Backus–Naur Form (nBNF). The mBNF option is also available.
Line Numbers	Turn line numbers on or off in the grammar editor.

Parsing AST Results

The Abstract Syntax Tree (AST) is the code that Enterprise Architect sees as it processes a grammar.

You parse the text in the bottom half of the Grammar Editor window and review what is displayed as a result. You can either open a file or paste text in. If you have pasted text that corresponds to something that cannot appear at the file level (such as Operation Parameters) you can select an alternative rule to use as a starting point. The parse will then commence from that rule.

Access

Ribbon	Develop > Source Code > Grammar Editor > Grammar Debugger > AST Results

Toolbar Options

Option	Action
Open File	Open a sample input file to test against.

Recent	Recently opened source files can be selected from this combo box.
Parse	Perform the parse operation. If the parse is successful, the 'AST Results' tab will contain the resulting AST.
Select Rule	This drop down allows you to select an alternative root rule for processing your sample source.
Help	Display this Help topic.

Profiling Grammar Parsing

When you parse a grammar that you have created, it might show errors that you cannot immediately diagnose. To help you resolve such errors, you can review the process that the parser followed to generate the AST you can see, using the Grammar Profiler.

You again parse the text in the bottom half of the Grammar Editor window, but this time the tree shows each rule that the parser attempted, where it got to and if it passed or not. Rules for opening a file, pasting a file and setting the starting rule remain the same.

Access

Ribbon	Develop > Source Code > Grammar Editor > Grammar Debugger > Profiler Results

Toolbar Options

Option	Action

Open File	Display a browser through which you can locate and open the file containing the grammar you want to edit.
Parse	Perform the parse operation. If the parse is successful, the 'AST Results' tab will contain the resulting AST, and the 'Profile Results' tab will contain debug information regarding the path that the parser took through your grammar. The profile data is extremely useful when debugging a new grammar.
Select Rule	If you want to use a different root rule for processing your sample source, click on the drop-down arrow and select the alternative rule.
Help	Display this Help topic.

Notes

- Because profiling can take a very long time for large files, the 'Profile Results' tab is not filled if you are not displaying that tab when you begin parsing

Macro Editor

The macro editor allows a user to supplement the grammar with a list of keywords and rules to exclude macros during grammar parse operations. The macro definition list is particularly useful when developing grammars for languages that support macros such as C++. It avoids the necessity of describing these rules in the grammar itself, and can be used with multiple grammars.

This feature is available from Enterprise Architect Release 14.1.

Access

Ribbon	Develop > Source Code > Grammar Editor > Macro Editor

Editing Macros

Open File	Open an existing macro definition list

Recent	Recently opened macro definition lists can be selected from this combo box
Save	Saves changes to the opened macro definition list
Save As	Saves a copy of the existing macro definition list
Validate	Validates the grammar of the macro definition list

Example Grammars

The Code Samples directory set up by the Enterprise Architect installer contains an example Grammar that you can load into the Grammar editor to review, and into the Grammar Debugger to parse and profile.

The Grammar example consists of two files:

- test.ssl - a simple sample language source file, in the style of C, and
- ssl.nbnf - a grammar for the simple sample language

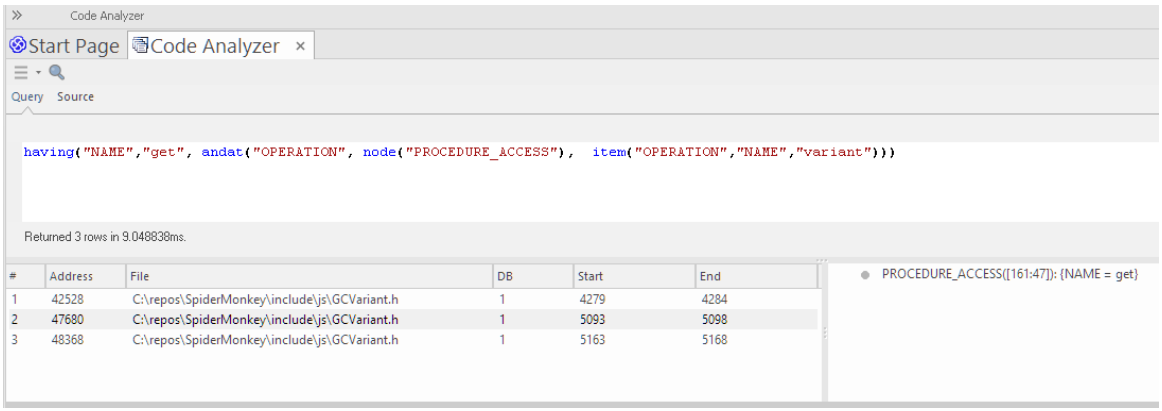
The example illustrates:

- Tokenization (using the Lexer)
- Creation of a Package
- Creation of a Class or Interface
- Creation of an attribute
- Creation of an operation (with parameters)
- Importing comments

The Code Samples directory also contains two other Grammar files that you can examine:

- Expressions Sample.nBNF - this illustrates how expression parsing is set up and processed, with detailed comment text providing explanations
- CSV Sample.nBNF - an example grammar for processing CSV files

Code Analyzer



The Code Analyzer is an essential tool for anyone who deals with source code every day.

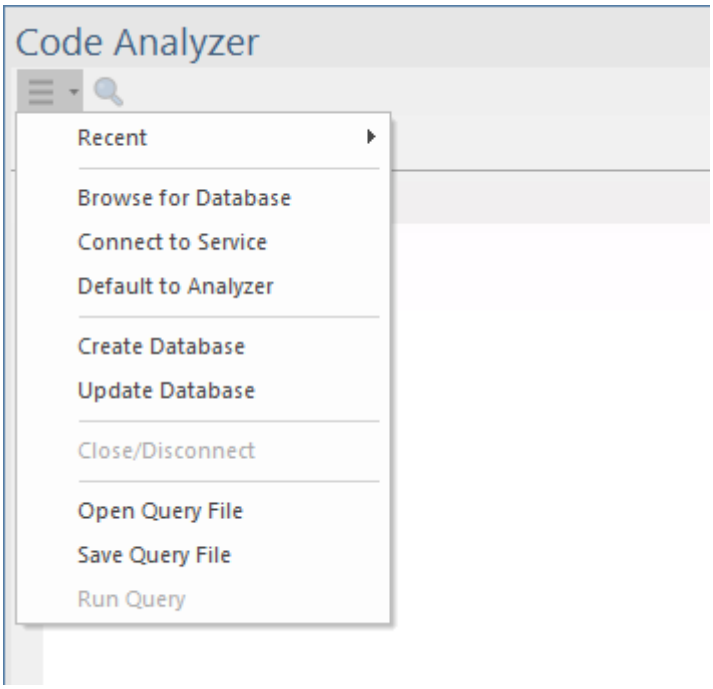
It can perform very complex queries on source code repositories at lightning speed either locally or on a Sparx Intel cloud service. The queries are composed using a high level language developed by Sparx System. The language uses a small but expressive vocabulary that is easily learned and permits code metrics to be queried much faster than conventional methods.

Access

Ribbon	Develop > Source Code > Code Analyzer

Code Analyzer Menu

The Code Analyzer menu is displayed when you click on the icon in the top-left corner of the window.



The menu provides various commands for activities associated with the use of the Code Analyzer, including such things as choosing a Code Miner database to use, updating the Code Miner database and Opening a Query File for editing.

This table describes each of the menu commands.

Command	Description
Recent	Displays a sub-menu that provides a list of recent connections to services and local database files.
Browse for Database	Displays a 'file chooser' dialog, allowing you to browse for a Code Miner database

	on your machine.
Connect to Service	Displays the 'Code Miner Database Connection' dialog, in which you specify connection details for a (list of) Code Miner Database services.
Default to Analyzer	Selecting this option results in the Code Analyzer automatically connecting to the Code Miner service configured for the active Execution Analyzer Script, when the Code Analyzer is started.
Create Database	Displays the 'Create Code Miner Database' dialog, which allows you to create a Code Miner database from a source code repository in the file system.
Update Database	Displays the 'Code Miner Database Update' dialog, which allows you to perform an incremental update to an existing Code Miner database, to incorporate recent changes to source code files.
Close/Disconnect	Closes or disconnects from the Code Miner Database library or service.
Open Query	Shows a 'file open' dialog allowing you to

File	choose an mFQL query file from the file system.
Save Query File	Shows a 'file save' dialog allowing you to save the current mFQL query to a named file.
Run Query	Runs the entire query or selected contents of the query entered in the 'Query' tab editor. Shortcut F6.

Before Using the Analyzer

Before you can use the Code Analyzer, you must first create a Code Miner database or locate an existing one that the Code Analyzer can access. Creating a Code Miner database is summarized here, or you can read a detailed description in the Help topic *Creating a New Code Miner Database*.

Depending on the location of the library you will be using, you should either:

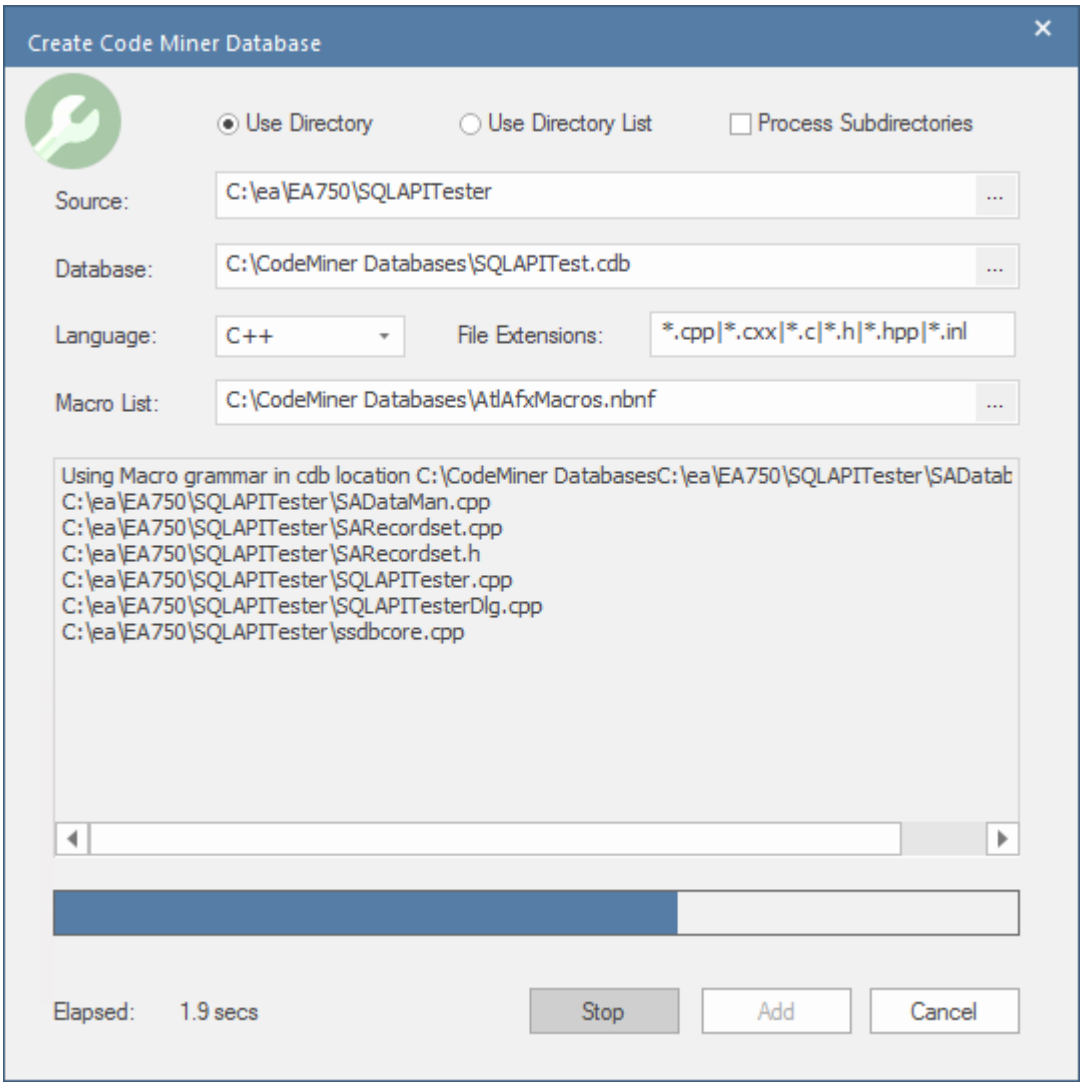
- Select a Code Miner library file to use, or
- Connect to a service that is hosting a Code Miner database.

Once you have completed these steps, you are ready to begin writing and running queries in the Code Analyzer.

Creating a Code Miner Database

Code Miner databases are built from source code repositories. The process is similar to code compilation, using the language grammar to analyze individual files.

There are two types of build - full and incremental. The initial full build might take some time, but the subsequent incremental builds are incredibly quick.



Using a Directory as input

You can select a single folder as the root of the source code you want to compile. With this option you can choose to include subdirectories

Using a Directory List

Sometimes, you want to use more than a single project, but not all the projects are under a single directory. In this case, you can create a text file that lists the full path to each folder you want to include and you specify that text file in the 'Source' field. Each directory path should be listed on a separate line.

c:\myprojects\project1\tools\scintilla

c:\myprojects\project2\src

d:\mylibs\lib1\src

If you want to recursively process the sub-directories within a directory, precede the path with an exclamation mark like this:

!d:\mylibs\lib1\src

Any line that begins with a # character is treated as a comment.

include scintilla

c:\myprojects\project1\tools\scintilla

Language

In this field, you specify the language used in the source code from which this Code Miner database is being built.

Available languages are: C++, C#, Java, XML, MDGTechnology and Custom.

Macro List

When the language selected is 'C++', the 'Macro List' selection field is displayed . For C++, the success and depth of information compiled into the database can be inextricably linked to the use of macros. This field can be used to select an nBNF macro file that will be used as an auxiliary grammar component for the compilation.

By default the macro file will default to the macro file in the Enterprise Architect installation folder. You are free to modify or extend the content of this file to suit your requirements - for example, when you need to correct errors reported in the compilation log file.

Grammar

Sparx Systems has developed grammars for all of the languages listed in the drop-down selection list; C++, C#, Java, XML and also MDGTechnology. For these languages a built-in grammar file is used.

There is also an option to select a 'Custom' language. When 'Custom' is selected, the 'Grammar' field is displayed. This field is used to specify a file containing the grammar for your custom language. The Code Miner will then use that

grammar to parse the source code written in that language.

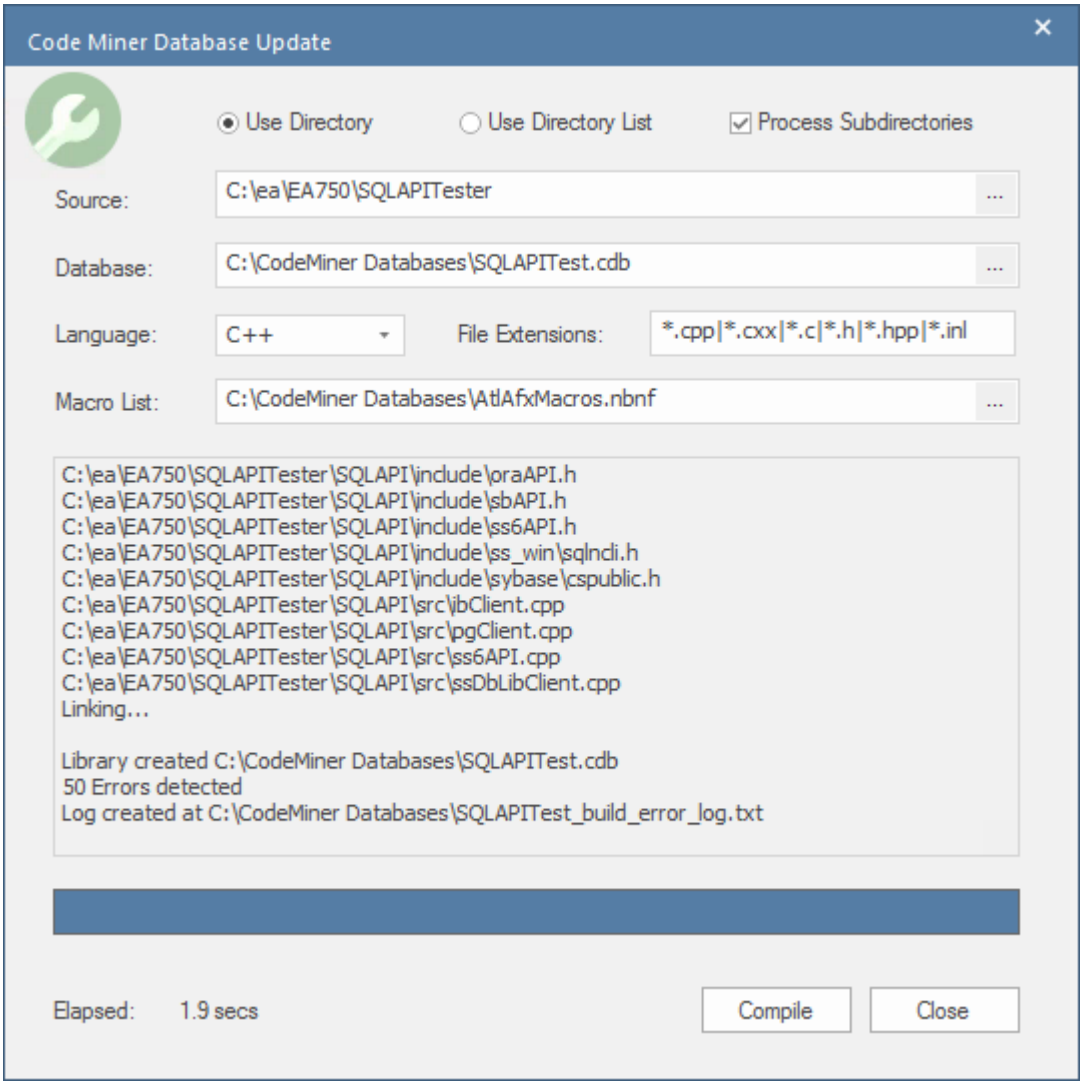
Users that develop a Custom language, will need to specify grammar rules for that language and save them into an nBNF file. Enterprise Architect's Grammar Editor is designed specifically for that purpose.

The Help Topic *Grammar Framework* provides detailed information on writing an nBNF grammar.

Updating a Code Miner Database

From time to time, you will want to update your Code Miner database. Typically, when you have made changes to your source code, but also after updating a grammar file or extending a macro file.

The process to update a database is very similar to creating a new database, but faster because you are not starting from scratch. Simply choose the menu option 'Update Database'. The 'Code Miner Database Update' dialog will display. The input fields will be populated with values from the last build. Proceed as for 'Creating a Code Miner Database'.



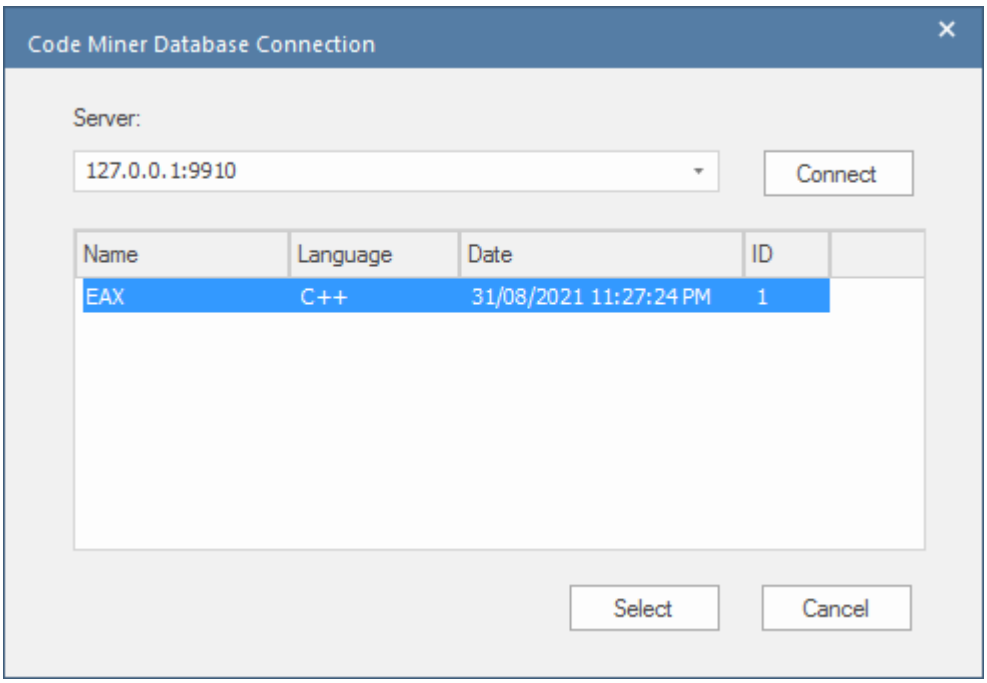
Selecting a Code Miner Database File

If you choose to use a library file for your Code Miner database, choose the menu option 'Browse for Database'. This will display a 'File Chooser', where you can browse for and select a *.cdb file.

Connecting to a Service

When connecting to a service, the dialog lists all databases hosted by the service.

You can choose to select an individual database in the list, or simply click the Select button, in which case queries will be executed across all databases listed by the service.



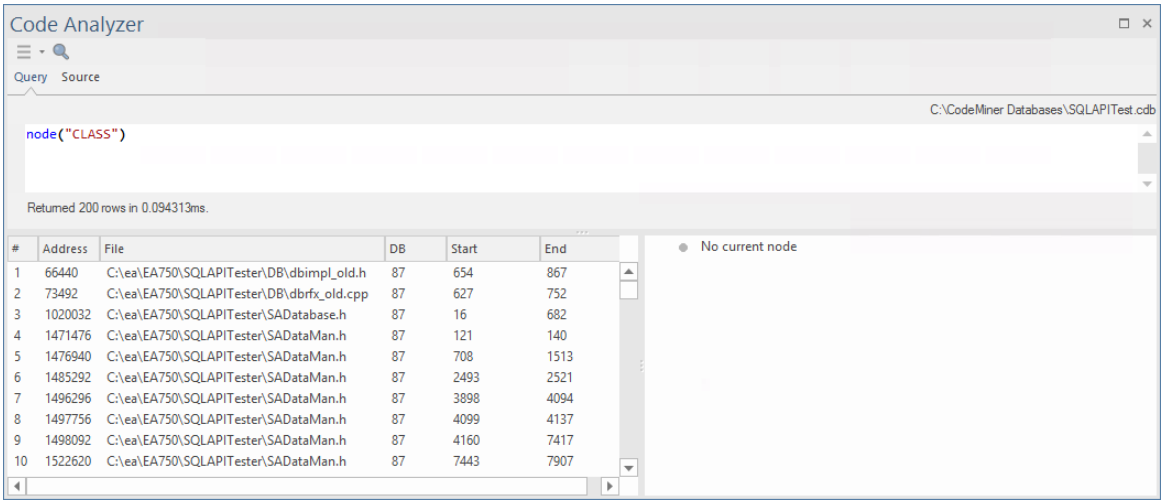
Running Queries

Once you have connected to a Code Miner database, you are ready to start running queries.

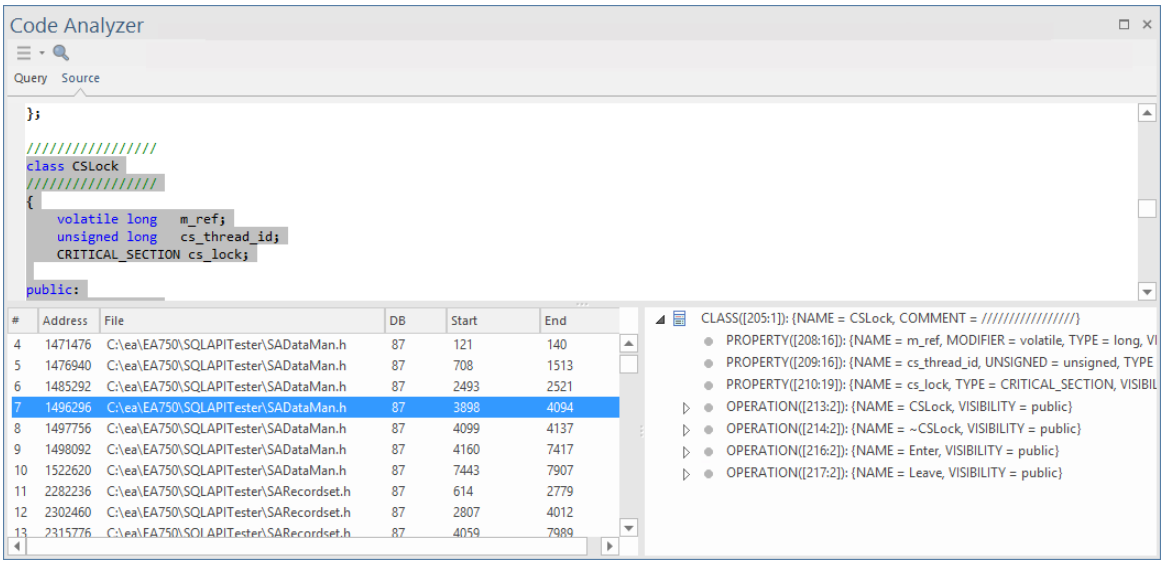
To run a query, select the Query tab in the Code Analyzer window, type in your query, then click on the icon to execute the query.

In this example, we have run a simple query node("CLASS"), which will return all 'Class' nodes found in

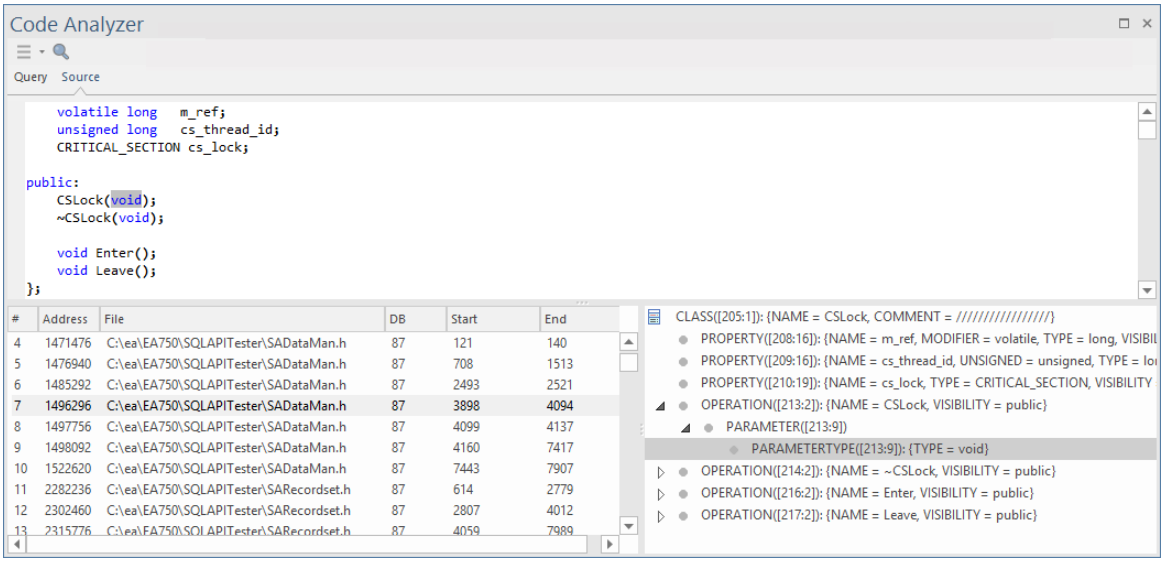
the Code Miner database.



By selecting a result in the lower-left panel, the 'Source' tab is activated and displays the source code corresponding to the selected node. Details for that class node are displayed in the lower-right panel.



Selecting a detail item in the lower-right panel, results in narrowing the selection within the source code, as shown here.



Query Example - Intersection

As an example, this mFQL query finds all the classes that have an operation named GetOption.

```
andat( "CLASS", item("OPERATION", "NAME", "GetOption"), node("CLASS"))
```

This clause returns a set of operations for which the 'NAME' value is "GetOption":

```
item("OPERATION", "NAME", "GetOption")
```

This clause returns a set of all Class nodes:

```
node("CLASS")
```

Formal syntax:

```
andat( string:rule, set:left, set:right)
```

'andat' takes the set of operations (left), applies the rule "CLASS" (only include rows that have a CLASS parent), then intersects that set with the set of all known classes (right). If the intersection succeeds, the operation node is added to the result set, otherwise it is excluded.

The Query Language - mFQL

The query language used with the Code Analyzer is described in full, in the *Code Miner Query Language (mFQL)* Help topic.

A brief description and some examples are also presented here.

The mFQL language is based on sets. Each statement works using the various types of set operations of which there are only a few.

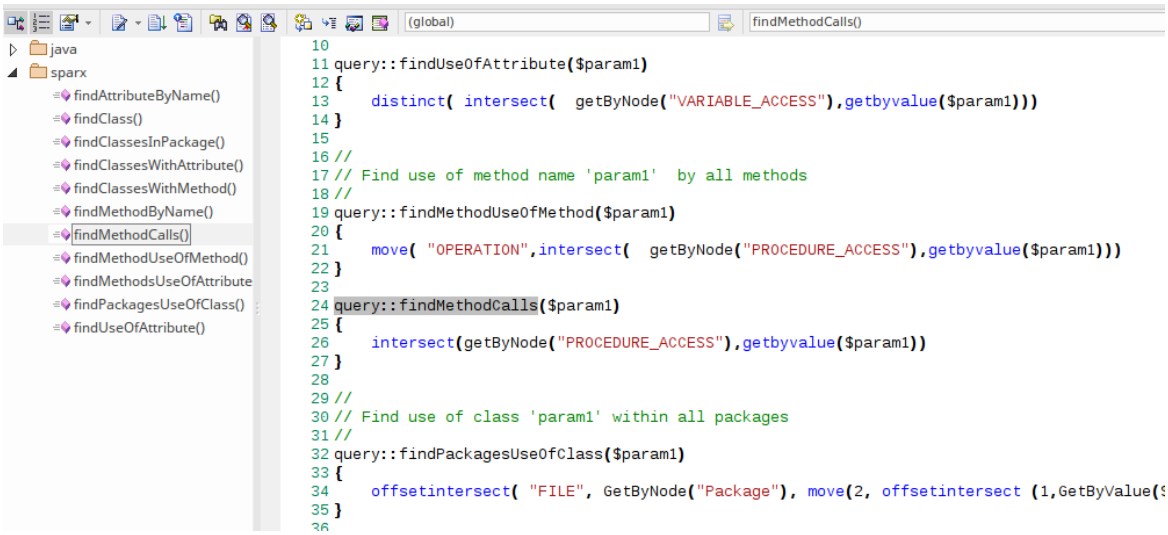
Code Miner Framework

The Code Miner system provides fast and comprehensive access to the information in existing source code. By parsing all source code and storing the resulting Abstract Syntax Tree in a read-optimized database, the system provides complete access to all aspects of the original source code, in a machine understandable format.

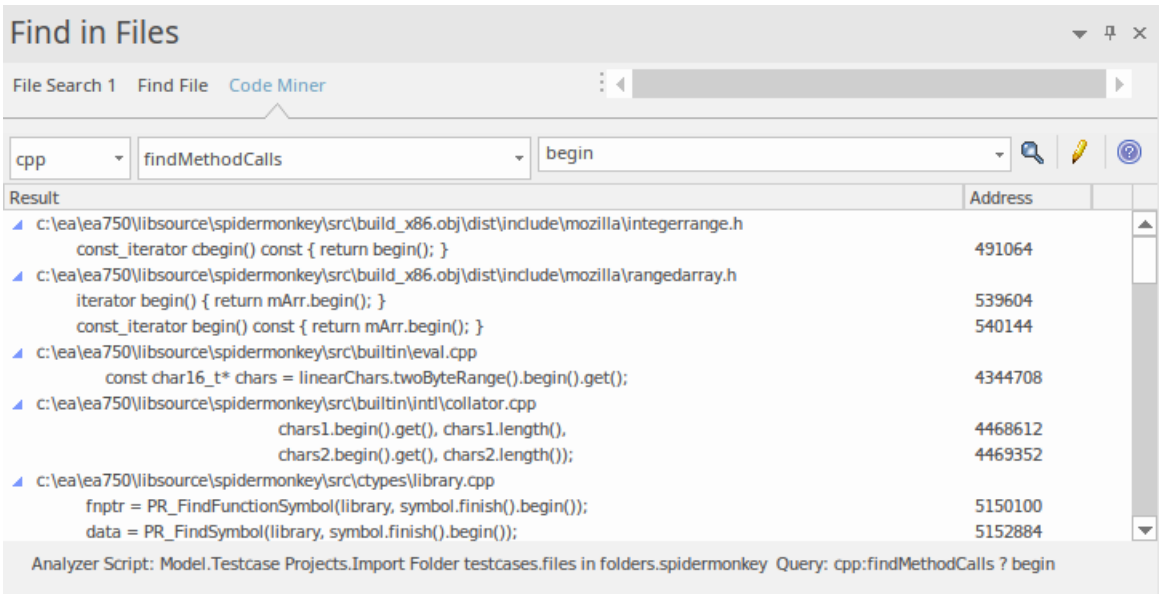
The core goal behind the system is to provide access to the data hidden within source code in a timely and effective manner. Great pains have been taken to ensure maximum performance, while providing the simplest interfaces possible. As a result the system can be used to analyze program structure, calculate metrics, trace relationships and even perform refactoring.

Information from Code Miner databases is retrieved using queries written in Code Miner NBNF Query Language (mFQL), Code Miner's own language. The language itself is reasonably simple, providing a small number of commands. Simple as the language is, it supports queries of arbitrary size and complexity. The design provides extreme performance for all queries, great and small.

This feature is available from Enterprise Architect Release 14.1.



Enterprise Architect's Code Analyzer, its search tools and the Intelli-sense features of its code editors all make use of the information mined from these databases.

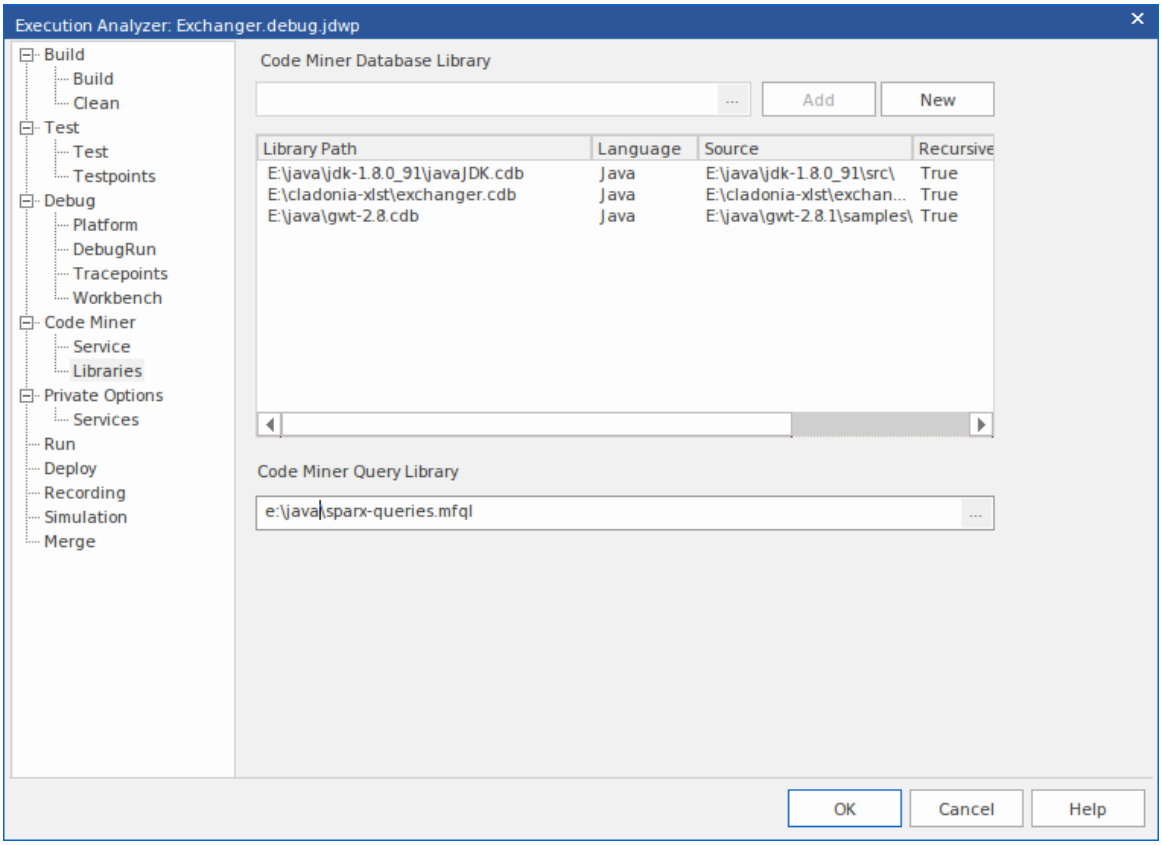


The currently active Analyzer Script, and also the query parameters, are indicated across the bottom of the 'Code Miner' page of the search tool.

Code Miner Libraries

Code Miner libraries are managed in Enterprise Architect using the Analyzer Script Editor. These Libraries are a collection of Code Miner databases, one of which would normally exist for each framework or project. The Analyzer Script Editor allows new databases to be created, and existing databases to be added, updated or removed. Together, these databases form the Code Miner Library used by the Code Analyzer and Intelli-sense features of Enterprise Architect. The library can be used locally, or it can be deployed to a server location where it can service multiple clients. You select the scenario to use on the 'Sparx Intel Service' page of the Analyzer Script.

This feature is available from Enterprise Architect Release 14.1.



Access

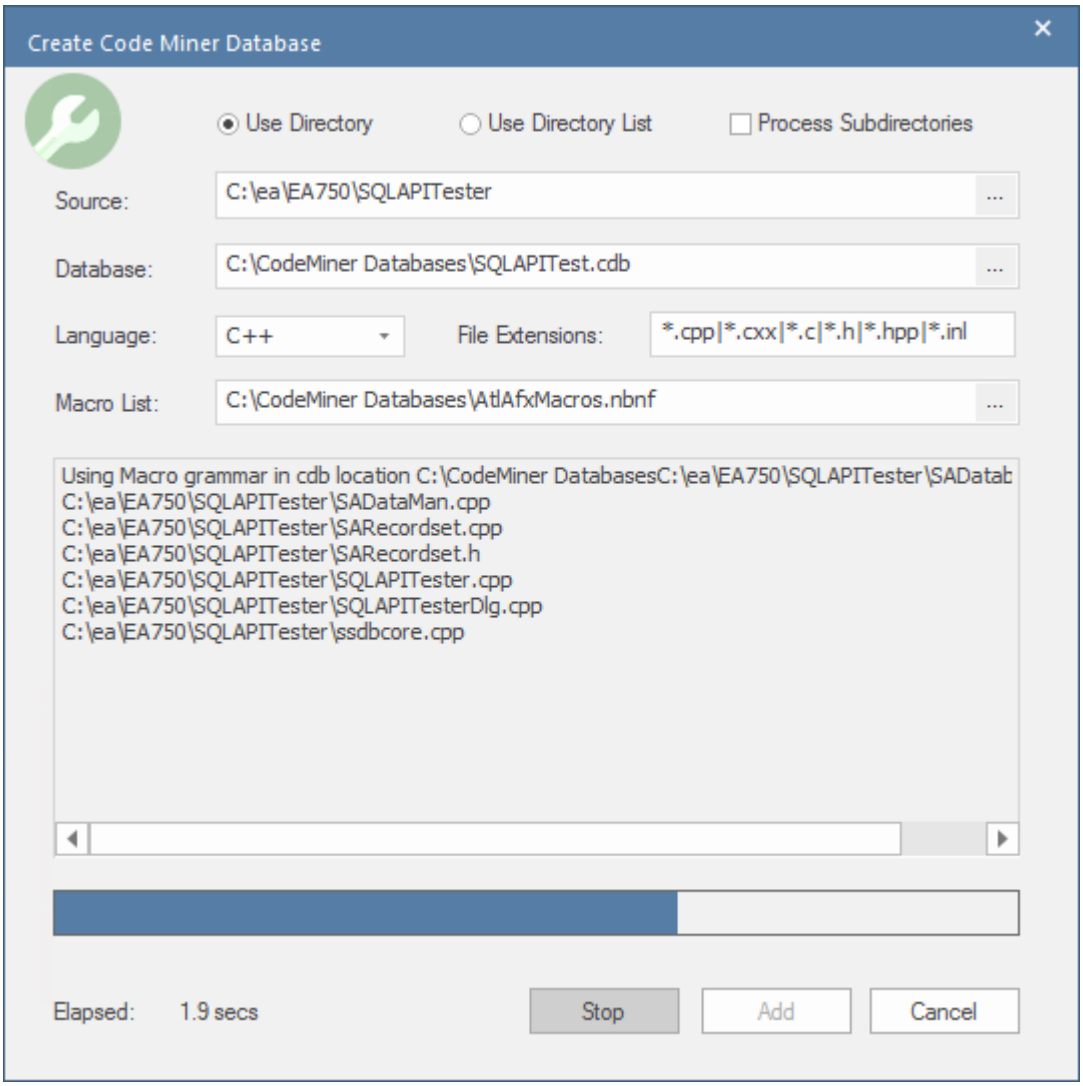
On the Execution Analyzer window, locate and double-click on the required script - the script editor dialog will display. On that dialog, select the 'Code Miner > Libraries' page.

Ribbon	Execute > Tools > Analyzer, or Develop > Source Code > Execution Analyzer > Edit Analyzer Scripts

Creating a New Database

On the 'Code Miner | Libraries' page of the Analyzer's Script Editor, use the 'New' button to create a new database.

In the 'Create Code Miner Database' dialog, specify the folder(s) containing the project source code, select the programming language and enter the destination path for the Code Miner database. When you click the 'Compile' button, details of the build are displayed in the log window.



When the process is complete press the 'Add' button to add the newly created database to the library.

For detailed information on creating new databases, please

see the Help topic *Creating a New Code Miner Database*.

Adding an Existing Database

Select an existing Code Miner database using the "..." selection button in the database path field.

(Code Miner databases have the .CDB file extension), then click on the Add button. Details about the database are listed in the library. The information presented displays the programming language grammar used to build the database. Also shown is the code base path parsed during the build and whether the parsing process was applied recursively through any sub directories.

Updating a Database

From time to time, as you update the source code for a project, you will want to update the Code Miner database built from that source code.

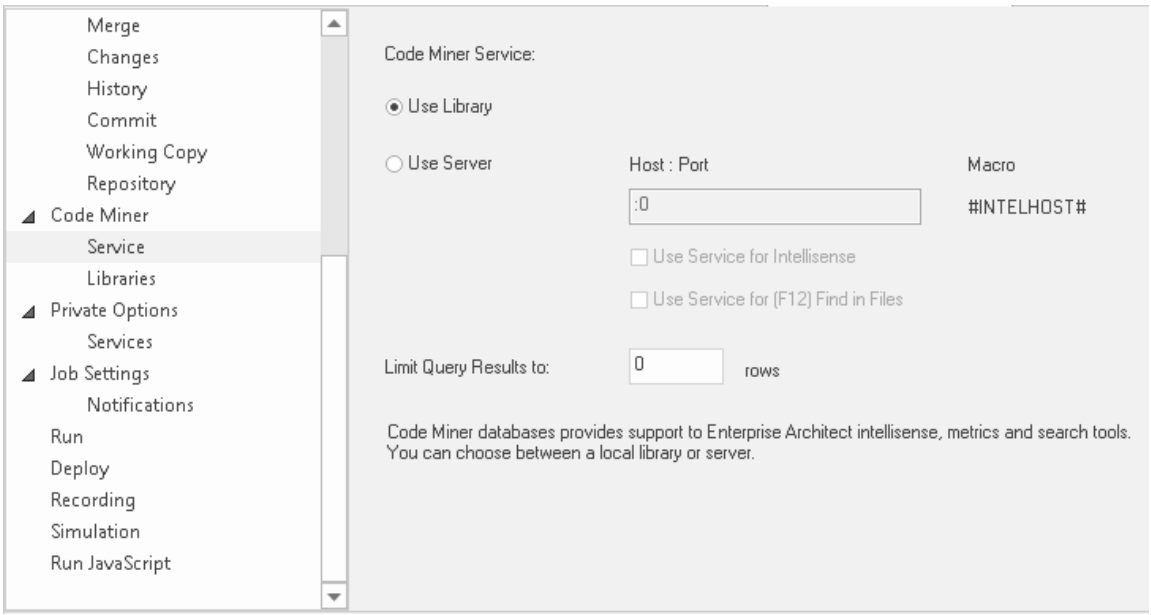
To update a single Code Miner database, select it from the list, right-click and choose 'Update selected' from its context menu. A dialog similar to the 'Create Database' dialog will display. Click on the 'Compile' button, the Code Miner will recreate the database from the updated code base.

Removing a Database

To remove a single Code Miner database, select it from the list and choose 'Remove Selected' from its context menu.

Configuring Enterprise Architect to use a Code Miner Library

In an Enterprise Architect Analyzer Script, choose the 'Sparx Intel Service' page and select 'Use Library'. Enterprise Architect then sources its Intelli-sense information from the databases listed in the 'Libraries' section of the currently active Analyzer Script.



Creating a New Code Miner Database

Enterprise Architect's Code Analyzer, the Intelli-sense features of its code editors and it's search tools all make use of Code Miner Databases.

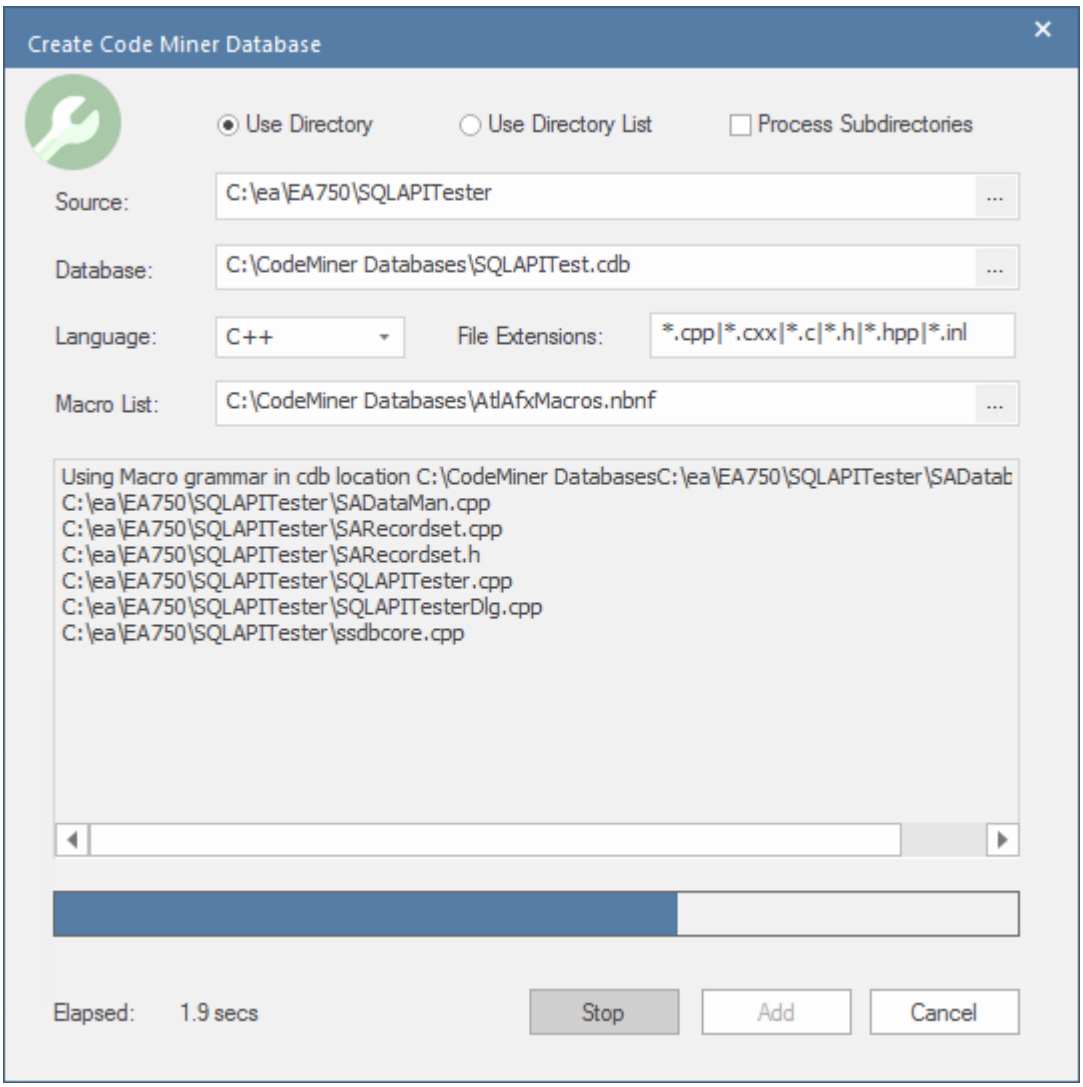
A Code Miner Database is created by parsing source code files according to grammar rules for the selected language and storing the resulting Abstract Syntax Tree, in a read-optimized database. One or more databases can be combined to form a Code Miner Library.

Access

Code Analyzer window	From the Code Analyzer window, click on the menu button, ≡, in the toolbar, then choose the menu option 'Create Database'.
Execution Analyzer Script Editor	With the Execution Analyzer's Script Editor window open, select the page 'Code Miner > Libraries', then click on the 'Create' button.

Create Code Miner Database Dialog

The 'Create Code Miner Database' dialog is used to initiate the process of parsing source code files to create a Code Miner database. On the dialog, you specify a range of inputs used by the process, such as Source Code folder, Language and Macro List file, as well as the output filename. The dialog fields are described in the table presented below.



Field	Description
Use	Select this option when all of the source files to be processed reside under one

Directory	directory. When this option is selected, the checkbox 'Process Subdirectories' is enabled.
Use Directory List	Select this option when your project source code resides in multiple separate directories. In this case, you use the 'Source' field to specify a file that contains a list of directories containing the source code to be processed.
Process Subdirectories	This check-box is enabled when the 'Use Directory' option is selected. When selected, source code file residing within any subdirectories of the specified 'Source' directory will also be processed.
Source	This field is used to specify the directory (or directories) containing source code files that will be processed to create the Code Miner database. When the option 'Use Directory' is selected, this field is used to specify the root folder in which to search for source code files. When the option 'Use Directory List' is selected, this field is used to specify a user created file containing a list of path

	names to the directories that contain the source files to be processed. Clicking the ... button opens a 'File Chooser' dialog, that allows you to browse for and choose a file with the extension '.ssdirlist'. For more information, see the section *Directory List File* below.
Database	This field specifies the full path name of the Code Miner database file that will be created. The filename extension '.cdb' is used for this file.
Language	This is a drop-down list, where you specify the language used in the source code files being processed. There are a number of languages for which Enterprise Architect provides 'built-in' support. (There are built-in grammars used for parsing the supported languages). There is also an option to choose a 'Custom' language. If you choose to use a custom language, you will need to create your own grammar to support parsing of that language. When the 'Custom' option is selected, the field 'Grammar File' will be displayed, allowing you to specify the file that defines your custom grammar.

File Extensions	This field lists a number of filename extensions that are typically associated with source code files of the chosen language. Only files with filename extensions matching those in the list will be processed by the parser. You can add or remove filename extensions to suit your needs.
Macro List	When the language selected is 'C++', the 'Macro List' selection field is displayed. The Macro List field lets you specify a file that provides a list of macros that the parser should skip when it encounters them. For the C++ language, macros present a problem to the parser because they hide native language constructs. Adding the name of a macro to the Macro List file and updating the database will usually clear all the errors related to that macro. For more information, see the section *Extending the Macro List File* below.
Grammar File	Sparx Systems has developed grammars for all of the languages listed in the drop-down selection list. C++, C#, Java, XML and also MDGTechnology.

	There is also an option to select a 'Custom' language. Users that develop a Custom language, will need to specify grammar rules for that language and save them into an nBNF file, so that the Code Miner can correctly parse source code written in that language. Enterprise Architect's Grammar Editor is designed specifically for that purpose. When you select "Custom" as the language, you should then specify the grammar file you created for that language, so that the Code Miner can correctly parse your source code. The Help Topic *Grammar Framework* provides detailed information on writing an nBNF grammar.
Output Window	The output window shows the progress of parsing the source code files. Upon completion, it also shows the names of the database file and the log file that were created along with the number of errors encountered.
Compile/Stop button	The 'Compile' button is used to start the processing operation. This button changes to a 'Stop' button once processing begins, allowing the user to

	abort the operation.
Add button	Once a database has been compiled, the 'Add' button can be used to add that database to a Code Miner Library. Multiple databases can be added together to build up a library that covers many source code projects. **Note**: When the 'Create Code Miner Database' dialog is opened from the Code Analyzer window, the 'Add' button is not displayed.

Directory List File

If you choose to specify a Directory List file, you will need to create a simple text file using the filename extension '.ssdirlist', that lists the full path to each directory you wish to process, with one path per line. For example:

c:\myprojects\project1\tools\scintilla

c:\myprojects\project2\src

d:\mylibs\lib1\src

If you wish to recursively process the subdirectories within a listed directory, precede that path with an exclamation

mark like this:

!d:\mylibs\lib1\src

Any line that begins with a # character is treated as a comment:

include scintilla

c:\myprojects\project1\tools\scintilla

Extending the Macro List File

For the C++ language, macros present a problem for grammars because they hide native language constructs. The parser cannot not perform substitution on macros as they are often defined conditionally and the parser has no idea about the architecture. The Macro List file provides a list of macros that the parser should skip when it encounters them.

When you build a Code Miner database for a C++ source code repository, you may see errors listed. When an error occurs, use the error log to find and inspect the line of code that caused the error. This almost always identifies a macro that is causing the grammar failure. Adding that name to the macro list and updating the database will usually clear all the errors related to that macro.

For example, the error log shows the following error:

C:\ea\EA750\SQLAPITester\SQLAPI\include\asa\sqlfuncs.

h, line:12, col:18, Unexpected symbol ','.

Upon inspection, the line of code causing the error is this:

```
    FUNC_INFO( extern, void, _esqlentry_, sqlstop,
(SQLCA *))
```

(There are also many other similar lines using the macro 'FUNC_INFO'.)

So, we edit the default Macro List file, 'AtxAflMacros.nbnf', adding the following line:

```
    "FUNC_INFO"    "(" skipBalanced("(", ")") ")" |
```

This line instructs the parser, upon encountering the macro "FUNC_INFO", to apply the function skipBalanced("(", ")"), which takes two parameters, in this case they are the opening and closing parentheses. So, the parser is instructed to ignore everything in between the opening and closing parentheses.

When the change to the Macro List file is saved and the database is recompiled (updated), all of the errors pertaining to the macro "FUNC_INFO" have been eliminated.

Learn more

- Grammar Framework
- Code Analyzer

Code Miner Queries

Code Miner queries are best considered as functions written in the Code Miner NBNF Query Language (mFQL). As such, they have unique names, can be grouped by namespace and can take one or more parameters. Queries are bundled together into one source file. This source file is identified to Enterprise Architect by naming it in your Analyzer Script.

When specified, the queries it contains are available in the Code Miner control. Parameters to these queries can be taken from selected text in a code editor, the model context or typed directly into the search field of the control.

This feature is available from Enterprise Architect Release 14.1.

```
188
189 namespace java
190 {
191 //
192 // Find all references
193 //
194 query::findByName($param1)
195 {
196     distinct(GetByValue( $param1 +))
197 }
198
199 query::findMethodByName($name)
200 {
201     move( 1, "METHOD", intersect( GetByNode("NAME"), GetByValue( $name ) )  )
202 }
203
204 query::findMethodCall($name)
205 {
206     filter( "METHOD_ACCESS", intersect(GetByNode("NAME"), GetByValue( $name )) )
207 }
208
```

This image illustrates an mFQL query from the Sparx Queries file distributed with Enterprise Architect installations. The syntax for composing an mFQL query and the mFQL language itself is described here.

Query Syntax

The syntax for composing mFQL queries is:

namespace
{
    query:*name*([$*param1 [*, $*param2]]*)
    {
        mfql-expression
    }
}

where:

- *namespace* names the collection of queries
- *name* is the 'function' name of the query
- *$param1 and 2* are placeholders for argument substitutions at runtime
- *mfql-expression* is an mFQL expression

Code Miner Query Language (mFQL)

The Code Miner system provides fast and comprehensive access to the information in existing source code. By parsing all source code and storing the resulting Abstract Syntax Tree (AST) in a read-optimized database, the system provides complete access to all aspects of the original source code, in a machine understandable format.

The core goal behind the system is to provide access to the data hidden within source code in a timely and effective manner. Great pains have been taken to ensure maximal performance, while providing the simplest interfaces possible. As a result the system can be used to analyze program structure, calculate metrics, trace relationships and even perform refactoring.

mFQL

mFQL is the query language of the Code Miner. The language itself is reasonably simple, providing a small number of commands. Simple as the language is, it supports queries of arbitrary size and complexity. The design provides extreme performance for all queries, great and small.

The language is set-based; it operates primarily on sets of abstract data obtained through discrete vertical indices. For our purposes, a set is an ordered array of numbers, each of which is a pointer to a node in the AST Store. A discrete

vertical index provides a mechanism to retrieve sets by discrete value.

The language includes the three basic set-joining operations. These are 'intersect', 'union', and 'except'. The 'except' join is, more precisely, a 'symmetric difference' join. A 'complement' join can be achieved by using a short sub-query; this is detailed in the 'except' join documentation. The 'offsetIntersect' join is also discussed in detail there.

The Code Miner database provides three discrete vertical indices in its AST Store. These indices are 'node name', 'attribute name', and 'attribute value'. Each vertical index can be queried for a discrete value, which will return a set of all nodes where that value is present. The three vertical indices are queried using the functions 'getByNode', 'getByName' and 'getByValue', respectively.

Set 'traversal routines' provide mechanisms to filter sets based on patterns in the AST. The traversal routines are either destructive (move) or non-destructive (filter). Destructive traversals modify the set member values to point to the target node; non-destructive traversals ensure the target node exists. In both cases, nodes that cannot complete the traversal are removed.

Please note that all traversals in mFQL are upwards. Downwards traversals are technically complex, as a node could have any number of child nodes. Conversely, upward traversals are much simpler, with every node having zero or one parent node. For these reasons, downward traversals are not supported in the query language.

Although there are only a small number of operations in

mFQL, the language is capable of expressing very finely grained and complex queries. The language is functional in design, and supports arbitrary nesting calls.

mFQL queries execute at lightning speed. The backend database was designed from the ground up for read performance. The query parser was hand optimized. Knowing that it always has pure ordered sets, the low-level code takes several shortcuts to perform joins with minimal work effort.

In order to use nBNF effectively one must possess a working knowledge of the target language, and an intimate knowledge of the grammar used to parse it.

The mFQL Language

This section provides a list of Code Miner NBNF Query Language (mFQL) queries with explanations and comments.

The queries shown here demonstrate different capabilities and different approaches to exploring and extracting data using mFQL and the Code Analyzer in Enterprise Architect. The mFQL queries help make the syntax human-readable and intuitive, and have been extended in Enterprise Architect to include additional functions necessary to do real things with Code Miner databases.

The Query Language

String parameters are indicated by **string**, set parameters are indicated by **set** and number parameters are indicated by **numbers**.

Notes

1. Case sensitivity is defined by the case sensitivity of the language of the source code used to populate the database. If the source language is case sensitive (such as C++) all string literal parameters are case sensitive. If the source language is case insensitive (such as SQL) all string literal parameters are case insensitive.
2. Hierarchical traversals in mFQL are generally upwards. Downwards traversals are not optimal, as a node might have any number of child nodes. Upward traversals are

much simpler, with every node having zero or one parent node. Downward-looking queries such as 'children' only query one level down.

3. Synonyms of some keywords are provided to better express a query intent or action in particular circumstances, and to support legacy queries. Synonyms are simple alternatives for the base function keyword. For example, 'type(str)' can be written as 'node(str)' or 'byNode(str)' or 'getByNode(str)'. The current specified version is the preferred one, with the synonyms only intended for use in exceptional circumstances.

Statement	Description
type(value)	type(value) Extracts a set based upon node name. The exact name for a node is defined by the grammar used to parse the original source. In this example, find all nodes within the database of type "CLASS". type("CLASS") *Synonyms:* • node • byNode • getByNode
with(name)	with(name) Searches the database for any element

	that has a named attribute matching the search string. The value of the attribute is ignored - this is a query for the attribute NAME only. All nodes with one or more attributes of the specified name are returned. If a single node has two attributes of the same name, one instance of that node is returned. This example will find all elements in the database that have an attribute named "Type": with("Type") *Synonyms:* • name • byName • getByName
find(value) find([+] value [+ value] [+])	find(value) find([+] value [+ value] [+]) Search the database for any element having an attribute value with the provided search term. The match is case sensitive and must match the whole word. You can extract a set based upon an attribute value; when extracting nodes by attribute value, the values of all attributes for the node are considered.

	Wildcards allow for specifying a subset of attribute values for a node. Wildcards can be used at either the beginning or end of a value specification: • A leading concatenation symbol allows for any number of attributes preceding the first matched attribute • A trailing concatenation symbol allows for arbitrary trailing attributes In both cases, if the node would match without wildcards, it will match with them – the wildcard specifies any number of leading/trailing attributes, including none. In this example, we retrieve a set of nodes that have their last two attributes being “.” and “sun”. The leading concatenation symbol specifies that any number of attributes (including none), with any value, can exist before the matched attributes, but none can follow. find(+ “.” + “sun”) The next example has a trailing wildcard. Any node with “com”, “.” and “sun” as the first three attributes will be returned. Any number of trailing attributes can exist. find(“com” + “.” + “sun” +)

	Both wildcards can be used together. In this example nodes with attributes with the three specified values as names, in order, regardless of leading or trailing attributes, will be returned. find(+ “com” + “.” + “sun” +) Example: Find all nodes in the database that have any attribute with a value of "CString": find("CString") Example: Find all nodes in the database with a set of attributes having these values in this order: find("com" + "." + "sun") *Synonyms:* • value • byValue • getByValue
has(name,value)	has(name, value) Finds all elements that have a named attribute with the value supplied. Unlike the intersection of 'find' and 'with', this query will only return rows with an exact name/value pair. has("Type","CString")

having(name, value, set)	having (name, value, set) Finds all elements within the supplied set that have a named attribute with the given value. Similar to 'has' but supplies a predefined input set to search. Whether to use 'has' or 'having' is generally determined by the kind of query structure being used, its depth and its readability. Example 1: Find all Property elements with a name of "m_strName" that have a Type attribute of CString: having("Type","CString",this("PROPERTY","NAME","m_strName")) Example 2: Extend Example 1 to only include those that store a CString *: having("Reference","*", having("Type","CString",this("PROPERTY","NAME","m_strName")))
this(type,name,value)	this(type, name, value) Function finds one or more elements that have a matching TYPE, and WITH a named attribute having the specified VALUE. Example: Find all operations named

	"Import Solution": this("OPERATION","NAME","ImportSolution") *Synonyms:* • object • item
like(name,like,set)	like(name, like, set) Finds a set of elements that have an attribute that starts with the search sub-string. Note that this is not a fully wild-carded search but is case sensitive and must be an exact match for the length of the search string. Example: Find all Classes in the database whose NAME attribute starts with "CMapStr": like("NAME","CMapStr",gettype("CLASS"))
and(set1,set2,...)	and(set1, set2, ...) Returns the intersection of nodes between two or more sets. To be included in the final set, an element must exist in ALL the input sets.

	Synonyms: • intersect(set, set,...) • {set, set, ...}
union(set1,set2,...)	union(set1,set2, ...) Returns the distinct union of ALL nodes present in the input sets. *Synonyms:* • or(set, set ...) • [set, set ...]
ancestor(str,set)	ancestor(str, set) ancestor(num, set) ancestor(num, str, set) The ancestor function traverses each node in a set of a number of parent nodes, excluding any nodes that fail the traversal. The number of nodes to traverse, the name of the target node for the traversal, or both can be provided as parameters. When the number of nodes is provided, but the target node name is not, any nodes with the specified number of parents will pass the traversal. Any node

	that runs out of parents will be dropped from the set. When the name of the target is specified, but the number of nodes to traverse is not, any nodes with a parent with a matching name, at any point in the hierarchy, will pass the traversal. Any node with no matching parent is excluded. When both the number of nodes and the target name are provided, only nodes that have a parent node with the specified name, at the specified offset, pass the traversal. All other nodes are removed from the set. In this example the set hasParameter("CString","&",1) is moved up to an ancestor node named "OPERATION". If the move fails the node is dropped from the result. ancestor("OPERATION",hasParameter("CString","&",1)) In this example the set is moved up one rung to its parent. If there is no parent, the node is dropped from the result. ancestor(1,hasParameter("CString","&",1

	)) In this example the set is moved up three steps to its parent->parent->parent . If there is no such node, the node is dropped from the result. ancestor(3,hasParameter("CString","&",1)) *Synonyms*: • move
filter(str,set)	filter(str, set) filter(num, set) filter(num, str, set) The filter function is the same as the 'ancestor' function, except that it returns nodes from the original child set rather than new ancestor nodes. If a node is unable to pass the specified traversal, it is removed from the set. Nodes that pass the traversal are left in place, unmodified. In this example the set hasParameter("CString","&",1) is tested for an ancestor node named "OPERATION". If the move fails the node is dropped from the result. The result set is a set of parameter types that

	meet the criteria. filter("OPERATION",hasParameter("CString","&",1))
match(NameA,setA,NameB,setB)	match(NameA,setA, NameB,setB) 'Match' takes two input sets and two attribute names and returns all those in 'setA' that have a matching record in 'setB', as determined by comparing the values of the named attributes 'strA' and 'strB'. That is, a 'setA' row is included if the value of attribute 'strA' in 'setA' exists in 'setB' as the value of an attribute of name 'strB'. 'Match' is useful for finding where one element feature is used in a different context elsewhere in the database. For example, where a unique element name or GUID is referenced by another element. In this example, we match the attribute named 'TYPE' from the right set to the attribute 'NAME' in the left set. The result will be all "CLASS" type objects from the left set with NAME == TYPE(s) as specified in the right set. match("NAME",type("CLASS"),"TYPE",this("PROPERTY","NAME","m_pLink"

	))
graph(targetType, targetName, linkType, linkName, start)	graph(targetType, targetName, linkType, linkName, start) Find a recursive set of elements that form some kind of graph when linked by attribute pairs, in a manner similar to 'match'. The starter set is queried for all owned instances of the linkType with link Name and these are matched against a new query based on the targetType with targetName. The new set is filtered in a manner similar to 'match', and all elements in the new query that share the same NAME/VALUE pair as from the starter set are kept; all others are discarded. The resultant set is then fed back into the original set as the starter for the next iteration, with the results at each stage being added together to form the final result set. Example: Return the Class hierarchy for a Class named "Car". graph("CLASS","NAME","GENERALIZATION","GENERAL",this("CLASS","NAME","Car"))
prune(set_tes	prune(set_test, str, set_base)

t,str,set_base)	prune(set_test, num, set_base) For two sets of nodes, temporarily move one set UP to the named or numeric position in its ancestry and filter out any nodes that do not exist by strict intersection in the TEST set. The first set is the TEST set, the right or last set is the BASE set. The set returned is all the elements in the BASE set that, when moved to the TEST position, matched something in the TEST set. The returned nodes are the original nodes from the BASE set and are not moved up when returned. Example 1 finds the set of parameter types used for operation parameters named "CustomerName" across the whole database. prune(this("PARAMETER","NAME","CustomerName"),"PARAMETER",type("PARAMETERTYPE")) Example 2 finds all Properties of a Class named Customer, assuming the grammar used to compile the database placed the Property definition two hierarchy levels below the Class definition.

	prune(this("CLASS","NAME","Customer"),2,type("PROPERTY"))
andat(str,test, base)	andat(str, base, test) andat(num, base, test) andat(num, str, base, test) For two sets of nodes, temporarily move one set UP to the named or numeric position in its ancestry and filter out any nodes that do not exist by strict intersection in the TEST set. The first set is the TEST set, the right or last set is the BASE set. The set returned is all the elements in the BASE set that, when moved to the TEST position, matched something in the TEST set. The returned nodes are the original nodes from the BASE set and are not moved up when returned. Similar to 'prune', this query supports additional options and structures the inputs in a different order to facilitate different kinds of stacked searches. The 'andat' function performs both a non-destructive tree traversal and an intersect join in one operation. Each node in the left set is traversed according to parameters provided, then the result of

	the traversal is intersected with the right set. If the intersect passes, the original node is added to the result set. If the intersect fails, the node is excluded from the result set. The traversal parameters for 'andat' are the same as for 'ancestor' and 'filter'. For more information about the traversal parameters, see the 'ancestor' function. Example: For the set of all "PROPERTY" nodes in the database, move them up to a parent node of type CLASS and then intersect the result with the right hand set - in this case a CLASS named CDiagram. All nodes that pass this test are returned as PROPERTY nodes, effectively giving the set of all properties of the Class CDiagram. andat("CLASS",type("PROPERTY"),this("CLASS","NAME","CDiagram")) *Synonyms:* • offsetIntersect • offx
unique(left,right) / except(left,right)	unique(left, right) except(left, right) Except joins return sets that contain any nodes from either set that do not appear

	in both sets. This join is similar to a bitwise XOR operation. In set theory, this type of join is referred to as a 'symmetric difference join'. {1, 2, 3} excepted with {2, 3, 4} results in {1, 4}
omit(left,right) / exclude(left,right)	omit(left, right) exclude(left, right) Exclude joins return a set that contains all nodes from the left set that do not appear in the right set. In set theory, this type of join is referred to as a 'relative complement join'. {1, 2, 3} complemented with {2, 3, 4} results in {1}
differ(name,set,name,set)	differ(name, set, name, set) Return a set of nodes that do not have a matching row in another set, using a NAME/VALUE pair from each set to match on. Example: This more complex example tests the complete set of Generalizations for a Class hierarchy and identifies missing or unresolved Class names in the total inheritance hierarchy. Like the 'match()' function discussed later, this function iterates over attribute

	name/value pairs as specified in the left and right input sets, but only includes rows in the final set where there is NO match. differ("GENERAL", children("GENERALIZATION", graph("CLASS","NAME","GENERALIZATION","GENERAL", this("CLASS","NAME","CMainFrame"))), "NAME", graph("CLASS","NAME","GENERALIZATION","GENERAL", this("CLASS","NAME","CMainFrame")))
children(type,set)	children(type, set) Return a set of child nodes of a specified type for one or more parents in the source set. For all children regardless of type, use an empty string. For example, in the first query we return ALL first level children of the CMainFrame Class. In the second query

	we restrict the nodes returned to be of type "REGION" only. children("",this("CLASS","NAME","CMainFrame")) children("REGION",this("CLASS","NAME","CMainFrame"))
childcount(num,type,set)	childcount (num,type,set) Return nodes that exactly match the number of specified children of a specified type. For example, only return operations that have 5 parameters. An example usage is in specifying an exact operation signature, so we check firstly that parameter1 and parameter2 match the type we are querying for, then move those to their operation ancestor and intersect the result with the operation name "GetFromCache" we are interested in. To rule out spurious hits with operations having more than 2 parameters, we explicitly add childcount(2, ...) to ensure we only get operations that have 2 parameters. childCount(2,"PARAMETER", {

	ancestor("OPERATION",hasParameter("CString","&",1)), ancestor("OPERATION",hasParameter("CString","&",2)), this("OPERATION","NAME","GetFromCache") })
byAddress(num)	byAddress(num) The byAddress function is used in applying the results of one query to another. For example, we might have a node of particular interest, and want our query to return only nodes that join (in some way) to the specified node. byAddress(node: number) This example builds a set containing the single node related to the address specified: byAddress(11256)
byPosition(File, Offset)	byPosition(File, Offset) The byPosition function is used to return the inner-most node that covers a certain position in a file. This function is useful

	for locating a position in the AST based upon a file position.
distinct(set)	distinct(set) The distinct function ensures that a set has no duplicate values. All duplicate values are excluded from the result set.

Set Extraction

These procedures extract sets from discrete vertical indices. There are three indices available, each with a specific extraction function. String literal parameters to these functions could be case sensitive. Case sensitivity is defined by the language of the source code used to populate the database. If the source language is case sensitive (as C++ is) all string literal parameters are case sensitive. If the source language is case insensitive (as SQL is) all string literal parameters are case insensitive.

type

type(value: string)

Extract a set based upon a node name. The exact name for a node is defined by the grammar used to parse the original source. In this example, all nodes with the name "OPERATION" are returned.

type("OPERATION")

_

with

with(value: string)

Extract a set based upon attribute name. All nodes with one or more attributes of the specified name are returned. If a

single node has two attributes of the same name, one instance of that node is returned. This example returns all nodes with one or more attributes named "NAMEPART".

with("NAMEPART")

find

find([+] value: string [+ value: string] [+])

Extract a set based upon an attribute value. When extracting nodes by attribute value, the value of all attributes for the node are considered. Wildcards allow for specifying a subset of attribute values for a node.

When a single value is provided, all nodes that have a single attribute with the value specified are returned. If a node has any other attributes, it is excluded. In this example, all nodes with exactly one attribute with the value of 'i' are returned.

find("i")

More than one value can be specified by using a concatenation symbol. When more than one value is specified, the resulting set will contain all nodes that have attributes with exactly the values specified, in the order specified. Any node with extra leading or trailing attributes is excluded. This example retrieves a set of all nodes with a set of three attributes with the values “com”, “.” and “sun”, in that order.

find("com" + "." + "sun")

Wildcards can be used at either the beginning or end of a

value specification. A leading concatenation symbol allows for any number of attributes preceding the first matched attribute. A trailing concatenation symbol allows for arbitrary trailing attributes. In both cases, if the node would match without wildcards, it will match with them – the wildcard specifies any number of leading/trailing attributes, including none.

In this example, we retrieve a set of nodes that have their last two attributes being “.” and “sun”. The leading concatenation symbol specifies that any number of attributes (including none), with any value, can exist before the matched attributes, but none can follow.

find(+ “.” + “sun”)

The next example has a trailing wildcard. Any node with attributes “com”, “.” and “sun” as the first three attributes will be returned. Any number of trailing attributes can exist.

find(“com” + “.” + “sun” +)

Both wildcards can be used together. In this example, nodes with attributes named as the three values specified, in order, regardless of leading or trailing attributes, will be returned.

find(+ “com” + “.” + “sun” +)

—

Set Traversal

ancestor

ancestor(count: number, source: set)

ancestor(value: string, source: set)

ancestor(count: number, value: string, source: set)

The 'ancestor' function traverses each node in a set up a number of parent nodes, excluding any nodes that fail the traversal. The number of nodes to traverse, the name of the target node for the traversal, or both can be provided as parameters.

- When the number of nodes is provided, but the target node name is not, any nodes with the specified number of parents will pass the traversal; any node that runs out of parents will be dropped from the set
- When the name of the target is specified, but the number of nodes to traverse is not, nodes with a parent with a matching name at any point in the hierarchy will pass the traversal; any node with no matching parent is excluded
- When both the number of nodes and the target name are provided, only nodes that have a parent node with the specified name at the specified offset pass the traversal; all other nodes are removed from the set

It is possible - even likely - that these calls will generate sets having duplicate values. This is by design, as the concrete

rules for sets do not define them as being discrete. If (as in most cases) you want your set to be discrete, use the 'distinct' function described in the *The mFQL Language* Help topic.

This sample extracts a set of all nodes named 'OPERATION', then traverses each node up one level to its immediate parent. Any 'OPERATION' node with no parent is excluded.

ancestor(1, getByNode("OPERATION"))

This sample extracts a set of all nodes named 'OPERATION', then traverses each node up to the first 'CLASS' parent node. Any 'OPERATION' node with no 'CLASS' parent is excluded.

ancestor("CLASS", getByNode("OPERATION"))

This sample extracts a set of all nodes named 'OPERATION', then traverses each node up one level to its immediate parent. If the parent node is not a 'CLASS' node, or the node fails to traverse though a lack of parent nodes, it is excluded.

ancestor(1, "CLASS", getByNode("OPERATION"))

_

filter

filter(count: number, source: set)

filter(value: string, source: set)

filter(count: number, value: string, source: set)

The 'filter' function is the same as the 'ancestor' function, except that it does not modify nodes – it is non-destructive. If a node is unable to pass the specified traversal, it is removed from the set. Nodes that pass the traversal are left in place, unmodified.

It is often desirable to filter a set by the current node name. This can be used to ensure that the nodes returned from a 'with' or 'find' call are of a particular node type. This example returns all nodes with an attribute with the value of “CFoo”, where the resulting node is a “TYPE” node.

filter(0, “TYPE”, find(“CFoo”))

For more details on the use of the 'filter' function, see the 'ancestor' function.

—

Set Joining

and

and(left: set, right: set)

An 'and' join will return a set containing all nodes that exist in both the left and right set. This join is comparable to a bitwise AND operation. In set theory, this type of join is called an 'intersection'.

{1, 2, 3} intersected with {2, 3, 4} results in {2, 3}

This example returns a set that contains all nodes that have a single attribute with the name of "TYPE" and the value of "int".

```
and(
find("int"),
with("TYPE")
)
```

union

union(left: set, right: set [, right: set])

'Union' joins return a set that includes all nodes found in either the left or the right set. This join is used to combine

the results of two or more sub-queries into a single set. A 'union' join is similar to a logical OR operation. In set theory, the 'union' join is known as a union.

The 'union' join is able to operate on more than two sets. The result is a set that contains all nodes from all supplied sets. The 'union' join is the only join able to operate on more than two sets.

The result of a 'union' join is always a discrete set, unless one of the source sets contained duplicates. This means that duplicates in source sets will be preserved, but the 'union' join itself will not generate duplicates.

{1, 2, 3} unioned with {2, 3, 4} results in {1, 2, 3, 4}

This sample creates a set containing all nodes with an attribute named “TYPE” or a single attribute with the value of “int”.

```
union(
find("int"),
with("TYPE")
)
```

except

except(left: set, right: set)

'except' joins return sets that contain any nodes from either set that do not appear in both sets. This join is similar to a bitwise XOR operation. In set theory, this type of join is

referred to as a 'symmetric difference' join.

{1, 2, 3} excepted with {2, 3, 4} results in {1, 4}

For more information on the 'symmetric difference' join in set theory, see https://en.wikipedia.org/wiki/Symmetric_difference

This sample returns a set of all nodes with an attribute named "TYPE" but no single attribute with the value of "int", plus all nodes with an attribute with the value of "int" that are not named "TYPE".

```
except(
 find("int"),
 with("TYPE")
)
```

exclude

exclude(left: set, right: set)

'exclude' joins return a set that contains all nodes from the left set that do not appear in the right set. In set theory, this type of join is referred to as a relative complement join.

{1, 2, 3} complemented with {2, 3, 4} results in {1}

This sample returns a set of all nodes with a value of “int” that are not “TYPE” nodes:

```
Exclude(
find(“int”),
```

```
with("TYPE")
)
```

andat

andat(count: number, left: set, right: set)
andat(value: string, left: set, right: set)
andat(count: number, value: string, left: set, right: set)

The andat function performs both a non-destructive tree traversal and an intersect join in one operation. Each node in the left set is traversed according to parameters provided, then the result of the traversal is intersected with the right set. If the intersect passes, the original node is added to the result set. If the intersect fails, the node is excluded from the result set.

The traversal parameters for andat are the same as for 'ancestor' and 'filter'. For more information about the traversal parameters, see the 'ancestor' function described in the *Set Traversal* Help topic.

This sample takes all "NAME" nodes, traverses them up one parent, and intersects them with a set of all "CLASS" nodes. If a "NAME" node passes both the traversal and intersect join, it is added to the result set. The result is a set of all "NAME" nodes whose immediate parent is a "CLASS" node.

andat(1,
type(“NAME”),
type(“CLASS”)
)

—

Sparx Intel Service

The Sparx Intel service program provides a means for development projects and players to gain valuable insight into the code bases and software frameworks they are working with. The service acts as a provider to Enterprise Architect clients, allowing access to Intelli-sense in code editing and insightful search results in search tools.

The Sparx Intel service is part of the Sparx Satellite Services umbrella. The service can run on a local network or Cloud running Microsoft Windows. The Sparx Intel Satellite service can be installed as a Windows service or run as a standalone process. The service allows multiple Enterprise Architect clients to access and query the same information from many different software domains and frameworks.

This feature is available from Enterprise Architect Release 16.0

Sparx Intel Service Configuration

The program SparxIntelService.exe runs one or more intel services for Enterprise Architect. The program is located in the same install folder as Enterprise Architect, and it uses a configuration file that names the services that can run on the local machine.

In the examples in this topic, the program will attempt to use the file c:\mystuff\myservices.config. It will look for a service named *EA* and, if found, start it.

```
SparxIntelService.exe listen service=EA
config=c:\mystuff\myservices.config
```

The Config File Format

The configuration file has this format:

```
# comment
# comment
# comment
{       # start of service definition
  ...   # list of directives as pairs
}       # end of service definition
{       # start of service definition
  ...   # list of directives as pairs
}       # end of service definition
```

Comments are indicated by the # character.

If the config directive is omitted (not recommended), the program will look for a config file of the same name as the program, in the same directory as the program.

In this example the program will attempt to use the file SparxIntelService.config in the same folder:

SparxIntelService.exe listen service:EA

Directive	Description
name	When a service is named on the command line, the service with the matching name attribute will be started.
status	When status = ON, the service will be started; otherwise, it will not be started.
lazyload	When lazyload is 'true', any Code Miner database will be delay loaded until an Intel request is made to the service.
loglevel	Defines the level of information logged, as a combination of keywords { information, warning, error} separated by a '|'. For example: loglevel= Information|warning|error
logoutput	Specifies the full pathname of the log file

	to write to. For example: logoutput=c:\logfiles\intel-service-project 1.log
database	Specifies the full path name of the Code Miner database to be loaded. For example: database=c:\intel--service\project1.cdb Multiple 'database' directives are allowed, each specifying a different database.
allow	Identifies the IP address that is permitted to connect to the service on the Port. For example: allow=localhost allow=127.0.0.1 allow=172.160.* (wildcards are allowed when the 'network' directive has a value of 'network' or 'public', but not 'local')
network	Allows service connections to be restricted. • local - the service will not listen on any connection other than localhost • network - when used with wildcard

	'allow' directives, allows clients on an allowed IP address wild card to connect • pubic - allows any connection
show	When 'true', the Console window for the service will be shown; the default is 'false'.
port	The Port on which the service will listen.

The Service Configuration Template

When choosing the 'Execute > Tools > Services > Code Miner Service > Edit Configuration File' ribbon option you display the Windows 'Save As' browser through which you can choose either the config file to open or where a file should be created.

If no config file is recorded in the registry and you specify a non-existent filename, that file is created, filled with a 'bare bones' configuration skeleton and saved. The selected/new configuration is then shown in the Enterprise Architect default editor.

The 'bare bones' template is shown here.

```
#-----------------------------------------------------------------
---------
```

```
# Sparx Intel Service Configuration File
#
-------------------------------------------------------------------------
--------
# This file is used to describe one or more intel services and
the code miner databases that they support
# This file can be used in EA to manage a number of
services on the local machine
#
-------------------------------------------------------------------------
--------
# Service Attributes
#
-------------------------------------------------------------------------
--------
# name              The unique name of the service in this file
# status            "ON" - service can run, "OFF" service
will never run
# lazyload                "true" - databases are loaded n
demand, "false" - databases are loaded when service starts
# port                    Unique port number that service
will listen on and EA will connect to
# network                 [optional,default=local] Restricts
service to listening to locahost only (local), to a range of
addresses (network) or any address (public)
# allow             Allows a specific IP address or wildcard
IP address to connect (if network is NOT local)
```

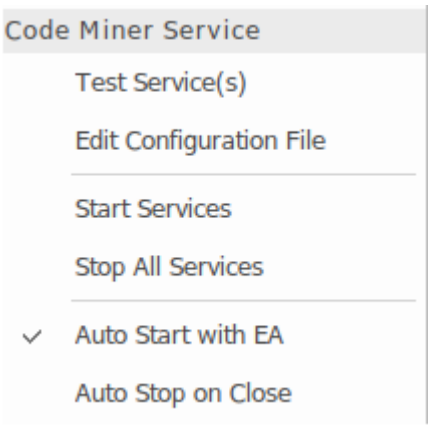
```
#                   (There can be multiple allow directives
present)
# autoupdate        "true" - will detect updates to listed
databases and reload them, "false" default, changes are not
detected
# show              [optional,default=false] shows the console
window for the service
# logoutput         [optional] The path of a log file
which service can write to
# loglevel          [optional] The levels of information
logged. Combine with "|" character eg: {
information|warning|error }
# database          [Required] The full path to a
codeminer database which usually has the .cdb file
extension
#                   (There can be multiple database directives
present)
#
#
-------------------------------------------------------------
--------
# Attribute Values
#
-------------------------------------------------------------
--------
#
# <string>   - text. (do not include quotes)
```

```
# <boolean>  - text, { true, false, ON, OFF }
# <path>     - fully specified file path to codeminer database
# <number>   - digits
#
-------------------------------------------------------------------
--------
#
{
	name=<string>,
	status=<boolean>,
	lazyload=<boolean>,
	port=<number>,
	allow=<string>,
	allow=<string>,
	network=<string>,
	autoupdate=<string>,
	show=<boolean>,
	logoutput=<string>,
	loglevel=<string>,
	database=<path>,
	database=<path>,
	database=<path>
},
{
	name=Project1,
```

```
        status=ON,
        lazyload=TRUE,
        allow=localhost,
        allow=127.0.0.1,
        port=9999,
        autoupdate=true,
        database=c:\Project1\Project1.cdb
}
```

The Sparx Intel Service Ribbon Options

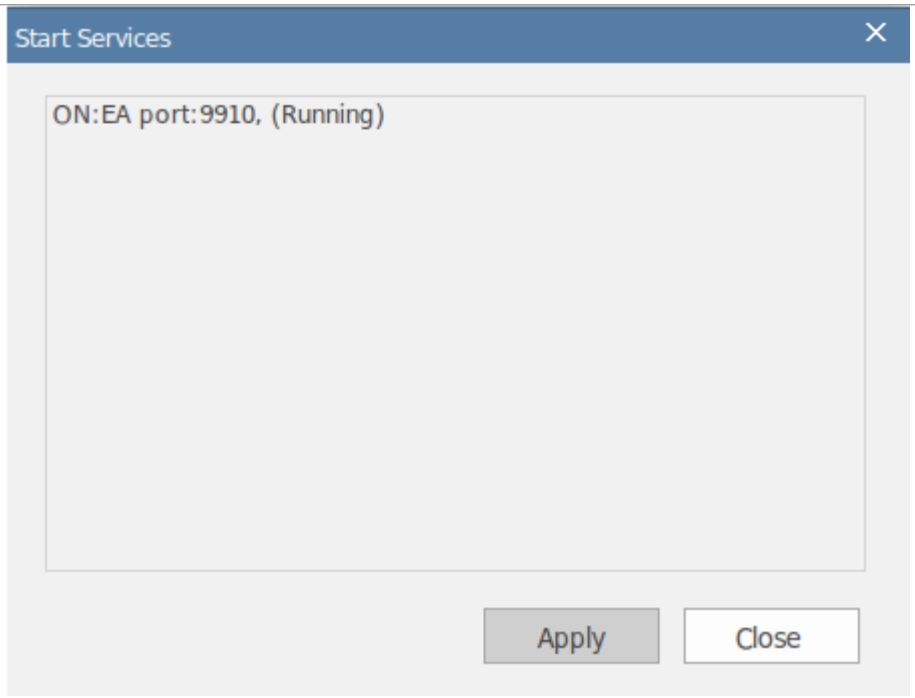
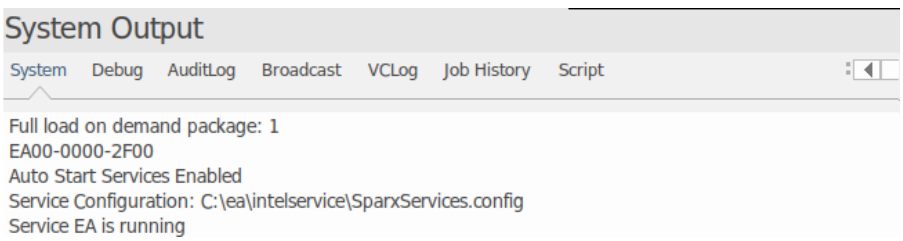
When a Service Configuration file exists, you can edit it or execute it using a number of options available from the 'Execute > Tools > Services' ribbon option within the Code Miner menu option group.



Test Service(s)	This option displays a dialog that lists the status of each Enterprise Architect service named in the current Configuration file, and its state.

Edit Configuration File	This option prompts for the Service Configuration file to use, then opens that file in an Enterprise Architect text editor. The system remembers where the file is held.
Start Services	This option reads the current Service Configuration file and starts services that are configured to run, and stops running services that are not configured to run. A service is configured if: 1. It is named in the config file. 2. It has the attribute status:ON.

```
# <number>   - digits
# ------------------------------------------------------------

{
    name=project1,
    status=ON,
    lazyload=true,
    port=9910,
    allow=localhost,
    network=local,
    autoupdate=true,
    show=true,
    logoutput=c:\My Documents\project1.txt,
    loglevel=information|warning|error,
    database=c:\My Documents\project1\project1.cdb
}

```

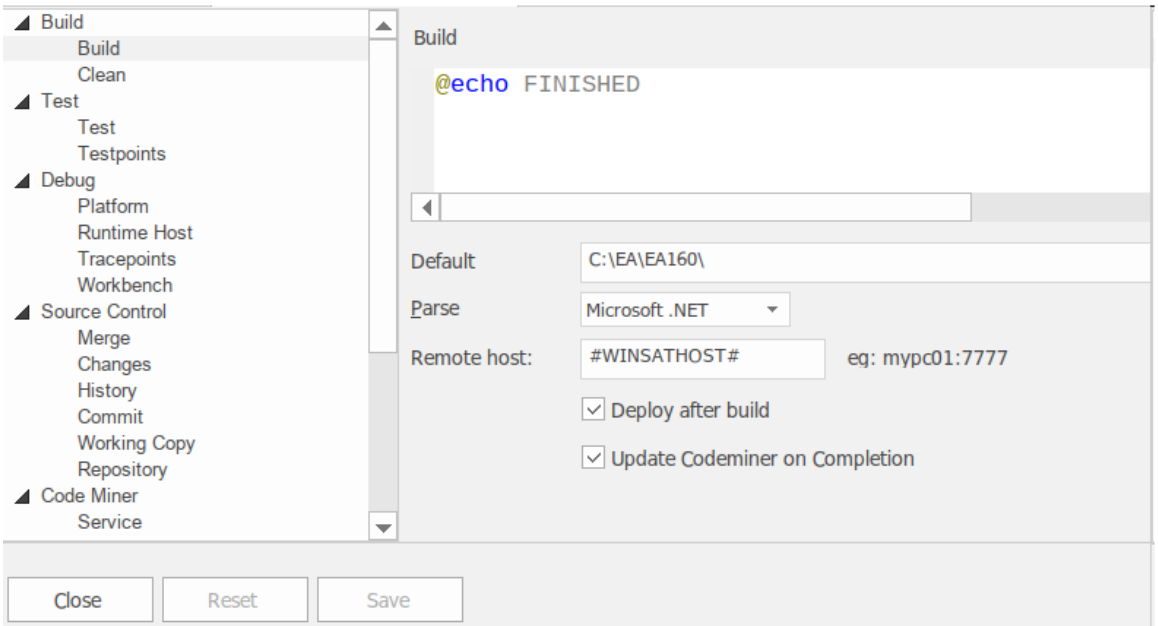
	
Stop All Services	This option stops any services that are currently running.
Auto Start with EA	This option automatically starts services having the 'status:ON' attribute when the model opens.  The messages logged to the System Output window here when the model is opened indicate that the service was already running.
Auto Stop on	This option automatically stops running

Close	services when Enterprise Architect is closed down.

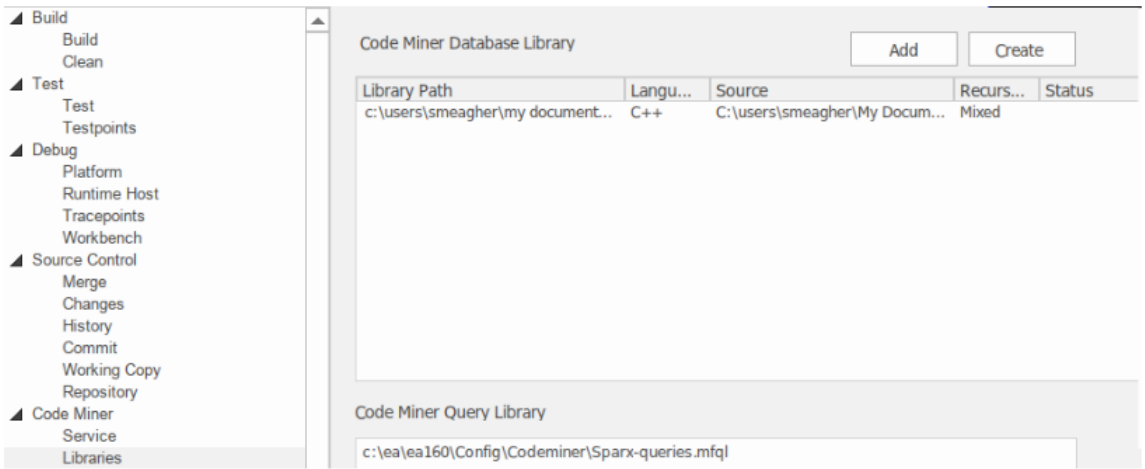
Sparx Intel Service Automatic Update

When you execute the Build command for an Analyzer Script , a job is added to the Job Queue.

If the Build script has the 'Update Codeminer on Completion' checkbox ticked in the Analyzer Script Editor, an additional task is added to the job to update each of the Codeminer databases listed in the script.



The libraries can be seen in the Code Miner | Libraries section of the script.



How the Task Runs

The Code Miner update task runs the program SSCodeMiner.exe with two arguments.

The first argument specifies the database to perform the incremental build on and has this form:

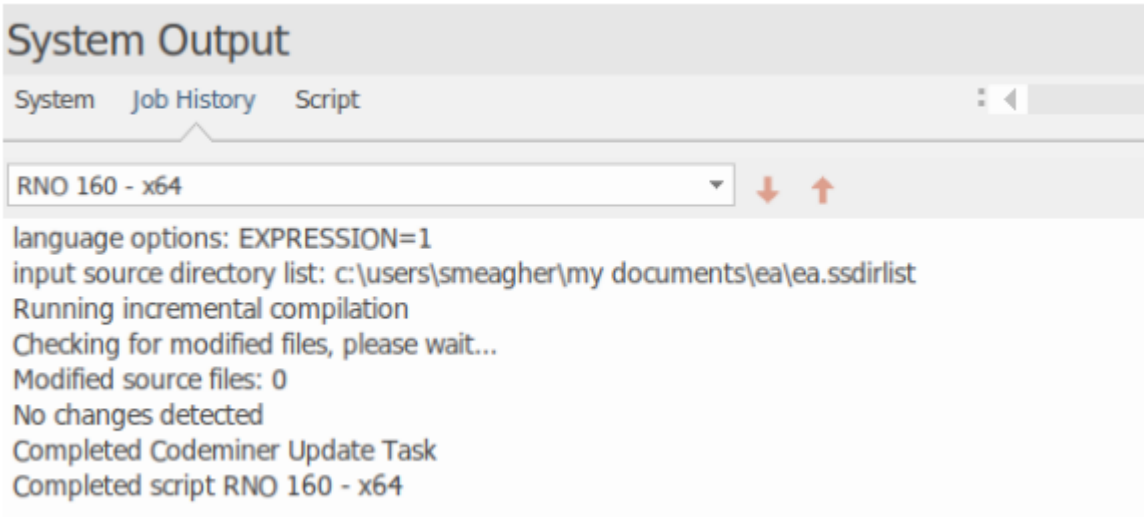
```
update="c:\path\ea.cdb"
```

The second argument is optional and specifies an auxiliary macro grammar file to use when compiling the database; it has this form:

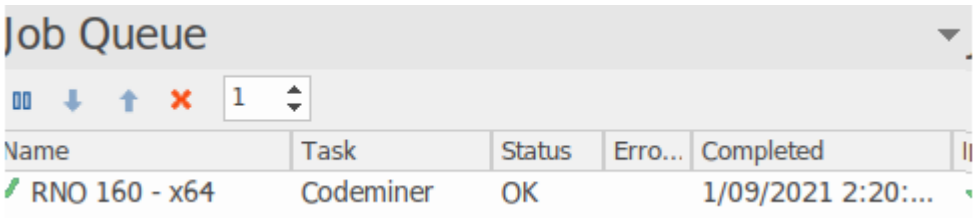
```
macros="c:\ea\ea160\config\CodeMiner\SparxProjectMacros.nbnf"
```

Job Output

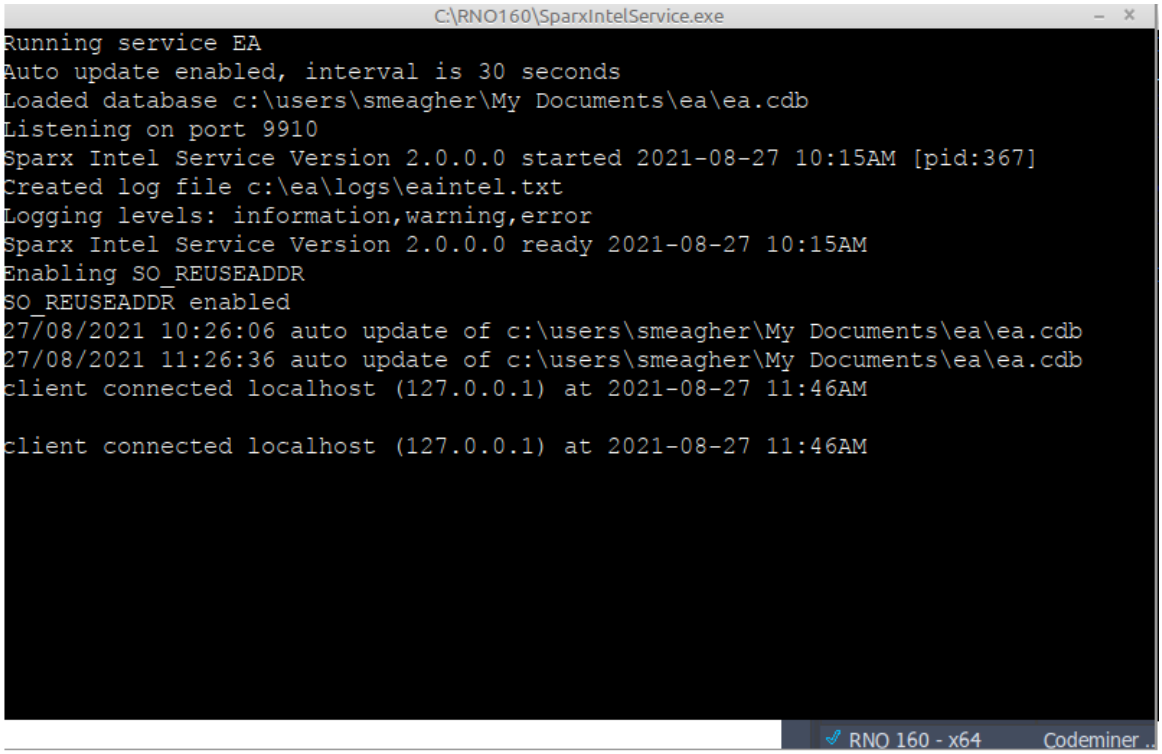
As the Code Miner update task runs, output from the captured SSCodeMiner.exe update process is sent to the 'Job History' tab of the System Output window, in the same form as is displayed when performing a manual update of a Code Miner database in Enterprise Architect. In this illustration we can see that the Analyzer Script RNO 160 -x64 has completed successfully.



The Job Queue window shows that the job has completed. The last task to run was the Code Miner update.



The 'Job History' tab showed that no source code files had changed. If modified source code changes are detected - that is, the Code Miner service has detected a new build of ea.cdb and automatically updated it - this information is displayed:


C:\RNO160\SparxIntelService.exe
Running service EA
Auto update enabled, interval is 30 seconds
Loaded database c:\users\smeagher\My Documents\ea\ea.cdb
Listening on port 9910
Sparx Intel Service Version 2.0.0.0 started 2021-08-27 10:15AM [pid:367]
Created log file c:\ea\logs\eaintel.txt
Logging levels: information,warning,error
Sparx Intel Service Version 2.0.0.0 ready 2021-08-27 10:15AM
Enabling SO_REUSEADDR
SO_REUSEADDR enabled
27/08/2021 10:26:06 auto update of c:\users\smeagher\My Documents\ea\ea.cdb
27/08/2021 11:26:36 auto update of c:\users\smeagher\My Documents\ea\ea.cdb
client connected localhost (127.0.0.1) at 2021-08-27 11:46AM
client connected localhost (127.0.0.1) at 2021-08-27 11:46AM
RNO 160 - x64
Codeminer

Service Configuration

Service program

The name of the service program is SparxintelService.exe.

Configuration File

The service is configured by the file SparxIntelService.config.

The file must be located in the same directory as the service program.

The file contains a number of directives and also lists the Code Miner databases to be served.

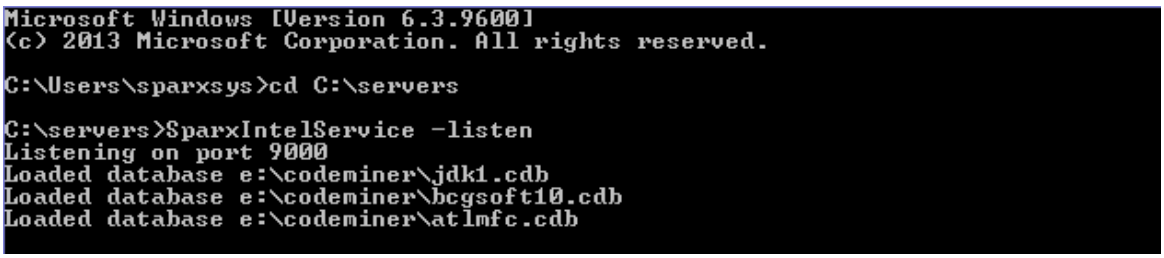
The file is read once when the service is started.

Directives	Description
port	The Port number on which the service will listen.
allow	Names a domain or IP address that is allowed access: 198.* or 127.0.0.1
network	Values can be "public", "network" or "private".

	• Use "private" when allow directives specify one or more single IP addresses • Use "network" when allow directives specify a wildcard domain: 198* • Use "public" to allow all clients
database	Names the full physical file path of a Code Miner database on the server.

Running the program standalone

From a normal console enter the command:
SparxIntelService -listen

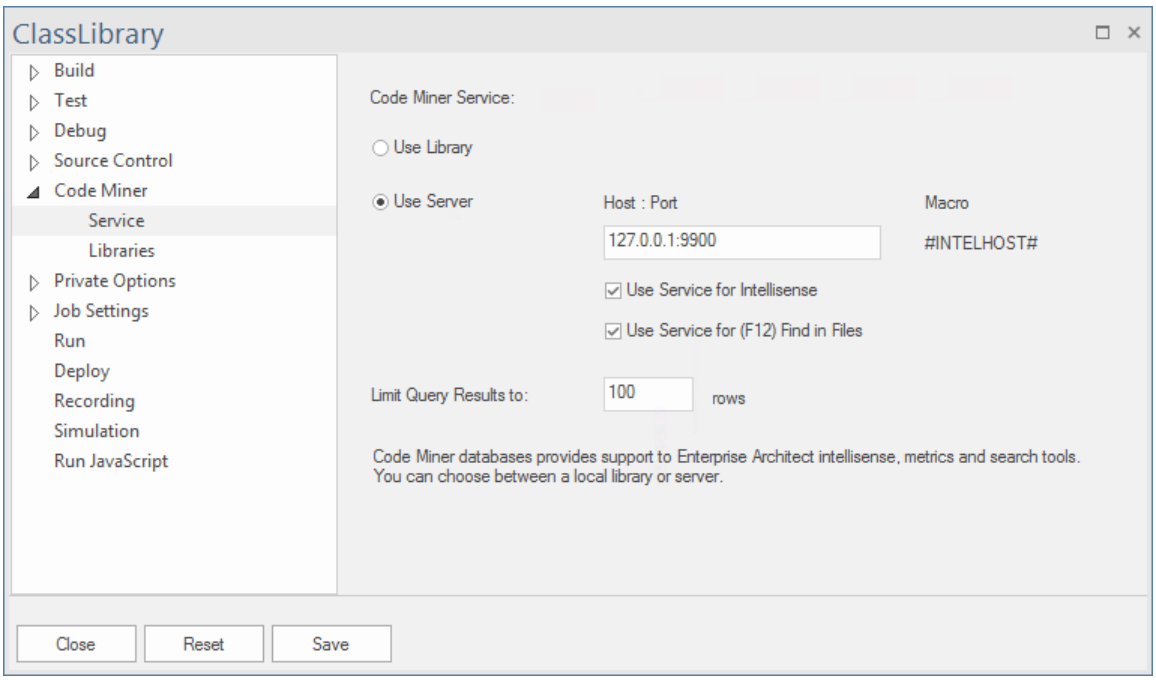


Installing as Windows Service

From an Administrative console enter the command:
SparxIntelService -install

Client Configuration - Configuring Enterprise Architect to Use a Code Miner Service

Enterprise Architect uses components known as Analyzer Scripts for the configuration of many support systems. This is where the location of the server is specified. This image shows the 'Code Miner Service' page of a script.



Access

Ribbon	Develop > Source Code > Execution Analyzer > Edit Analyzer Scripts > Double-click on a Script > Code Miner >

	Service

Configuration Fields

Use Server	Select this radio button to set up the Code Miner server to use.
Host : Port	Type in the number of the Port through which the Service will operate.
Use Service for Intellisense	Select the checkbox to use the Intel Service for Intelli-sense field completion.
Use Service for [F12] Find in Files	Select the checkbox if you want to use the Service instead of the Find In Files window to run search queries, when you press F12.
Limit Query Results to rows	Type in the number of rows of query results to display per page.
Save	Click on this button to save the

	configuration details you have entered.

Geospatial Models

The popularity of the internet, the ever-present mobile phone and the prevalence of location-based services have resulted in almost everyone interacting with location-based information in some form in their daily lives. It has also become critical for governments and organizations to embrace this type of information as part of strategic decision making. Geospatial information can be modeled in Enterprise Architect and also integrated with other data to form a single and comprehensive view of information not possible in other tools.

Enterprise Architect, through the use of MDG Technologies, supports the Geography Markup Language (GML) application schemas and the modeling of ArcGIS geodatabases. The information precursors to these models - such as community conceptual models - can also be modeled, and traceability can be used to connect the models together.

Modeling Tools

Tool	Description
ArcGIS Profile 	Enterprise Architect supports the design of geodatabases for the ArcGIS 10.0 suite of tools developed by Esri Inc.

Geography Markup Language 	Geography Markup Language (GML) in Enterprise Architect is the implementation of the Open Geospatial Consortium's Geography Markup Language 3.3, which provides an XML grammar for geographical feature modeling capabilities within Enterprise Architect.

Getting Started

Enterprise Architect partitions the tools extensive features into perspectives this ensures that you can focus on a specific task and work with the tools you need without the distraction of other features. To work with the ArcGIS Geodatabases or Geography Markup Language features you first need to select the following perspectives:

- Database Engineering > ArcGIS
- Information Exchange > Geographic GML

Setting the perspective ensures that the ArcGIS Geodatabases and Geography Markup Language diagrams and their tool boxes and other features of the perspective will be available by default.

ArcGIS Geodatabases

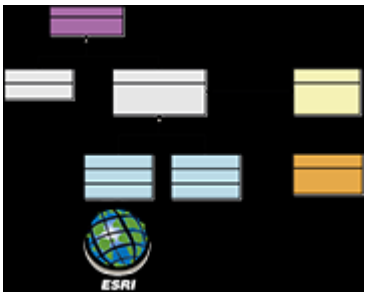
Using the ArcGIS features in Enterprise Architect you can visualize geodatabases with ease. This allows you to unify teams working in traditional software centric and engineering systems with your geospatial teams defining features and domains. Teams defining the strategy business rules and requirements for a system or the components that deliver the system functionality can share models with the

geospatial teams creating an integrated model that will help with integration and risk reduction.

Geography Markup Language (GML)

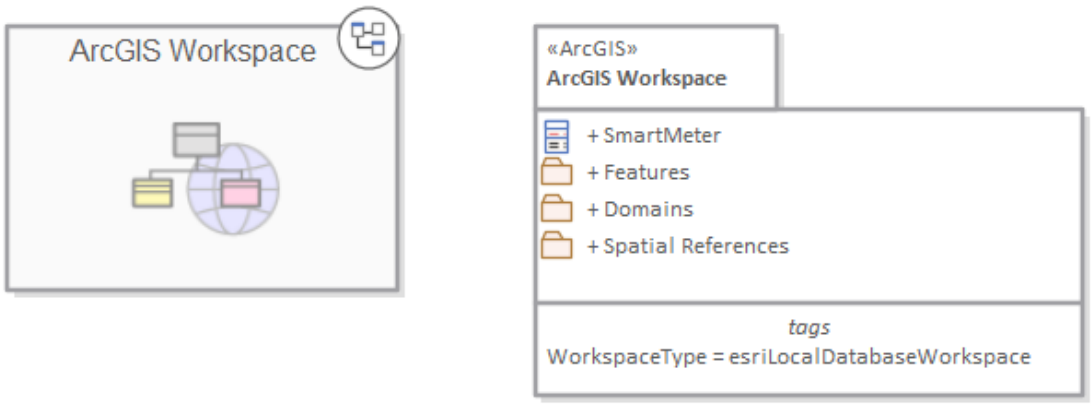
Using the Geography Markup Language (GML) facility you can model organization or community based application schemas. The models can be used to show the relationship between Features and these elements in turn can contain any number of Properties that qualify their characteristics. These can be based on defined Types, DataTypes, CodeLIsts or Enumerations.

ArcGIS Geodatabases



Exchange, Model and Visualize ArcGIS Geodatabases

Enterprise Architect supports the import and export of ArcGIS geodatabases, allowing you to visualize Features and Domains within this multi-featured collaboration platform. In the recent past there has been a separation of the disciplines between system software development and geospatial development. In this age of social architecture and digital disruption almost every project and endeavor requires some aspect of location information, from simple delivery services to agricultural, mining, exploration, weather, real estate and disaster recovery systems.



Package diagram showing a Navigation Cell and a Package containing Features Domains and a Geospatial Reference

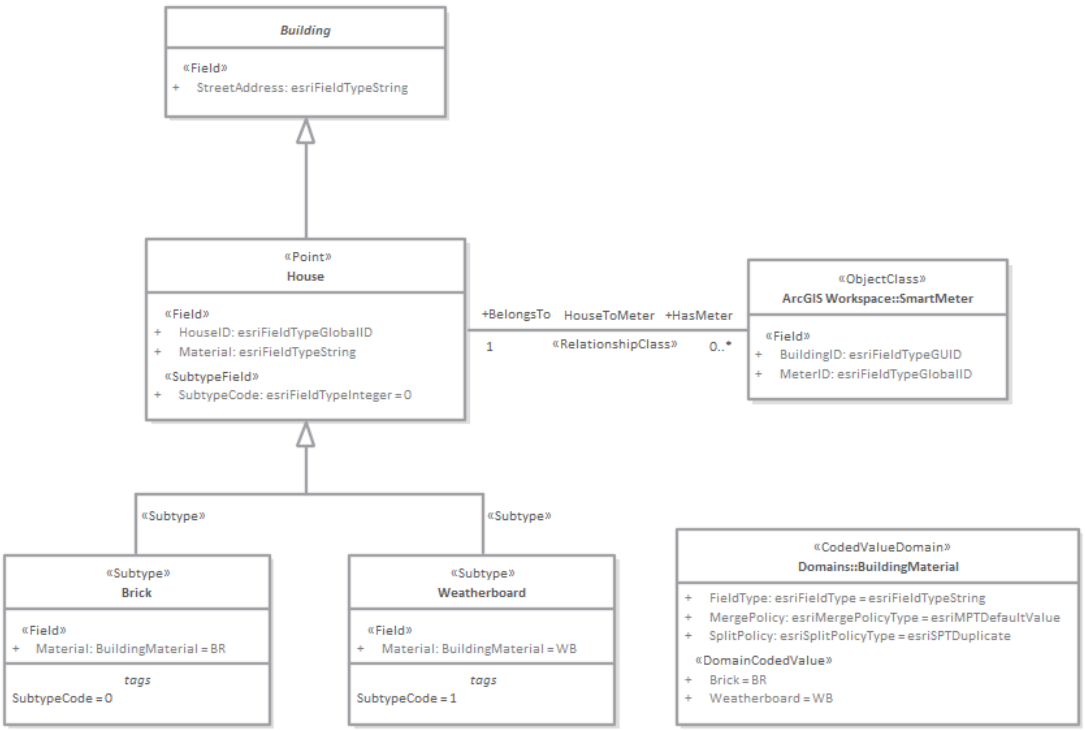
The ArcGIS system, developed by Esri, supports the development and management of geodatabases. As it is for other databases, it is useful to model the design of a geodatabase using a standard notation such as UML. You can perform such modeling in Enterprise Architect, using the UML profile for ArcGIS. Once you have modeled an ArcGIS schema in Enterprise Architect, you can export the model to ArcGIS as an XML Workspace document. You can also visualize an existing ArcGIS geodatabase schema, by importing the ArcGIS XML Workspace document into Enterprise Architect.

Example Diagram

ArcGIS diagrams allow you to visualize the geographic features, domains and other elements that make up a geodatabase schema. In this example a Building has been sub-typed as a house, the house in turn is sub-typed based on the material type. The subtypes of the House references a Coded Value Domain also presented in the diagram with two Domain Code Values:

- Brick
- WeatherBoard

A Smart Meter is associated with the house. The House is a type of Building and the Building contains the property of Street Address


Building
«Field»
+ StreetAddress: esriFieldTypeString
«Point»
House
«Field»
+ HouseID: esriFieldTypeGlobalID
+ Material: esriFieldTypeString
«SubtypeField»
+ SubtypeCode: esriFieldTypeInteger = 0
+BelongsTo
1
HouseToMeter
«RelationshipClass»
+HasMeter
0..*
«ObjectClass»
ArcGIS Workspace::SmartMeter
«Field»
+ BuildingID: esriFieldTypeGUID
+ MeterID: esriFieldTypeGlobalID
«Subtype»
«Subtype»
«Subtype»
Brick
«Field»
+ Material: BuildingMaterial = BR
tags
SubtypeCode = 0
«Subtype»
Weatherboard
«Field»
+ Material: BuildingMaterial = WB
tags
SubtypeCode = 1
«CodedValueDomain»
Domains::BuildingMaterial
+ FieldType: esriFieldType = esriFieldTypeString
+ MergePolicy: esriMergePolicyType = esriMPTDefaultValue
+ SplitPolicy: esriSplitPolicyType = esriSPTDuplicate
«DomainCodedValue»
+ Brick = BR
+ Weatherboard = WB

Exporting ArcGIS XML Workspaces

When you have modeled your Geodatabase Workspace XML Document (containing the ArcGIS schema), you can export it to an external directory (using the Publish Model Package facility), from which you can then import it to the Esri ArcCatalog.

Access

Click on an ArcGIS stereotyped Package (your ArcGIS Workspace Package) in the Browser window.

Ribbon	Specialize > Technologies > ArcGIS > Export to ArcGIS Workspace XML or Publish > Model Exchange > Publish As...
Context Menu	Right-click on Package | Specialize | ArcGIS | Export to ArcGIS Workspace XML
Keyboard Shortcuts	Ctrl+Alt+E : Publish

Export the Workspace

Option	Action
Root Package	Display the name of the selected ArcGIS Workspace Package.
Filename	Type in or browse for the file path into which the XML file is to be generated.
XML Type	Select 'ArcGIS' as the XML/XMI version to export the Package to.
Format XML Output	Format the output into readable XML (this takes a few more seconds at the end of the run).
Write Log File	Write a log of the export activity (recommended). The log file is saved to the directory into which the XML file is exported.
View XML	Click on this button to view the exported XML file.
Export	Click on this button to initiate the XML export.

Close	Click on this button to close this dialog.
Progress	Observe the progress of the XML export.

Notes

- ArcGIS is available in the Professional, Corporate, Unified and Ultimate Editions of Enterprise Architect
- In the Corporate, Unified and Ultimate Editions of Enterprise Architect, if security is enabled you must have 'Export XMI' permission to export to XML
- Before exporting your model to an ArcGIS schema, you must define at least one Spatial Reference element; Spatial Reference elements are referred to by other schema elements via a dynamically linked Tagged Value, named SpatialReference
- The DefaultSpatialReference tag on an ArcGIS Package is used to specify a Spatial Reference that can be applied to all Feature Datasets and Feature Classes in the workspace; therefore, you do not need to apply a Spatial Reference element to each Feature Dataset or Feature Class
- If you do not reference a Spatial Reference Class from any Feature Dataset or Feature Class in your ArcGIS model, Enterprise Architect by default will generate an XML schema with Unknown type of Spatial Reference for these elements

Importing ArcGIS XML Workspaces

If you have a Geodatabase Workspace XML Document (containing the ArcGIS schema) you can import it into your Enterprise Architect project as a UML model.

Before running the import, deselect the 'Sort Features Alphabetically' checkbox on the 'Objects' page of the Preferences window (Start > Appearance > Preferences > Preferences). This ensures that the fields are imported and organized in Enterprise Architect in the same order as in the source.

Access

Click on the target Package in the Browser window.

Ribbon	Publish > Technologies > Publish > ArcGIS > Import ArcGIS Workspace XML or Publish > Model Exchange > Import Package > Import Package from Native/XMI File : Other XML Formats > ArcGIS
Context Menu	Right-click on Package | Specialize | ArcGIS | Import ArcGIS Workspace XML

Keyboard Shortcuts	Ctrl+Alt+I : Other XML Formats | ArcGIS

Import a Geodatabase Workspace XML document

Option	Action
Filename	Type in or browse for the name of the ArcGIS XML file to import.
Create Diagrams	Select the checkbox to create Class diagrams under the imported Packages.
Hide System-Level ArcGIS Fields on Diagrams	Select the checkbox to hide these stereotyped attributes: • RequiredField • AttributeIndex • SpatialIndex on these stereotyped Classes: • Point • Polyline • Polygon

	• MultiPatch The 'RequiredField' and 'AttributeIndex' attributes are also hidden for the Table (Object Class) Class. This option is enabled only when the 'Create Diagrams' checkbox is selected.
Strip GUIDs	The 'Strip GUIDs' feature is currently mandatory for ArcGIS imports, which means that elements are created 'as new' each time an ArcGIS schema is imported.
Write Log File	Select the checkbox to write a log of import activity (recommended). The log file is saved in the directory from which the file is being imported, with the same name as the imported file plus the suffix _import.log.
View XML	Click on this button to view the XML before import.
Import	Click on this button to import the ArcGIS XML file.
Close	Click on this button to close this dialog.
Help	Click on this button to display this Help

	page.
Import Progress	This field indicates the progress of the import.

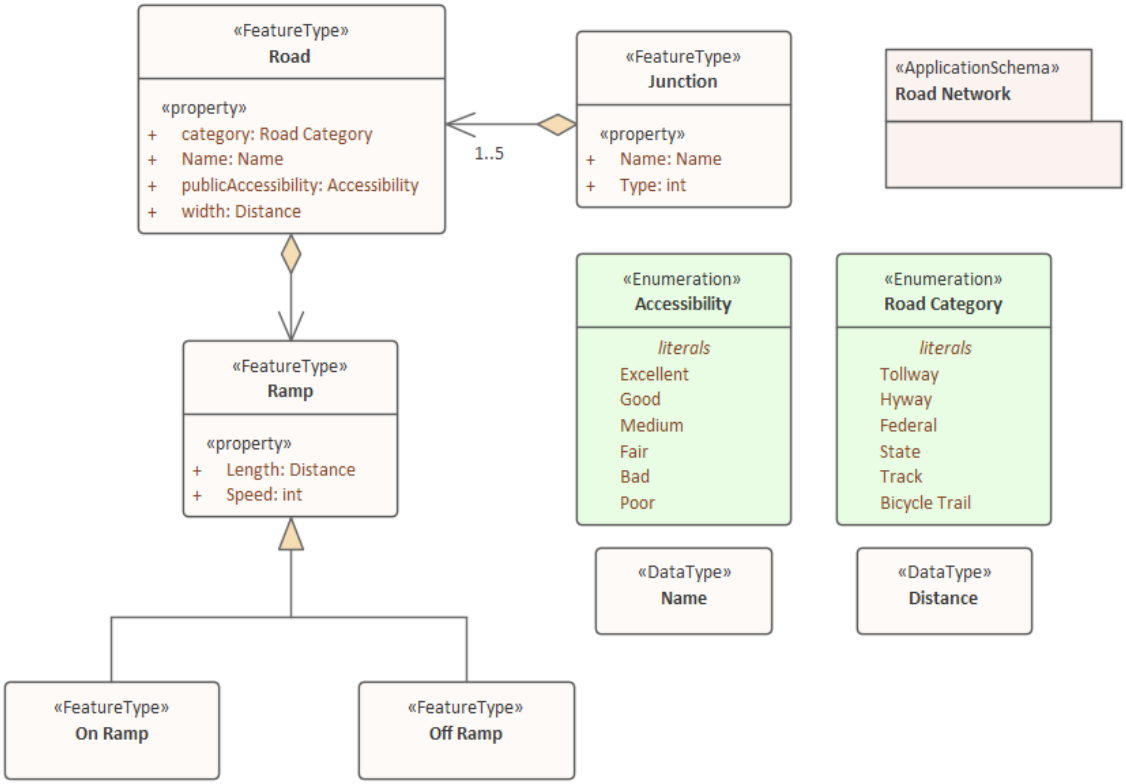
Notes

- ArcGIS is available in the Professional, Corporate, Unified and Ultimate Editions of Enterprise Architect

Geography Markup Language (GML)

Model Geographic Features and Generate Application Schemas

You can create expressive and collaborative models of the important features in your domain and use these to generate Geography Markup Language (GML) compliant application schemas that can be consumed by other applications. Many of the disruptive forces and technologies that have changed the way we interact with each other and the world we inhabit, involve geographic locations and features. We drive along roads and stop at lookouts to view coastal features or cityscapes, we travel abroad to view monuments and buildings such as churches and museums, we rely on wind farms for energy and we take off and land at airports to name a few. You can model any geographic features of interest using Enterprise Architect's implementation of the Geography Markup Language which is fundamental for geographic information systems as well as its use as an open interchange format for geographic transactions on the Internet.



GML Model of roads showing two Features with properties that access two Data Types and Enumerations

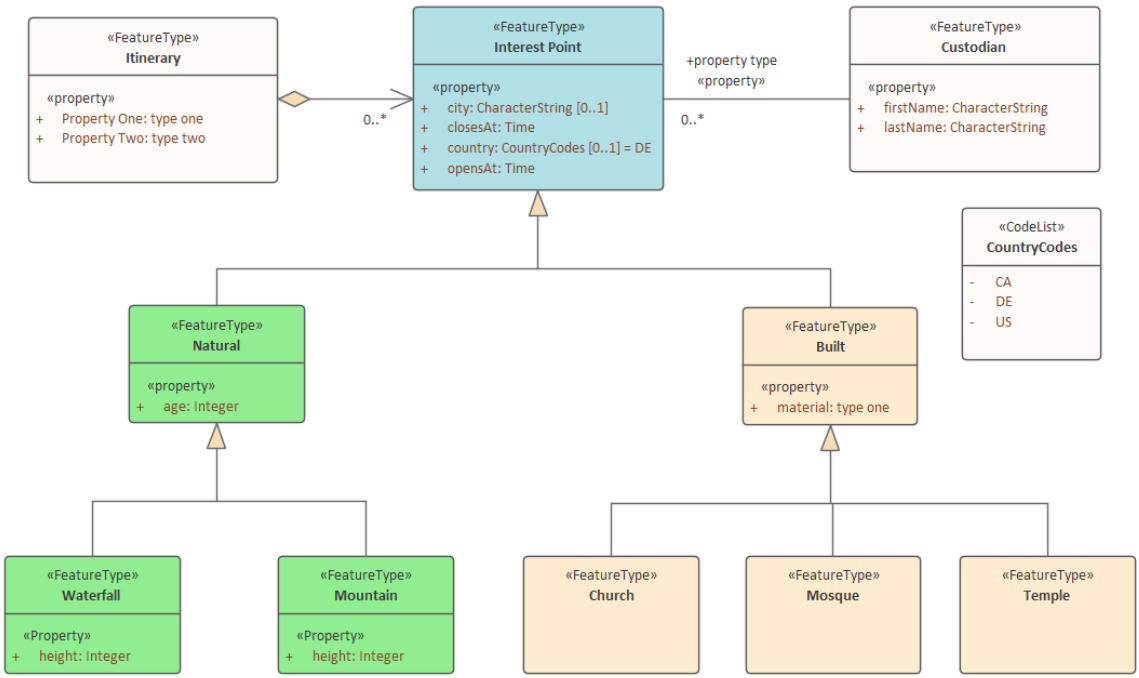
GML for Enterprise Architect is an implementation of the Open Geospatial Consortium's Geography Markup Language (GML) 3.3 , which provides an XML grammar for geographical feature modeling capabilities within Enterprise Architect at or later than Release 10.

Through GML, you can:

- Apply a UML Profile for the Geography Markup Language (GML) 3.3
- Make use of customized diagram types and toolbox pages, for convenient access to elements and relationships to model geographical features effectively
- Generate GML Application Schema files

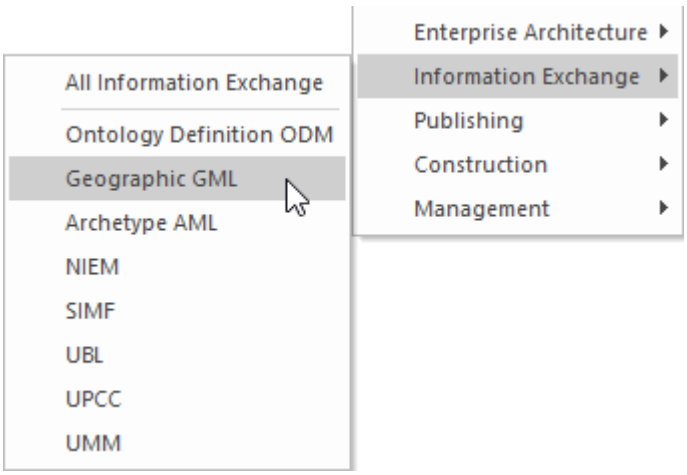
Example Diagram

Using the Geography Markup Language (GML) facility you can model organization or community based application schemas. The models can be used to show the relationship between Features Types that contain any number of Properties that qualify their characteristics. These can be based on defined Types, Data Types, Code Lists or Enumerations. You can collaborate with other geospatial colleagues or with people working in traditional systems implementations in disciplines that manage artifacts including: Strategies, Goals, Requirements, Data Models, Software Models, Deployment Descriptions and more.



Modeling with GML

You can create GML models using the comprehensive diagramming and modeling facilities in Enterprise Architect. First you need to select the GML or Information Exchange Perspective. Perspectives are a useful focusing tool facility that ensure you remain focused and can concentrate on GML modeling.



Perspective menu - GML Perspective Selection

This activates the UML Profile for GML, allowing you to create models with elements and connectors that describe your organization or community domains.

Access

Ribbon	Specialize > Technologies > GML

Context Menu	Right-click on Package | Specialize | GML

Features

Feature	Detail
Profile Support	You can develop GML constructs quickly and simply, through use of the built-in GML facilities provided in the form of: • A GML diagram type, accessed through the 'New Diagram' dialog • GML pages in the Diagram Toolbox that map GML concepts to appropriately stereotyped UML elements • GML element and relationship entries in the 'Toolbox Shortcut Menu' and 'Quick Linker'
GML Toolbox Page	The GML Toolbox pages contain elements and connectors to model geographical features effectively.
UML Classes	(Optional) You can download the UML Classes implemented in ISO/TC 211 as

from ISO	an XMI file, then import the XMI file into Enterprise Architect as a Package containing diagrams and standard UML Classes, which you can reuse in your model. • Not all UML Classes implemented in ISO/TC 211 have a corresponding mapping in GML; the Classes that have a mapping (as specified in the GML 3.2.1 specification) are specified in the configurable file GMLClassMapping.xml in the 'Sparx Systems > EA > Config > GML' folder • The Namespace information for these Classes is specified in the configurable file GMLNamespaces.xml in the 'Sparx Systems > EA > Config > GML folder'
GML Application Schema Generation	Any model you create using GML in Enterprise Architect can be exported as a GML Application Schema. Using the configurable file GMLStereotypes.xml in the 'Sparx Systems > EA > Config > GML' folder, you can specify aliases for the standard GML stereotypes. The GML Application Schema Generator will also consider these aliases during Schema generation.

Notes

- GML is available in the Professional, Corporate, Unified and Ultimate Editions of Enterprise Architect

More Information

Geospatial modeling can be explored in more detail by accessing the following two topics:

ArcGIS Geodatabases

Geography Markup Language

User Interaction & Experience

Visualize User Interfaces for Smart Phones, Tablets and Notebooks Using Wireframes

The Wireframe Toolbox pages provide a wide range of icons that you can use in wireframe modeling to represent the appearance of a device at a particular point in the execution of an application. Devices you can model include:

- Android Phones and Tablets
- Apple iPhones and Tablets
- Windows Phones
- Screen dialogs
- Web pages - to model how the web pages work

Access

On the Diagram Toolbox, click on  to display the 'Find Toolbox Item' dialog and specify 'Wireframing', 'Android', 'Apple', 'Dialog', 'Windows Phone' or 'Webpage'.

Ribbon	Design > Diagram > Toolbox
Keyboard Shortcuts	Ctrl+Shift+3

Notes

- Each of the Wireframing Diagram Toolboxes provides one or more Patterns that you can drag onto a diagram as an illustration of what you might achieve, or to act as the basis of the model you are developing
- The 'Properties' dialog for Wireframe elements automatically opens to either a top-level 'Wireframe' tab on which you can edit the element rendering directly, or a second-level 'Wireframing' Tagged Values tab if you define the element rendering by editing the XML for the properties of that element type
- Some of the elements you create from the 'Wireframe' Toolbox pages are properly rendered after you edit the Tagged Values that define their characteristics
- As you develop your Interface diagrams you can establish the positions and layout of the elements freehand by dragging and 'nudging' the elements, or impose some regularity using the 'Snap To Grid' diagram options

Getting Started

Selecting the Perspective

Enterprise Architect partitions the tool's extensive features into Perspectives, which ensures that you can focus on a specific task and work with the tools you need without the distraction of other features. To work with the User Experience Modeling features you first need to select this Perspective:

 <perspective name> > UX Design> Wireframes

Setting the Perspective ensures that the Wireframe diagrams, their tool boxes and other features of the Perspective will be available by default.

Example Diagram

An example diagram provides a visual introduction to the topic and allows you to see some of the important elements and connectors that are created in specifying or describing the way a wireframe, dialog or web page is defined including: Android, Apple and Windows Phones and Tablets, Dialogs and Web Pages with Buttons, Combo Boxes and Radio Buttons.

Interaction Flow Models IFML

The Interaction Flow Modeling Language (IFML) provides system architects, software engineers, and software developers with tools that support the platform independent description of graphical user interfaces for applications accessed or deployed on such systems as desktop computers, laptop computers, PDAs, mobile phones, and tablets. The language was developed by the Object Management Group; the IFML specification (version 1.0. February 2015) is available from the OMG website.

You can access the OMG's UML Profile for IFML within Enterprise Architect.

IFML in Enterprise Architect

In Enterprise Architect you can model application interaction flows quickly and simply through use of the MDG Technology integrated with the Enterprise Architect installer. The IFML facilities are provided in the form of:

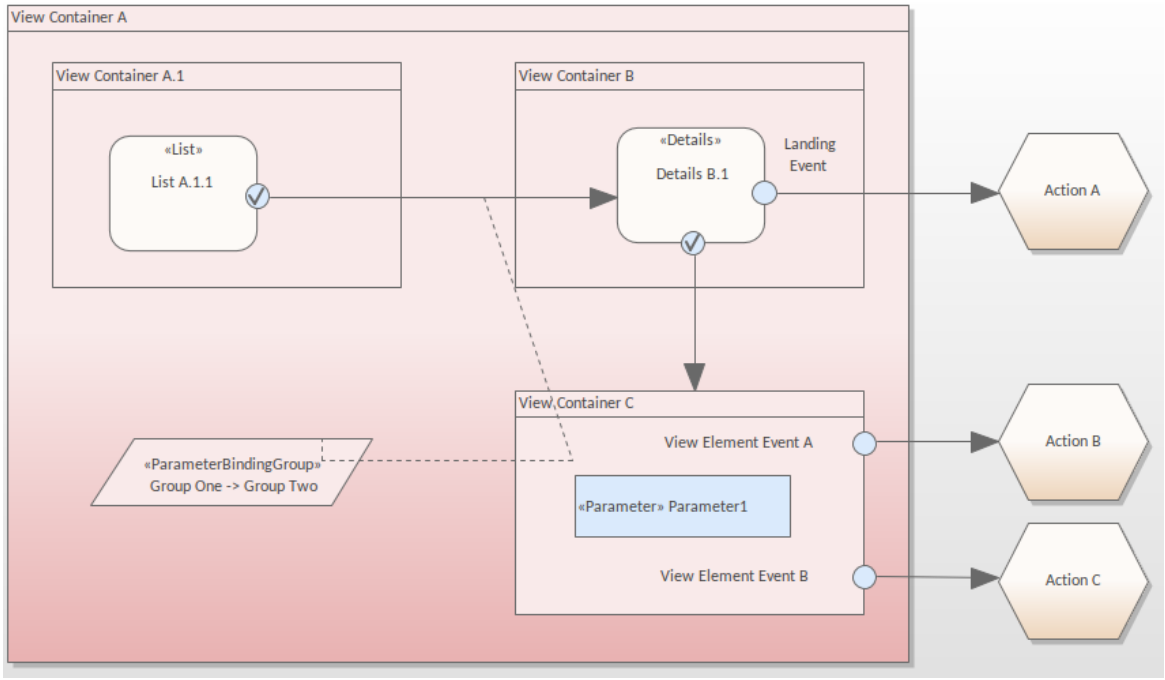
- Eleven IFML model patterns for Information Entry, Interaction and Wireframe, Searches and Desktop Applications, available through the 'Interaction Flow IFML' page of the 'Create from Pattern' tab (the Model Wizard) of the Start Page
- Two IFML diagram types - IFML diagram and IFML Domain Model diagram - accessed through the 'New

Diagram' dialog

- IFML 'Essential Concepts', 'Core' and 'Extensions' pages in the Diagram Toolbox

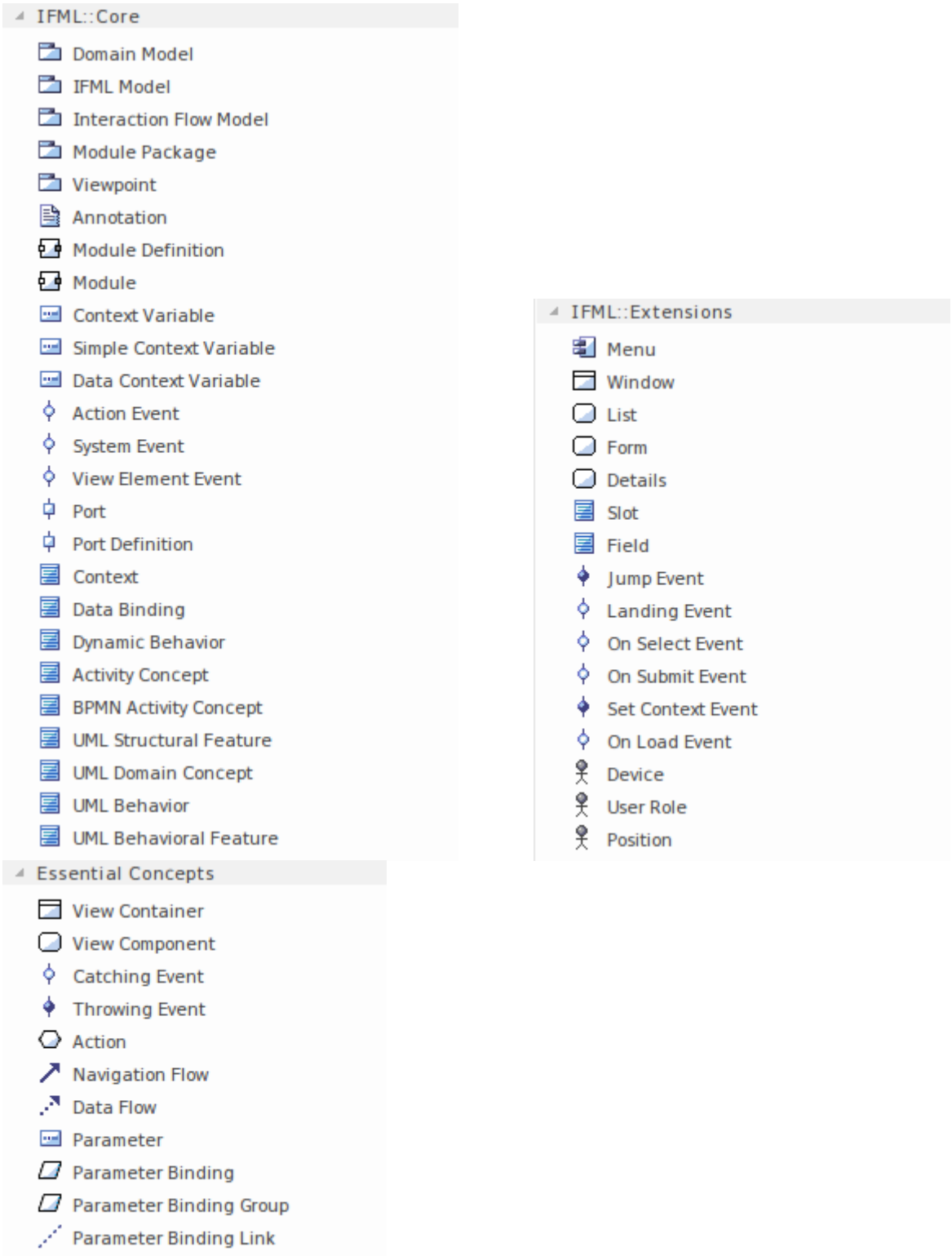
You can, if you wish, make the IFML Technology your default by selecting it in the MDG Technologies window and clicking on the Set Active button.

Example Diagram



IFML Toolbox Pages

The objects defined by the IFML Specification can be created in your model using the icons from these pages of the Diagram Toolbox:

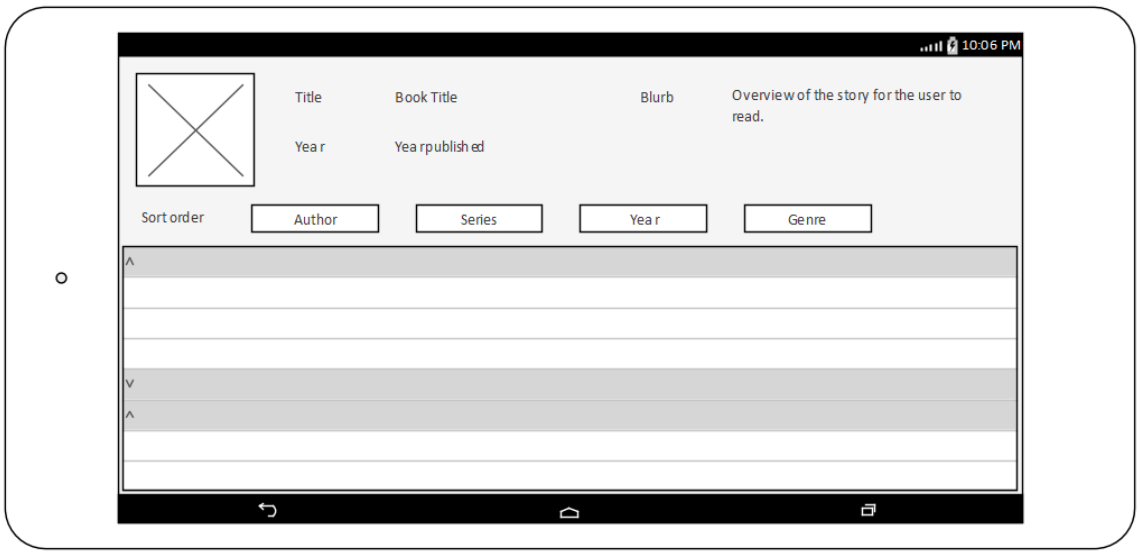


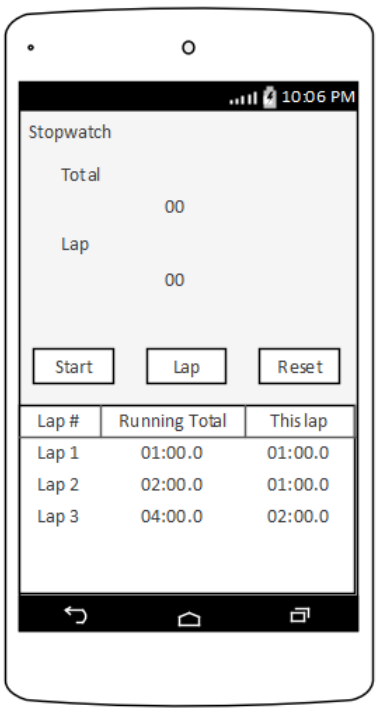
Some objects from the IFML Specification are configured by Tagged Values on the main element type. For example, the View Container element can be set as a Default, Landmark or XOR View Container by setting the

appropriate Tagged Value on the element to 'True'.

Android Wireframe Toolbox

The 'Android Wireframing' pages of the Diagram Toolbox provide the templates for modeling the physical appearance of an Android tablet or phone at a given state of execution of an application. They also provide Patterns for generating a standard model structure for each Android appliance.





Access

On the Diagram Toolbox, click on  to display the 'Find Toolbox Item' dialog and specify 'Wireframing' or 'Android'.

Ribbon	Design > Diagram > Toolbox
Keyboard Shortcuts	Ctrl+Shift+3

Android Screen Types

Item	Description
Android Phone	Generates a frame for the face of the Android Phone you are modeling. A prompt displays for you to specify portrait or landscape orientation. Child controls will be contained within the area of the screen. Tagged Values: • MenuButtons - click on the drop-down button and select to 'Show' or 'Hide' the menu button bar at the bottom of the

	screen • NotificationBar - click on the drop-down button and select to 'Show' or 'Hide' the Notification bar at the top of the screen
Android Tablet	Generates a frame for the Android Tablet you are modeling. A prompt displays for you to specify portrait or landscape orientation. Child controls will be contained within the area of the screen. Tagged Values: • MenuButtons - click on the drop-down button and select to 'Show' or 'Hide' the menu button bar at the bottom of the screen • NotificationBar - click on the drop-down button and select to 'Show' or 'Hide' the Notification bar at the top of the screen
Client Area	Generates a frame element that represents the client area of the device. Tagged Values: • Border Style - click on the drop-down arrow and select to render the border as a solid line or a dashed line, or to have

	no border at all (None) • ScrollbarH - click on the drop-down arrow and select to place a horizontal scrollbar at the top or bottom of the client area, or to have no horizontal scrollbar (<none>) • ScrollbarV - click on the drop-down arrow and select to place a vertical scrollbar on the left or right of the client area, or to have no vertical scrollbar (<none>)

Composite

Item	Description
Expandable List View	Generates an element that represents a two-level grouped list that can be expanded to show one or both levels of item. Double-click on the element to display the 'Properties' dialog at the 'Wireframe' page, which shows the root node (the Expandable List element name), the group heading items (directly under the element name) and the group items

	(under the group headings). • To add a new item click on the level above (the element name or the group heading names) and on the Add button, and in response to the prompt type in the name of the item, which adds the name to the bottom of the section of the list you have created it in To position the item on the list, click on it in the list and click on the  buttons • To remove an item from the list, click on it and click on the Remove button; the item is immediately removed from the list Click on the Item Name (the first-level list item directly underneath the element name). • Name - type in the name for this item group • Expanded - click on the drop-down arrow and select True to show the sub-items for this item, or False to hide the sub-items Click on a sub-item name (the second-level list item) directly under an Item name. Sub-item nodes do not have any child nodes.

	• Name - type the name of the sub-item • Selected - click on the drop-down arrow and select True to highlight the item as selected, or False to omit the highlight
Table	Generates a Table element with labeled columns, rows and cells. Double-click on the table to bring up the element 'Properties' dialog at the 'Wireframe' page, which provides the facilities for editing the table (adding, renaming and deleting columns and rows, changing the column width and editing the cell text) through context menu options and buttons. Note that the editor does not provide a true image of the table's appearance on the screen. • Draw Lines - click on the drop-down arrow and select True to display horizontal and vertical lines between the table cells, or False to hide the lines • Highlight Headers - click on the drop-down arrow and select True to highlight the header of each column, or False to leave the table columns a uniform color
Tab Host	Generates a tab control element on the

	diagram. You can name the tabs and mark them as selected; however, child elements will not 'switch' when changing tabs (that is, setting a different tab as selected will still display the same child items in the tab space). Tagged Values: • Tabs - click on the drop-down arrow and click on the tab that is to be highlighted as selected You can name the tabs, and add more to the list, by editing the 'Values:' list in the Tagged Value notes If you reduce the size of the element so that all tabs cannot be shown, a scroll icon () automatically displays in the top right corner of the element.
Simple List	Generates a list box containing a list of items with no sub-items. Double-click on the element to display the 'Properties' dialog at the 'Wireframe' page. To: • Change an item to different text, overtype the 'Item<n>' text in the 'Name' field • Highlight the item as selected, in the 'Selected' field click on the drop-down

	arrow and select True • Add another item, click on the element name and on the Add button, and type in the item name • Remove an item from the list, click on it and click on the Remove button • Move an item to a different position in the list, click on it and click on one of the  buttons as appropriate
2 Lane List	Generates a list box as for a Simple List, but each item name is in bold and can have a description underneath the item name. Double-click on the element to display the 'Properties' dialog at the 'Wireframe' page. • To add the description, click on the item name and, in the 'Text' field, type the description text Other options are the same as for the Simple List.
Checklist	Generates a list box as for a Simple List, but each item name has a tick outline to the right of it. For selected items, the outline is filled. Double-click on the element to display

	the 'Properties' dialog at the 'Wireframe' page. • To set a tick to selected (filled) click on the 'Checked' drop-down arrow and click on True; to change the tick to unselected (outline), set the field to False Other options are the same as for the Simple List.
Single Choice List	Generates a list box as for a Simple List, but each item name has a radio button outline to the right of it. For a selected item, the outline is filled. Double-click on the element to display the 'Properties' dialog at the 'Wireframe' page. • To set a radio button to selected (filled) click on the 'Selected' field drop-down arrow and click on True; to change the radio button to unselected (outline), set the field to False All items can be set to False (unselected), but if more than one item is set to True (selected) only the item lowest on the list displays as selected Other options are the same as for the Simple List.

Form Widgets

Item	Description
Switch	Generates an element representing a simple Android switch. The switch can have two states (such as On and Off) and a label taken from the name of the element. Tagged Values: • States - click on the drop-down arrow and select the switch position depicted by the element You can edit the 'Values:' field in the Tagged Value Notes to change the text of the state values You can add more than two values, but you can only toggle the 'States' field between the first two values in the list; the other values are ignored if selected
Rating Control	Generates an element depicting a star-rating band. The element always shows five stars, and the number of filled stars indicates the rating. Tagged Values:

	• Rating - click on the drop-down arrow and select the number of stars to show filled (the rating) You can only re-size the element horizontally; the vertical dimension adjusts automatically to always depict five uniformly-shaped stars.
Toggle Button	Generates an element depicting a single-celled switch with no label (the element name is not shown). You can edit the depicted state in the same way as for the 'Switch' element.
Progress Bar (Large)	Generates an element representing the circular Android progress icon.
Progress Bar (Horizontal)	Generates an element representing the progress of a process, defaulted to 30% complete. Tagged Values: • Progress - type the percentage completion of the progress to depict on the element
Button	Generates a simple labeled Button element, the label text being the name of the element.

Radio Button	Generates a labeled radio button element, the label text being the name of the element. Tagged Values: • State - click on the drop-down arrow and select the state to depict - 'Selected' (filled) or 'Unselected' (outline)
Checkbox	Generates a labeled checkbox element, the label text being the name of the element. Tagged Values: • State - click on the drop-down arrow and select the state to depict - 'Checked' (tick icon) or 'Unchecked' (box outline)
Seek Bar	Generates an element representing the progress in playing through an audio or video file. Tagged Values: • Progress - type the percentage progress to depict on the element
Keyboard	Generates an element that depicts a keyboard for Android applications. Tagged Values: • Type - switches the image between a text (QWERTY) keyboard and a

	numeric keyboard
Spinner	Generates an element representing the Android version of a drop-down combo box. Tagged Values: • Expanded - click on the drop-down arrow and select True to depict the combo box in use, displaying all values, or False to depict the combo box with a single selected value • Values - click on the drop-down arrow and select the value to depict as currently selected in the combo box You can change the text of the values, and add more to the list, by editing the 'Values:' list in the Tagged Value notes

Text Fields

Item	Description
Plain Text	Generates a text element with no border, the text being the element name. Tagged Values:

	• Align Text - click on the drop-down arrow and select to align the text to the left, center or right of the element frame • Multiline - click on the drop-down arrow and select True to allow the text to wrap around onto more than one line (automatically increasing the element depth), or False to only show the text that fits on one line within the current element width
Multiline Text	Generates a text element with no border, but that can contain multi-line text with basic HTML formatting. Tagged Values: • Align Text - click on the drop-down arrow and select to align the text to the left, center or right of the element frame • Text - click on the ... button to edit the Tagged Value Note screen, on which you can create the text to depict on the diagram; this can use HTML formatting tags such as for bold or <u> </u> for underlined - not all HTML formatting is supported

Image_Media

Item	Description
Image	Generates a place holder to indicate where an image will be placed on the phone or tablet. You can display an actual image by assigning an alternative image to the element.
Video Player	Generates an element that represents a video player control on the phone or tablet.
Audio Player	Generates an element that represents an audio player control on the phone or tablet.

Time_date

Item	Description
Calendar	Generates an element depicting a

	calendar (the default image), showing the current month, day and year based on the system date. Tagged Values: • DateFormat - click on the drop-down arrow and select the date format to display if spinners are shown: - d/m/y - m/d/y - y/m/d (not possible if the calendar is also displayed) • ShowCalendar - defaults to True to display the calendar; if spinners are also displayed, the calendar will force the spinner display to show the year at the right-hand end, and will replace the year spinner If set to False, the day, month and year spinners display regardless of the 'ShowSpinners' value • ShowSpinners - If set to True AND 'ShowCalendar' is True, displays spinner controls for the day and month (the calendar acts as a year marker) - If set to False AND 'ShowCalendar' is True, no spinners are

	shown with the calendar - If 'ShowCalendar' is set to False, the day, month and year spinners are automatically displayed in the format defined in 'DateFormat', showing the system date You cannot resize this element.
Date Picker	Generates an element that depicts a set of spinners showing today's date, derived from the system date. Tagged Values: • Date - click on the drop-down arrow and select a different date from the calendar; if the current date is not today, you can reset it to today's date by clicking on the Today button, or by simply clearing the 'Date' value field • DateFormat - as for the Calendar element • ShowCalendar - defaults to False to hide the calendar; otherwise, as for the Calendar element • ShowSpinners - as for the Calendar element You cannot resize this element.

Time Picker	Generates an element that depicts a pair of spinners showing the current time, in hours and minutes, derived from the system clock. Tagged Values: • 24 Hour View - click on the drop-down arrow and set to True to show the time in 24-hour format, or False to display the time in 12-hour format with AM or PM to the right of the time • Time - overtype the hours, minutes and AM/PM setting to display a time other than the system time (you can only set the time in 12-hour format); to revert to the system time, overtype the field with '12:00 AM' You cannot resize this element.
Clock	Generates an element that represents an analog clock face with hour and minute hands and no numerals, displaying the system time. You can change the rendition to a digital display. Tagged Values: • 24 Hour View - if you set the 'Type' to 'Digital', click on the drop-down arrow and select True to display the time in 24-hour format, or False to display the time in 12-hour format with AM or PM

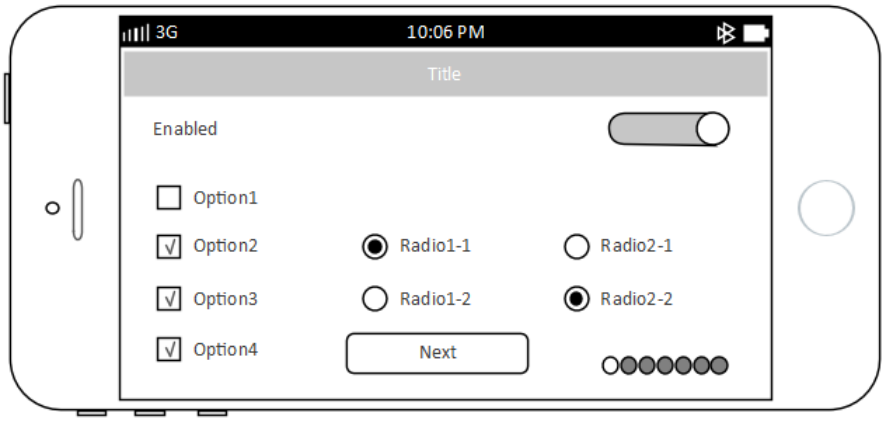
	to the right, as appropriate • Time - overtype the hours, minutes and AM/PM setting to display a time other than the system time (you can only set the time in 12-hour format); to revert to the system time, overtype the field with 12:00 AM • Type - click on the drop-down arrow and select 'Digital' to display the time as a digital display, or 'Analog' to display the time as the clock face You can resize the element in 'Analog' format, but not in 'Digital' format.

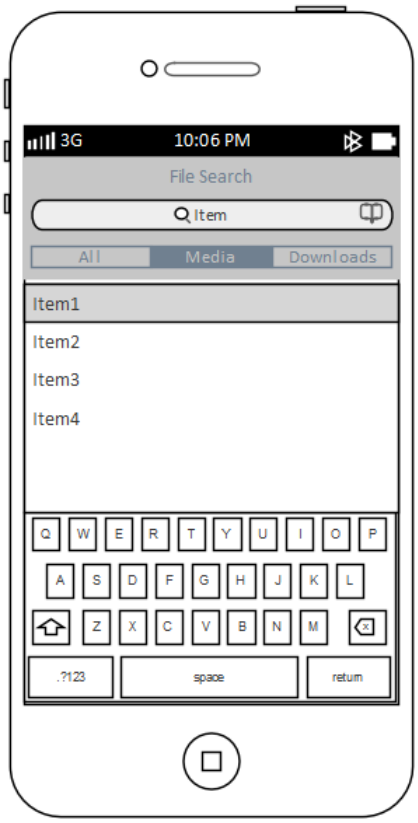
Android Patterns

Item	Description
Android Phone / Android Tablet	These Patterns create example models of the two Android product configurations. You can use them as your examples of how the elements are designed, as basic components of a larger model, or as the starting point to develop a more detailed model of one or more of the products.

Apple iPhone/Tablet Wireframe Toolbox

The 'Apple Wireframing' Diagram Toolbox pages provide the templates for modeling the physical appearance of an Apple iPhone or tablet at a given state of execution of an application. They also provide a number of Patterns for generating model structures for different versions of the iPhone or iPad.





Access

On the Diagram Toolbox, click on to display the 'Find Toolbox Item' dialog and specify 'Wireframing' or 'Apple'.

Ribbon	Design > Diagram > Toolbox
Keyboard Shortcuts	Ctrl+Shift+3

Apple Screen Types

Item	Description
iPad Air, iPad Mini, iPhone 4s, iPhone 5c, iPhone 5s, iPhone 6, iPhone 6 Plus	These icons each generate a frame for the device type you have selected. A prompt displays for you to specify portrait or landscape orientation. (The main illustration shows a landscape iPhone 5s frame and a portrait iPhone 4s frame.) Tagged Values: • ShowStatusBar - click on the drop-down button and select to 'Show' or 'Hide' the status bar image on the element

Controls

Item	Description
Check Box	Generates a labeled checkbox element, the label text being the name of the element. Tagged Values: • State - click on the drop-down arrow and select the state to depict - 'Checked'

	(tick icon) or 'Unchecked' (box outline)
Radio Button	Generates a labeled radio button element, the label text being the name of the element. Tagged Values: • State - click on the drop-down arrow and select the state to depict - 'Selected' (filled) or 'Unselected' (outline)
Combo Box	Generates an element representing a drop-down combo box. Tagged Values: • DropDownState - click on the drop-down arrow and select 'Open' to depict the combo box in use, displaying all values, or 'Closed' to depict the combo box with a single selected value • Items - click on the drop-down arrow and select the item to depict as currently selected in the combo box and, if the list of items is expanded, highlighted in the list You can change the text of the items, and add or remove items in the list, by editing the 'Values:' list in the Tagged Value notes
	Generates a Label text element. The

Label	name of the element is the text of the label. Tagged Values: • Align Text - sets the alignment of the text to either left, centered or right • Multiline - sets the label to display text over multiple lines
List	Generates a List box element. Tagged Values: • Items - click on the drop-down arrow and select the item to show highlighted in the list You can add, remove and rename items by editing the 'Values:' list in the Tagged Values Notes • ListType - click on the drop-down arrow and select to display the list as 'Simple', 'Numbered' or 'Bulleted'
Table	Generates a Table element with labeled columns, rows and cells. Double-click on the table to bring up the element 'Properties' dialog at the 'Wireframe' page, which provides the facilities for editing the table (adding, renaming and deleting columns and rows, changing the column width and editing

	the cell text) You can either edit the text by clicking on it, or by right-clicking and selecting an option. Note that the editor does not provide a true image of the table's appearance on the screen. Tagged Values: • Draw Lines - hides (or shows) the lines on all cells under the column headings so the table resembles a List element instead • Highlight Headers - highlights the header of each column so that it is easily distinguishable • Properties - displays the HTML of the table
Image	Generates a place holder to indicate where an image will be placed on the dialog. You can display an actual image by assigning an alternative image to the element.

Apple Controls

Item	Description
Address Bar	Generates a URL Address Bar element. Tagged Values: • Address - defines the text displayed in the address bar
App Icon	Generates an App Icon element. Tagged Values: • Image - specifies the name of an image held in the Image Manager, which is displayed as the appearance of the App icon (simply type in the name of the image as listed in the 'Name' column of the Image Manager) • Notification count - indicates the number of notifications this app has waiting; the number is displayed in the circle in the top right corner of the element
Button	Generates a labeled Button element. The label text is the name of the element. Tagged Values: • Button Style - sets the element shape: - 'Normal' draws a rectangle with rounded edges - 'Previous' draws a pennant shape

	pointing to the left - 'Next' draws a pennant shape pointing to the right • Text Alignment - defines the alignment of the button text within the element; left aligned, centered or right aligned
Date/Time Picker	Generates a Date and Time display element. Tagged Values: • Date - sets the date that the element displays; if no date is specified this will be the current system date • Expanded - if True, draws the element as a section of calendar with the current date and/or time selected (as defined by the 'Type' Tagged Value); if False, draws the element as a simple text string showing the date and/or time • Time - sets the time displayed on the element; overtype the hour, minute and AM/PM segments as necessary If no time is specified it will display the current system time (non-specified time is displayed as 12:00 AM) • Type - draws the element showing the Date, the Time or both Date & Time
	Generates a grouped list element with

Group List	two levels of entry. Double-click on the element to open the 'Properties' dialog at the 'Wireframe' page. Use this page to compose the list entries. Click on the root node (the element name). • Show Groups - click on the drop-down arrow and select True to show the first-level list items, or False to hide them; the second-level list items display in either case but must belong to a group item whether it is shown or not Click on the Group node (the first-level list item) directly underneath the root node. • Name - type in the text for the first-level list item (the item group name) Click on an item node (the second-level list item) directly under a Group node. Item nodes do not have any child nodes. • Name - type the name of the list item • Text - type any additional text to be displayed (by default) under the item name • Text near link - click on the drop-down

	arrow and select True to display the additional text opposite the item name, where you can also add a link, or False to keep the text underneath the item name • Image - select an image name from the drop-down list or simply type in the name as listed in the 'Name' column of the Image Manager, to display the image to the left of the item name • Is Link - click on the drop-down arrow and select True to indicate that the item links to another page or additional information by displaying a '>' character to the right of the item name; select False to hide the link character • Link Image - select an image name from the drop-down list or simply type in the name as listed in the 'Name' column of the Image Manager, to display the image (if 'Is Link' is True) as the link icon instead of the > character • Selected - click on the drop-down arrow and select True to highlight the item name as selected, or False to not highlight the item
Keyboard	Generates an element that depicts a

	keyboard. Tagged Values: • Type - switches the image between a text (QWERTY) keyboard and a numeric keyboard
Loading Icon	Generates an element that depicts the Apple loading icon.
Page Control	Generates an element rendered as a row of gray circles, indicating the number of pages available and which of those represents the currently-displayed page. Tagged Values: • Current page - draws a white circle in the sequence of gray circles to indicate which represents the current page displayed • Pages - the number of circles to draw, indicating the number of pages this control moves through (resize the element manually to display all the specified number of circles)
Search Bar	Generates an element representing a search field. Tagged Values: • DrawStyle - toggles the element's

	appearance between the default app search with a query spyglass, and a web search on an Apple device • Placeholder - type in the text that will be displayed in the control, to prompt the user to enter the search term • Prompt - type the prompt text to display above the query field, such as a reminder of what to search for; for example, if the search is to look up movies/recorded videos on the device, you might type 'Search for Videos' ('DrawStyle' must be set to default) • Scope - click on the drop-down arrow and select which search location is highlighted in the Scope Bar ('DrawStyle' must be set to default and 'Show Scope Bar' must be set to True) • Show Bookmarks - when set to True will draw a small book symbol on the right side of the query field, to indicate that this search control will store previous searches and can use them again ('DrawStyle' must be set to default) • Show Cancel Button - displays a Cancel button to the right of the query field ('DrawStyle' must be set to default)

	• Show Scope Bar - displays a row of scope options for this search control, underneath the query field ('DrawStyle' must be set to default) • Show search results - displays a drop down arrow on the right side of the search area, to indicate that the query will display the search results (overrides the Bookmark icon if both are set to display) ('DrawStyle' must be set to default) • Text alignment - aligns the Placeholder text to the Left, Right or Center of the query field
Segment Control	Generates an element depicting a set of tabs (defaulted to three tabs). Tagged Values: • Tabs - click on the drop-down arrow and select the number of the tab to highlight to indicate the current tab; open and edit the Tagged Value Notes to add, remove or rename tabs
Spinner Control	Generates an element representing a spinner control with a list of items that can be selected from. Tagged Values: • Check Selected Item - when set to True

	will draw a tick on the left hand side of the item defined as selected • Items - click on the drop-down arrow and select the item to indicate as selected; you can open and edit the Tagged Value Notes to add, rename or remove items from the list • Rounded Edges - click on the drop-down arrow and select the side(s) that show rounded corners; you can use this property to dock multiple spinner controls next to each other to create more complex spinner control selections, such as a page displaying a selection for Country, State, City and Suburb - set: - Both to set rounded corners on both sides, if the spinner control is on its own - Left to set rounded corners on the left edge and sharp corners on the right edge, if this spinner is the first in a row of docked spinner controls - Right to set rounded corners on the right edge and sharp corners on the left edge, if this spinner is the last in a row of docked spinner controls, or

	- None to set sharp corners on both sides, if this spinner is docked between two others • Text alignment - click on the drop-down arrow and select where to align the item text - to the left, right or center of the control
Stepper	Generates an element depicting a decrement/increment (minus/plus) control.
Switch	Generates an element representing a sliding on/off switch. Tagged Values: • DisplayText - toggles between displaying and hiding the two state text values • State - toggles between the 'on' state (gray background with white circle on the right and - if DisplayText is True - the 'on' text) and the 'off' state (white background with white circle on the left and - if DisplayText is True - the 'off' text) The state text values default to On and Off; you can edit the Tagged Value Notes to change these to any other pair of values (you can add more values but

	only the first two are applied)
Text Field	Generates a text field in which the end user can type free text, such as the name for a login page. The field contains the text 'TextField' and a crossed circle. Tagged Values: • Border Style - click on the drop-down arrow and select the border style for the element: - Rounded Rect - a rectangle with rounded corners - Bezel - a rectangle with bevelled top and left edges - Line - a rectangle with sharp corners and single-line edges - None - a rectangle with no borders • Text alignment - click on the drop-down arrow and set the text to align to the left, right or center of the control
Title	Generates an element that represents the title for a page, the element name being the title text (such as 'Settings'). The element is a rectangle with gray background and white text.

Toolbar	Generates an image that represents a Toolbar with default icons, which you can add to or replace with images from the Image Manager. Double-click on the element to display the 'Properties' dialog at the 'Wireframe' page, which lists the icons displayed in the frame of the Toolbar element. • To add a new icon click on 'Toolbar' and on the Add button, and in response to the prompt type in the name of the icon, which adds the name to the bottom of the list; click on the name and in the 'Image' Property field type the name of the icon file, as displayed in the 'Loaded Image' panel of the Image Manager You can add a 'Separator' (displayed as |) to the list, to separate different groups of icons on the toolbar; simply type the name 'Separator' in the name prompt To position the icon on the toolbar, click on it in the list and click on the  buttons • To remove an icon from the list, click on it and click on the Remove button; the icon is immediately removed from the list and, when you close the dialog,

	from the Toolbar
Tab Bar	Generates an element that depicts a row of tabs represented by images. Double-click on the element to display the 'Properties' dialog at the 'Wireframe' page, which shows the root node (the TabBar element itself) and the items contained in the TabBar (as images). • To add a new icon click on 'TabBar' and on the Add button, and in response to the prompt type in the name of the item, which adds the name to the bottom of the list To position the item on the TabBar, click on it in the list and click on the  buttons • To remove an item from the list, click on it and click on the Remove button; the item is immediately removed from the list and, when you close the dialog, from the TabBar Root node Properties: • Show Text - displays (True) or hides (False) the item names as text underneath the tab images • Background Color - click on the drop-down arrow and select the

	background color of the TabBar • Selected Font Color - click on the drop-down arrow and select the font color of the text if the item is selected • Non Selected Font Color - click on the drop-down arrow and select the font color of the text if the item is not selected Item node Properties: • Name - the name of the item, which can be displayed underneath the image in the TabBar (see *Show Text*) • Image - the name of the image file to show if the item is not selected; the file name is as listed in the Image Manager • Selected Image - the name of the image file to show if the item is selected; the file name is as listed in the Image Manager Select the 'General' page of this Tab Bar element's 'Properties' dialog, and click on the 'Tags' tab. Element Tagged Value: • SelectedTab - Click on the drop-down arrow and select the name of the item to represent as selected

Apple Patterns

You can use any of these Patterns as examples of how the elements are used, as basic components of a larger model, or as the starting point to develop a more detailed model of one or more of the products.

Item	Description
Apple iPad Air	Creates an example model for the Apple iPad Air.
Apple iPad Mini	Creates an example model for the Apple iPad Mini.
Apple iPhone 4s	Creates an example model for the Apple iPhone 4s.
Apple iPhone 5c	Creates an example model for the Apple iPhone 5c.
Apple iPhone 5s	Creates an example model for the Apple iPhone 5s.
Apple iPhone 6	Creates an example model for the Apple iPhone 6.
Apple iPhone	Creates an example model for the Apple

6 Plus	iPhone 6 Plus.

Windows Phone Wireframe Toolbox

The 'Windows Phone' Diagram Toolbox pages provide the templates for modeling the physical appearance of a Windows 8.1 Phone at a given state of execution of an application.

Access

On the Diagram Toolbox, click on  to display the 'Find Toolbox Item' dialog and specify 'Wireframing' or 'Windows Phone'.

Ribbon	Design > Diagram > Toolbox
Keyboard Shortcuts	Ctrl+Shift+3

Windows Screen Types

Item	Description
Windows Phone	Generates a frame with a screen area for the Windows 8 Phone device. A prompt displays in which you specify portrait or landscape orientation. Elements created within the screen area cannot be resized or moved to sit outside the borders of the screen. Elements created outside the frame can be dragged onto and off the frame, and can be as large as the view or element properties

	permit.

Text

Item	Description
Text Block	Generates an element that represents dominant text such as headings and labels. The element name is the displayed text. Double-click on the element to open the 'Properties' dialog at the 'Wireframe' page. This displays a default set of six levels of heading styles. Click on a style name to populate the 'Properties' panel on the right of the dialog, and click on the down-arrow for each property and select the appropriate settings for the style. If you prefer, you can also change the name of the style in the 'Name' property. If necessary, you can add further styles to the list. Click on the style **group** name and on the Add button. In the 'Enter name for item' prompt, type a name for the style and click on the OK button. The new style is added to the end of the list; if

	you want to move it further up the list, click on it and on the  icon. Again, you define the style using the 'Properties' panel. If you want to remove styles from the list, click on the style name and on the Remove button. When you have set the styles that can be used for this text, click on the 'General' page of the 'Properties' dialog and, in the 'Header Type' Tagged Value, click on the drop-down arrow and select the style to apply to the text of this specific Text Box.
Text Box	Generates a simple text field with a border, into which you can type any text you require. The element name is the displayed text, and does not wrap.

Controls

Item	Description
Button	Generates a rectangular icon representing a screen button, with the element name as the button text.

	Tagged Values: • State: click on the drop-down arrow and select the button state to represent: - Normal - the unselected button - Focused - the button when the cursor is passed over it - Selected - the button when it is clicked on - Disabled - the button grayed out, when it is not available
Checkbox	Generates an element representing a labeled checkbox, the element name being the label. Tagged Values: • Enabled: click on the drop-down arrow and select True to show the checkbox enabled for selection, or False to show the checkbox disabled and unavailable • State: click on the drop-down arrow and select 'Unchecked' to show the checkbox empty and unselected, or 'Checked' to show the checkbox selected with a tick inside it
Hyperlink Button	Generates a text element with the element name as the underlined text displayed, representing a hyperlink on the screen. Double-click on the element to open the

	'Properties' dialog at the 'Wireframe' page, which lists the three hyperlink states of normal 'Link', 'Visited' and 'Hover'. Click on a state name to populate the 'Properties' panel on the right of the dialog, and click on the down-arrow for each property and select the appropriate settings for the style to apply to that state. If you prefer, you can also change the name of the state in the 'Name' property. If necessary, you can add further states to the list. Click on the state **group** name and on the Add button. In the 'Enter name for item' prompt, type a name for the state and click on the OK button. The new state is added to the end of the list; if you want to move it further up the list, click on it and on the  icon. Again, you define the style using the 'Properties' panel. If you want to remove states from the list, click on the state name and on the Remove button. When you have set the states that the hyperlink can have, click on the 'General' page of the 'Properties' dialog and, in the 'State' Tagged Value, click on the drop-down arrow and select the state in which this hyperlink is to be depicted.

Image	Generates a rectangular object containing an 'X', to indicate the location of an image on the screen. There are no properties to set.
Radio Button	Generates an element representing a labeled radio button, the element name being the label. Tagged Values: • Enabled: click on the drop-down arrow and select True to show the radio button enabled for selection, or False to show the radio button disabled and unavailable • State: click on the drop-down arrow and select 'Unselected' to show the radio button empty, or 'Selected' to show the radio button with a filled circle inside it

Tiles

Tile elements add to the phone screen a panel that, depending on type, shows an image and/or some text. The panel cannot be resized, and if it displays text the text

occupies the top half of the element only. The amount of text displayed is influenced by the tile type, so you will need to experiment with the required type to see how much meaningful text you can display.

Item	Description
Collection Tile	Adds a tile with a random pattern, to represent a Windows Collection Tile. Tagged Values: • Header: type a suitable text string as the tile heading; the text is displayed when 'Show Back' is set to True and if the 'Tile Type' supports it • Show Back: click on the drop-down arrow and select True to display the back of the tile instead of the front; for some tile types the back does not display regardless of this setting • Text: a <memo> Tagged Value in which you type the text to display on the back of the tile; the format and font of the displayed text depends on the 'Tile Type' • Tile Type: click on the drop-down arrow and select the type of the collection tile; this will only affect the display of the back of the tile, the front will always remain the same (see the *Windows Tile Template Type*

	Descriptions web page for more information on tile types)
Image Tile	Adds a tile that initially displays as a box with an 'X' in the center, but is intended to show an image that you select. Tagged Values: • Image: click on the ... icon and select the image to display for this tile, from the 'Image Manager' dialog • Text: type in the text that will be displayed in white at the bottom of the image, dependent on the 'Tile Type' • Tile Type: click on the drop-down arrow and select the type of Image tile to display; this will either be an image only, or an image with text (see the *Windows Tile Template Type Descriptions* web page for more information on tile types)
Peek Tile	Adds a tile similar to an Image Tile, except that it can display the back of the tile to show more information. Tagged Values: • Header: type a suitable text string as the tile heading; the text is displayed when 'Show Back' is set to True and if

	the 'Tile Type' supports it • Image: click on the ... icon and select the image to display on the front of this tile, from the Image Manager window • Show Back: click on the drop-down arrow and select True to display the back of the tile instead of the front • Text: a <memo> Tagged Value in which you type the text to display on the back of the tile; the format and font of the displayed text depends on the 'Tile Type' • Tile Type: click on the drop-down arrow and select the type of Peek tile to display (see the *Windows Tile Template Type Descriptions* web page for more information on tile types)
Text Tile	Adds a tile that displays text only. Depending on tile type, you can show a text string in the top half of the panel and two text items in the bottom right corner of the panel. Tagged Values: • Block Text: type in a two-part text string to display at the bottom right of the tile, comprising a longer string that will be displayed in a small font, followed by a shorter string that will be

	displayed in a large font, the two strings separated by a semicolon; the short string will only display two characters in a square tile, or up to 5 characters in a wide tile, whilst the longer string can contain many more characters, for example: Messages;16 • Text: type in some additional text to display at the top of the tile (dependent on tile type) such as a description or definition of the object identified in the lower text • Tile Type: click on the drop-down arrow and select the type of Text tile to display (see the *Windows Tile Template Type Descriptions* web page for more information on tile types)

Windows Phone Controls

Item	Description
App Bar	Generates an element that represents the 'Windows App Bar', which is displayed at the bottom of the phone screen to supply additional commands. This can include

	up to a maximum of five icons and six strings. When you drag the icon onto the diagram, you are prompted to select the orientation of 'Portrait' or 'Landscape' to match the screen orientation. Double-click on the element to display the 'Properties' dialog at the 'Wireframe' page, displaying the element name at the top as the root node. Click on the element name and, in the right-hand 'Properties' panel, click on the drop-down arrow in the value field for the 'Mode' property and select: • 'Mini' - to represent the App Bar as a thin bar with just the expand menu icon () in the top right, with no other icons or text • 'Default' - to represent the App Bar as a thin bar unless it contains items, in which case it will display just the icon in a circle, with no text or icon names • 'Expanded' - to show the App Bar containing each icon in a circle, the name of the item under the circle and up to six text strings representing additional menu options You might prefer to set the App Bar properties after you have added some icons to it as child nodes. To add a child

	node, click on the root node and on the Add button and type in the name of the icon or object. In the right-hand panel, set the properties of the child node: • 'Name': displays the name of the item, which you can edit if necessary; if the App Bar is rendered in 'Expanded' mode, the name of a symbol or font item will be displayed below the icon, whilst for a text item it will be displayed below and to the left of the icons in a vertical list • 'Type': click on the drop-down arrow and select from the list of item types; the type you select will determine what other property prompts are displayed: - 'SymbolIcon': displays the item as a symbol icon - 'FontIcon': displays the item as a glyph, using a font - 'BitmapIcon': draws a selected image as the icon - 'Text': (applies only in 'Expanded' mode) displays the item name as a member of a vertical list below and to the left of the icons; a maximum of six items can be listed at once - 'Separator': draws a vertical line

	between icons, which counts as one of the five available spots for icons on the App Bar; Separator items do not display names • 'Symbol': (displays if the 'Type' is set to 'SymbolIcon') click on the drop down arrow and select the symbol from the list • 'FontFamily': (displays if the 'Type' is set to 'FontIcon') type in the name of the font to draw with, such as 'Segoe UI Symbol' • 'Glyph': (displays if the 'Type' is set to 'FontIcon') type in the Hex value of the glyph to draw - for example, for the © symbol you can set 'FontFamily' to 'Arial' and type the Hex code '00A9'; font codes in Windows can be found via 'Control Panel | Fonts | Find a Character' • 'Bitmap': (displays if the 'Type' is set to 'BitmapIcon') click on the drop-down arrow and select a bitmap (as listed in the Image Manager)
Date Picker	Generates an element that depicts three blocks showing today's day and date, month and year, derived from the system

	date. Tagged Values: • Date - if necessary, click on the drop-down arrow and select a different date from the calendar; if the displayed date is not today, you can reset it to today's date by clicking on the Today button • DateFormat - click on the drop-down arrow and select the date format to display: - d/m/y - m/d/y - y/m/d
Password Box	Generates an element that represents a password field on the screen. Tagged Values: • Password: a text string that represents a password • Password Character: a character that replaces each character of the 'Password' string when the password is hidden (when either 'Reveal Button' or 'Show Text' are set to False) • Reveal Button: if set to True (the default) draws a button that displays the 'Password' text string; if set to False

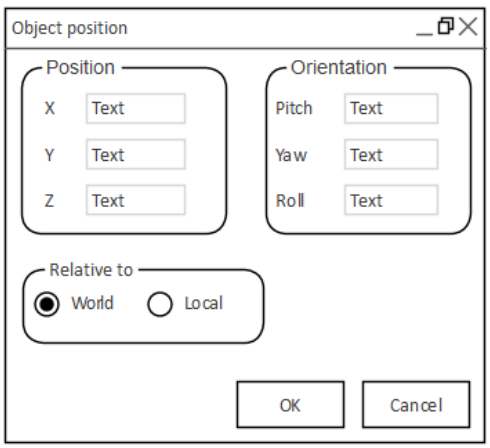
	the button is not displayed and the password string is represented by a string consisting of the 'Password Character' • Show Text: when 'Reveal Button' is set to True, setting 'Show Text' to True will display the 'Password' text string; otherwise a string displays composed of just the 'Password Character'
Progress Bar	Generates an element representing a 'process in progress' status bar, showing a number of 'dot' stages.
Progress Ring	Generates an element depicting the Windows 'processing in progress' circle of dots.
Search Bar	Generates an element representing a Windows search field, with the start search 'magnifying glass' icon at the end of it. Tagged Values: • Placeholder Text - defaults to the word 'Search'; if necessary, overtype this with an alternative text string
Slider	Generates an element representing a slide control switch, with the slider 50% of the

	way across. Tagged Values: • Fill amount - overtype the field with a value between 1 and 100, to set the percentage of the icon shown dark behind the slider
Time Picker	Generates an element that depicts two blocks showing the time in hours and minutes, in either 12-hour or 24-hour clock format. Tagged Values: • 24 Hour Display - click on the drop-down arrow and select True to display the time in 24-hour format, or False (the default) to display the time in 12-hour format • Time - displays the time in three sections - hours, minutes and AM/PM; click on and overtype each section with the required value for the time
Toggle Switch	Generates an element depicting a slide-over toggle switch with the switch on the left, representing the 'off' state. Tagged Values: • State - click on the drop-down arrow and select 'On' to represent the On state

	with the switch on the right of the icon, or 'Off' to move the switch back to the left of the icon to represent the Off state

Dialog Wireframe Toolbox

The 'Dialog Wireframing' Diagram Toolbox pages provide the templates for modeling the physical design, appearance and operation of a screen dialog. You can see and build on an example of how to model the dialog interface by dragging the 'Dialog Pattern' icon onto a diagram.



Access

Ribbon	Design > Diagram > Toolbox : > Specify 'Wireframing - Dialog' in the 'Find Toolbox Item' dialog
Keyboard Shortcuts	Ctrl+Shift+3 : > Specify 'Wireframing - Dialog' in the 'Find Toolbox Item' dialog
Other	You can display or hide the Diagram Toolbox by clicking on the or icons

	at the left-hand end of the Caption Bar at the top of the Diagram View.

Screen Types

Item	Description
Dialog	Generates an element that represents a dialog outline, with a title (the element name). This has several components that you can expose and define in the element 'Properties' dialog, some using the Tagged Values on the 'Wireframing' tab of the 'General' page, and some using the 'Wireframe' page of the dialog. **Tagged Values:** • Close Button - defaults to True to show a 'close dialog' icon in the top right corner of the dialog; click on the drop-down arrow and select False to omit the icon • Maximize Button - defaults to False to omit a 'maximize dialog' icon from the top right corner of the dialog; click on the drop-down arrow and select True to show the icon

	• Minimize Button - defaults to False to omit a 'minimize dialog' icon from the top right corner of the dialog; click on the drop-down arrow and select True to show the icon • ScrollbarH - defaults to '<none>' to omit a horizontal scrollbar on the dialog; click on the drop-down arrow and select 'Bottom' or 'Top' to show a scrollbar in the corresponding position • ScrollbarV - defaults to '<none>' to omit a vertical scrollbar on the dialog; click on the drop-down arrow and select 'Left' or 'Right' to show a scrollbar in the corresponding position • StatusBar - define the display of the Status Bar using the 'Wireframe' page • User Icon - type in the name of an icon exactly as listed in the Image Manager, to display that icon in the top left corner of the dialog before the dialog (element) name **Wireframe Page:** Displayed by default when you double-click on the element. Use the options to modify the Status Bar at the bottom of the dialog. Click on the element name.

	• Zoombar - defaults to True to depict a zoom bar at the right hand end of the Status Bar; click on the drop-down arrow and select False to omit the zoom bar • Resize Handle - defaults to True to depict a resize icon (triangle of dots) in the bottom right corner of the Status Bar; click on the drop-down arrow and select False to omit the icon Click on 'Label' - this defines the first segment of the progress bar at the left hand end of the Status Bar. • Name - if necessary, overtype the text with another name for the progress bar; this text is not displayed but the field cannot be blank • Text - if necessary, overtype the default text with different text to display next to the progress bar • Type - defaults to 'Text' to display the string contained in the 'Text' field; if necessary, click on the drop-down arrow and select: - Filled Progress Bar - to replace the text with a part-filled rectangle (to depict a section of Progress Bar with a portion of processing complete), or

	- Block Progress Bar - to replace the text with rectangle containing blocks (to depict a section of Progress Bar with processing in action) Click on 'Progressbar' - this defines a second segment of the progress bar. The properties are the same as for the first segment except that there is no default 'Text' and the 'Type' defaults to 'Filled'. If you want to add another segment to the progress bar, click on the element name, click on the Add button and provide a name for this segment. Provide values for the 'Text' and 'Type' properties as before. If you want to remove segments from the progress bar, click on the segment name and click on the Remove button. If you remove all segments and the 'Zoom Bar' and 'Resize Handle', the status bar itself is removed. You can change the sequence of segments by clicking on a segment name and on the  buttons.
Client Area	Generates a frame element that represents the client area of the device. Tagged Values: • BorderStyle - click on the drop-down

	arrow and select to render the border as a 'Solid' line or a 'Dashed' line, or to have no border at all ('None') • ScrollbarH - click on the drop-down arrow and select to place a horizontal scrollbar at the 'Top' or 'Bottom' of the client area, or to have no horizontal scrollbar ('<none>') • ScrollbarV - click on the drop-down arrow and select to place a vertical scrollbar on the 'Left' or 'Right' of the client area, or to have no vertical scrollbar ('<none>')

Controls

Item	Description
Button	Generates an element that represents a simple button with the element name as the button text. Tagged Values: • State - click on the drop-down arrow and select a status for the button: - Normal - simple rectangle, for normal display where the button

	is just available - Focused - a highlighted inner border, indicating, for example, that the button is the default selection - Selected - filled rectangle, indicating that the button is selected - Disabled - pale text and border, indicating that the button is not available
Check Box	Generates a labeled checkbox element, the label text being the name of the element. Tagged Values: • State - click on the drop-down arrow and select the state to depict - 'Checked' (tick icon) or 'Unchecked' (box outline)
Radio Button	Generates a labeled radio button element, the label text being the name of the element. Tagged Values: • State - click on the drop-down arrow and select the state to depict - 'Selected' (filled) or 'Unselected' (outline)
Combo Box	Generates an element representing a

	drop-down combo box. Tagged Values: • DropDownState - click on the drop-down arrow and select 'Open' to depict the combo box in use, displaying all values, or 'Closed' to depict the combo box with a single selected value • Items - click on the drop-down arrow and select the item to depict as currently selected in the combo box and, if the list of items is expanded, highlighted in the list You can change the text of the items, and add or remove items in the list, by editing the 'Values:' list in the Tagged Value notes
Label	Generates a Label text element, on which the label text is the name of the element. Tagged Values: • Align Text - click on the drop-down arrow and select to align the text to the left, center or right of the element frame • Multiline - click on the drop-down arrow and select True to allow the text to wrap around onto more than one line (automatically increasing the element depth), or False to only show the text

	that fits on one line within the current element width
List	Generates a List box element. Tagged Values: • Items - click on the drop-down arrow and select the item to show highlighted in the list You can add, remove and rename items by editing the 'Values:' list in the Tagged Values Notes • ListType - click on the drop-down arrow and select to display the list as 'Simple', 'Numbered' or 'Bulleted'
Table	Generates a Table element with labeled columns, rows and cells. Double-click on the table to display the element 'Properties' dialog at the 'Wireframe' page, which provides the facilities for editing the table (adding, renaming and deleting columns and rows, changing the column width and editing the cell text) through context menu options and buttons. Note that the editor does not provide a true image of the table's appearance on the screen. Tagged Values: • Draw Lines - click on the drop-down

	arrow and select True to display horizontal and vertical lines between the table cells, or False to hide the lines • Highlight Headers - click on the drop-down arrow and select True to highlight the header of each column, or False to leave the table columns a uniform color
Image	Generates a place holder to indicate where an image will be placed on the phone or tablet. You can display an actual image by assigning an alternative image to the element.

Dialog Controls

Item	Description
Checkbox List	Generates an element depicting a checklist where each item has a checkbox on the left hand side. Double-click on the table to display the element 'Properties' dialog at the 'Wireframe' page, which you use to

	maintain this element. For each 'Checkbox' item, complete these fields: • Name - type the name or text of the item • Checked - click on the drop-down arrow and select True to show a ticked checkbox against the item, or False to show a cleared checkbox To add another item to the list, click on the element name and on the Add button, then provide a name for the item. To remove an item from the list, click on the item and on the Remove button. You can change the sequence of items by clicking on an item name and on the  buttons.
Format Bar	Generates a simple element representing a text formatting bar. If you want to represent a toolbar containing icons you have defined, use the 'Toolbar' icon.
List View	Generates an element representing a horizontal, rectangular or vertical list of text items (depending on the size of the element) with or without associated

	images. Double-click on the table to display the element 'Properties' dialog at the 'Wireframe' page, which you can use to add, remove or change the items and their icons. For each item, complete these fields: • Name - type the name or text of the item; this field cannot be left blank • Image - type the name of the image, or click on the drop-down arrow and select the name, as listed in the Image Manager • Selected - click on the drop-down arrow and select True to highlight the name of the item, or False to omit any highlight; more than one item can be highlighted at once To add another item to the list, click on the element name and on the Add button, then provide a name for the item. To remove an item from the list, click on the item and on the Remove button. You can change the sequence of items by clicking on an item name and on the  buttons. If you reduce the size of the element so that not all items can be shown, a scroll

	bar automatically displays on the right edge of the element: 
Status Bar	Generates a status bar element identical to the automatically-generated status bar on the 'Dialog' element, except that you can position this element independently of the dialog, as required.
Toolbar	Generates an element to represent a toolbar of icons, already set up with some standard toolbar icons. Double-click on the element to display the 'Wireframe' page of the 'Properties' dialog, which you can use to add, remove or change the items and their icons. For each item, complete these fields: • Name - type the name or text of the item; this field cannot be left blank • Image - type the name of the icon image, or click on the drop-down arrow and select the name, as listed in the Image Manager To add another item to the list, click on the element name and on the Add button, then provide a name for the item. You can add one or more items called

	'Separator' to the list, which display as a |, to partition groups of related icons in the toolbar. If you add an image to this item, the image is overridden by the |. To remove an item from the list, click on the item and on the Remove button. You can change the sequence of items by clicking on an item name and on the  buttons.
Audio Player	Generates a simple element to indicate an audio player control.
Calendar	Generates an element representing a basic calendar, showing today's date derived from the system date. Tagged Values: • Date - either: - Overtype the date displayed in this field or - Click on the drop-down arrow to display an active calendar page and select the date on that; click on the Today button to reset the date to the system date • Highlight Date - click on the drop-down arrow and select True to highlight the set date on the calendar

	icon, or False to omit the highlight
Header	Generates an element representing a title or header text on a dialog. The text itself is the element name. The element can reflect one of a range of header levels, each with a different font style. You specify which level of header to display using the 'HeaderType' Tagged Value within the element. Double-click on the element to display the 'Wireframe' page of the 'Properties' dialog, which you can use to add, remove or change the header levels and styles. For each item, complete these fields: • Name - type the name of the header level; this field cannot be left blank • Color - click on the drop-down arrow and select the appropriate color from the palette • Font Size - type in the font size, or click on the drop-down arrow and select the type size from the list • Font Family - click on the drop-down arrow and select the font type from the list • Font Style - click on the drop-down arrow and select the style from the list;

	select blank for no applied style • Text Align - click on the drop-down arrow and select to align the text left, right or centered • Text Decoration - click on the drop-down arrow and select whether to show an underline or a line-through, or neither (blank) To add another heading level to the list, click on the element name and on the Add button, then provide a name for the level. To remove a level from the list, click on the item and on the Remove button. You can change the sequence of heading levels by clicking on a level name and on the  buttons. Tagged Values: • HeaderType - click on the drop-down arrow and select the heading level to display
Hyperlink	Generates an element representing a hyperlink in one of three states: 'Link', 'Visited' and 'Hover' (mouse-over). The Hyperlink text is the element name. Double-click on the element to display the 'Wireframe' page of the 'Properties'

	dialog, which you can use to add, remove or change the hyperlink states. For each state, complete these fields: • Name - type the name of the state; this field cannot be left blank • Color - click on the drop-down arrow and select the appropriate color from the palette • Font Size - type in the font size, or click on the drop-down arrow and select the type size from the list • Font Family - click on the drop-down arrow and select the font type from the list • Font Style - click on the drop-down arrow and select the style from the list; select blank for no applied style • Text Align - click on the drop-down arrow and select to align the text left, right or centered • Text Decoration - click on the drop-down arrow and select whether to show an underline or a line-through, or neither (blank) To add another hyperlink state to the list, click on the element name and on the Add button, then provide a name for the state.

	To remove a state from the list, click on the item and on the Remove button. You can change the sequence of states by clicking on a state name and on the  buttons. Tagged Values: • State - click on the drop-down arrow and select the state to represent on the diagram
Menu Bar	Generates an element representing a standard menu bar at the top of the screen, initially with three options ('File', 'Edit' and 'View') with the 'File' option expanded into a sub-menu. Double-click on the element to display the 'Wireframe' page of the 'Properties' dialog, which you can use to add, remove or change the menu options in the top level, sub-menu and - if you prefer - further sub levels. For each menu option - at any level - complete these fields: • 'Name' - type in the name for this menu item • 'Expanded' - click on the drop-down arrow and select True to show the sub-menu for this option (if it has one),

	or False to hide the sub-menu • 'Highlighted' - click on the drop-down arrow and select True to highlight this option in the menu, or False to leave it un-highlighted; if 'Expanded' is set to True, the option is automatically highlighted To add a menu sub-option at any level, click on the parent option name and on the Add button, then provide a name for the sub-option. You can add one or more items called 'Separator' to the list, which displays as a horizontal line across the list, to partition groups of related options in the menu. To remove an option from the list, click on the item and on the Remove button. You can change the sequence of options by clicking on an option name and on the  buttons. To move an option between two levels of menu, click on the option name and on the  buttons.
Paragraph	Generates a text element with no border, but that can contain multi-line text with basic HTML formatting. Tagged Values:

	• Align Text - click on the drop-down arrow and select to align the text to the left, center or right of the element frame • Text - click on the ... button to edit the 'Tagged Value Note' screen, on which you can create the text to depict on the diagram; this can use HTML formatting tags such as for bold or <u> </u> for underlined - not all HTML formatting is supported
Progress Bar	Generates a status bar element representing the progress of a process. Tagged Values: • Fill Percentage - defaults to 30% complete; type the percentage completion to depict (the amount the Progress Bar is filled) • Fill Style - click on the drop-down arrow and select: - 'Filled' to represent the percentage completion as a solid bar, or - 'Block' to represent the percentage completion as a series of blocks or vertical bars
Rating	Generates an element depicting a

Control	star-rating band. The element always shows five stars, and the number of filled stars indicates the rating. Tagged Values: • Rating - click on the drop-down arrow and select the number of stars to show filled (the rating) You can only re-size the element horizontally; the vertical dimension adjusts automatically to always depict five uniformly-shaped stars.
Scrollbar - Horizontal	Generates an element representing a horizontal scrollbar.
Scrollbar - Vertical	Generates an element representing a horizontal scrollbar.
Tab Control	Generates an element representing a series of tabs or pages. You can name the tabs and mark them as selected; however, child elements will not 'switch' when changing tabs (that is, setting a different tab as selected will still display the same child items in the tab space). Tagged Values: • Tabs - click on the drop-down arrow and click on the tab that is to be highlighted as selected

	You can name the tabs, and add more to the list, by editing the 'Values:' list in the Tagged Value notes If you reduce the size of the element so that all tabs cannot be shown, a scroll icon () automatically displays in the top right corner of the element.
Text Field	Generates a text element with a pale border, the text being the element name, representing a simple data entry field.
Video Player	Generates an element that represents a video player control.
Date/Time Picker	Generates an element that represents the Microsoft Date/Time Picker. Tagged Values: • CustomFormat - type in a custom format for any or all of the day, date, month, year, hour, minute and second components, using these case-sensitive codes (listed in alphabetical order): - d - display the day of the month using either one or two digits - dd - display the day of the month using two digits, digits 1 to 9 preceded by a 0 - ddd - display the day of the week

	as a three-character abbreviation - dddd - display the name of the day in full - h - display the hour using either one or two digits, in 12 hour clock format - hh - display the hour using two digits, digits 1 to 9 preceded by a 0, in 12 hour clock format - H - display the hour using either one or two digits, in 24 hour clock format - HH - display the hour using two digits, digits 1 to 9 preceded by a 0, in 24 hour clock format - m - display the minutes using either one or two digits - mm - display the minutes using two digits, digits 1 to 9 preceded by a 0 - M - display the number of the month using either one or two digits - MM - display the number of the month using two digits, digits 1 to 9 preceded by a 0 - MMM - display the name of the month as a three-character abbreviation

	- MMMM -display the name of the month in full - s - display the seconds using either one or two digits - ss - display the seconds using two digits, digits 1 to 9 preceded by a 0 - t - identify morning or afternoon with a single character (A for AM, P for PM) - tt - identify morning or afternoon with the two-character abbreviation AM or PM - y - display the year using a single digit (2022 is displayed as 2) - yy - display the year using two digits (2022 is displayed as 22) - yyyy - display the year in full (for example, 2022) • Date - overtype the date, or click on the drop-down arrow to display a calendar from which you can select the date to display; defaults to today's date - if you change this to a fixed date and want to return to the current (system) date, click on the Today button • Format - click on the drop-down arrow and select the code for the format to use to display the date and time:

	- Long - the full day name, the day date, the full month name, the full year (for example: Wednesday, 17 February 2022) - Short - the day of the month in single/double digits, the month as a two-digit number, the year in full (for example: 17/02/2022) - Time - the hour, minutes and seconds in 12-hour format (for example 12:59:34 PM) - Custom - applies the custom format you defined in the 'CustomFormat' Tagged Value
Tree Control	Generates an element representing a hierarchy or tree of nodes, with broken lines connecting sibling nodes and an expansion box (+ or -) next to nodes that have subnodes. Double-click on the element to display the element 'Properties' dialog at the 'Wireframe' page, which you can use to add, remove or change the tree nodes in the top level, sub-level and - if you prefer - further sub levels. For each node - at any level - complete these fields:

	• Name - type in the name for this node • Expanded - click on the drop-down arrow and select True to show the subordinate nodes for this node (if it has any), or False to hide the subordinate nodes • Selected - click on the drop-down arrow and select True to highlight this node, or False to leave it un-highlighted To add a sub-node at any level, click on the parent node name and on the Add button, then provide a name for the sub-node. To remove a node from the hierarchy, click on the node name and on the Remove button. You can change the sequence of nodes by clicking on a node name and on the  buttons. To move a node between two levels of the hierarchy, click on the node name and on the  buttons. You can also directly edit the XML of the element on the 'Wireframe' tab of the Properties window.
Groupbox	Generates an element representing a

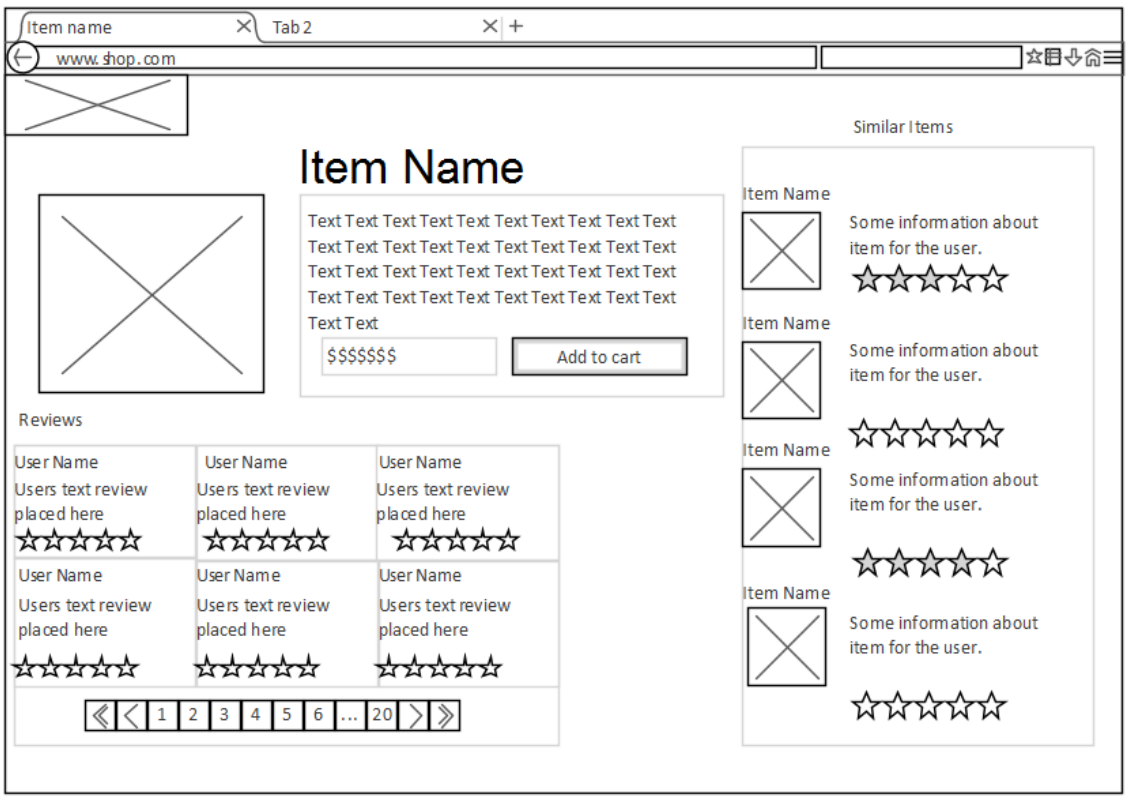
	Groupbox, with the name of the element in the top left corner. You can use this element to enclose and group other elements of the dialog.

Patterns

Item	Description
Dialog	This Pattern generates a small dialog containing three panels with data entry fields and radio buttons, and two buttons, as depicted at the start of this topic. You can use this as an example, or as the basis for a similar dialog design.

Webpage Wireframe Toolbox

The 'Webpage Wireframe' Diagram Toolbox pages provide the templates for modeling the schematics, blueprints or framework of a website, defining how the web pages work. You can see and build on an example of how to model the webpage interface by dragging the 'Webpage Pattern' icon onto a diagram. Then, using the Toolbox, you can drag Element-types on to your diagram to add any key user-interface features.



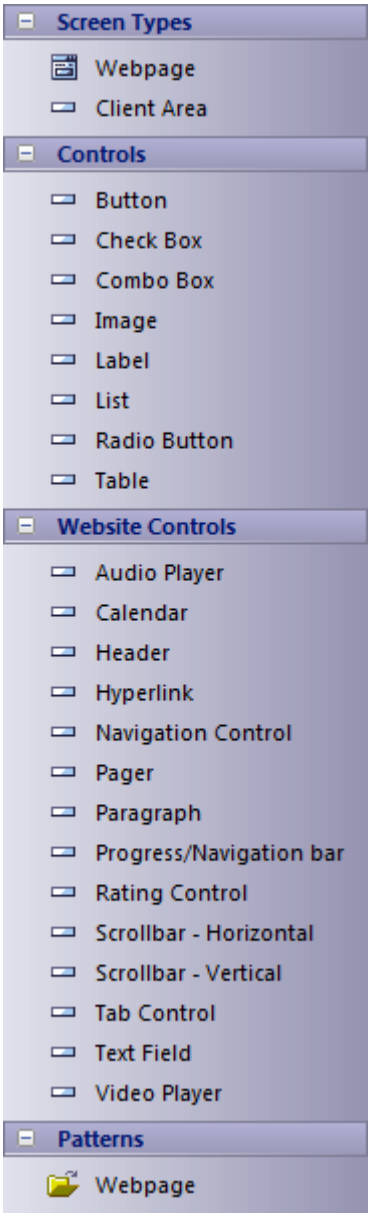
Access

On the Diagram Toolbox, click on to display the 'Find Toolbox Item' dialog and specify 'Wireframing' or

'Webpage'.

Ribbon	Design > Diagram > Toolbox
Keyboard Shortcuts	Ctrl+Shift+3

Toolbox



Element Features

Image	Detail
ComboBox	Combobox, List, Progress/Navigation bar and Tab Control all provide lists of values.

	To extend a list: • Open the Properties Dialog (Double-click or Shift+Enter) • Select the Wireframing tab • Select the Tagged Value • Click on the  icon • and then add, edit or remove items from the 'Values' list.
Calendar Element	The Calendar element, when created, defaults to the current day and continues to update each day unless you set a value in the element's Date Tagged Value. If set, the date remains static until it is reset to 'Today' in the Tagged Value.
Paragraph	In the Paragraph element you can: • Set text alignment in the 'Align Text' Property • Edit the text in using the [...] option in the 'Text' Property (which is of type <memo>). The text can be use basic HTML text formatting, as for other formatted notes in elements.
Navigation Control	Navigation Control defines a menu with, if required, multiple levels of sub-menu options. You can add and remove options

	at any level using the 'Wireframe' tab of the 'Properties' dialog. Each option, at any level, has these properties: • 'Name': Set the text of the menu option • 'Expanded': Indicate if the option will display its sub options (if any); expanded items are always highlighted, regardless of the setting of 'Highlighted' • 'Highlighted': Draw the item with a different colored background
Progress/Navigation Control	Progress Navigation Control provides a visual progression through pages. To extend a list: • Open the Properties Dialog (double-click or shift+enter) • Select the Wireframing tab • Select the *Steps* Tagged Value • Click on the  icon • and then add, edit or remove items from the 'Steps' list.
Hyperlink	Hyperlink and Header elements are both also defined on the 'Wireframe' tab, and have a number of style properties that you set using simple drop-down lists: • Color

	• Font Size • Font Family • Font Style • Text Align • Text Decoration

Simple Analysis Models

Create Rigorous and Articulated Models of Analysis Concepts from Strategy to Implementation

Analysts have typically had to work in a variety of text-based tools and general purpose drawing packages. The disparate nature of the output from these tools required analysts to copy and paste content between the tools and there was also the considerable task of keeping all the heterogeneous documents up-to-date. Enterprise Architect equips the analyst with a single tool that can seamlessly tie all their work together in a rigorous model that can be updated in a single place, versioned, and used to produce high quality documentation in a variety of formats. The models they create can be traced back to strategic models including stakeholder goals and business drivers and plans, and the same models can also be traced down to architectural solutions and designs, creating a single fabric of knowledge.

Analysis Model Diagram Types

Analysis Tool	Detail
Data Flow Diagrams	A Data Flow diagram (DFD) is a graphical representation of the flow of

	data through an information system; it can also be used to visualize data processing (structured design). Developing a DFD helps in identifying the transaction data in the data model.
Custom Models	Custom models provide a number of extensions to the UML model and help you to perform exploratory and non-rigorous experimentation with model elements and diagrams. For example, using a Custom diagram you can model requirements, user interfaces or custom designs.
Mind Mapping	A Mind Map is an image-centered diagram used to represent semantic or other connections between words, ideas, tasks or other items arranged radially around a central key word or idea. A Mind Map is used to generate, visualize, structure and classify ideas, and as an aid in study, organization, problem solving, decision making, and writing.

Mind Mapping

This text is derived from the Mind Map entry in the online Wikipedia:

"A Mind Map is a diagram used to represent words, ideas, tasks or other items linked to and arranged radially around a central key word or idea. It is used to generate, visualize, structure and classify ideas, and as an aid in study, organization, problem solving, decision making, and writing."

"A Mind Map is an image-centered diagram that represents semantic or other connections between portions of information. By presenting these connections in a radial, non-linear graphical manner, it encourages a brainstorming approach to any given organizational task, eliminating the hurdle of initially establishing an intrinsically appropriate or relevant conceptual framework to work within."

"The elements are arranged intuitively according to the importance of the concepts and are organized into groupings, branches, or areas. The uniform graphic formulation of the semantic structure of information on the method of gathering knowledge, may aid recall of existing memories."

The use of the term 'Mind Maps' is trademarked in the UK and the USA by The Buzan Organization, Ltd.

Mind Mapping in Enterprise Architect

Within Enterprise Architect you can develop Mind Maps quickly and simply, through use of an MDG Technology integrated with the Enterprise Architect installer. The Mind Mapping facilities are provided in the form of:

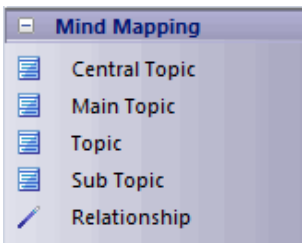
- A Mind Mapping diagram type, accessed through the 'New Diagram' dialog
- A 'Mind Mapping' page in the Toolbox
- Mind Mapping element and relationship entries in the 'Toolbox Shortcut' menu and Quick Linker

Access

Use any of the methods outlined here to display the Diagram Toolbox, then click on  to display the 'Find Toolbox Item' dialog and specify 'Mind Mapping'.

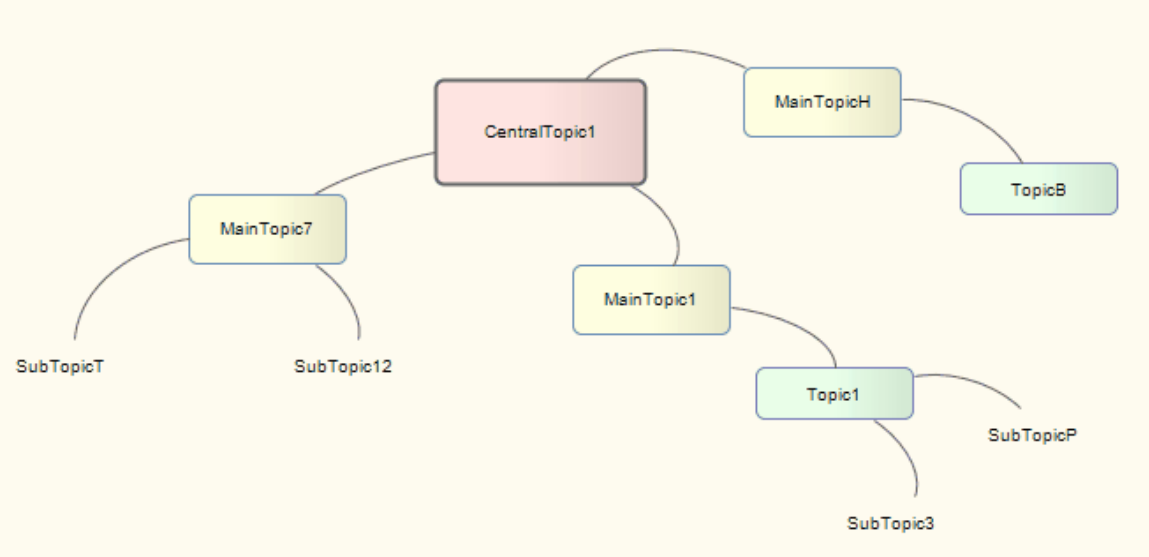
Ribbon	Design > Diagram > Toolbox
Keyboard Shortcuts	Ctrl+Shift+3
Other	Click the  icon on the Diagram caption bar to display the Diagram Toolbox

Mind Mapping Toolbox Page

Page Appearance	Description
	• Central Topic is the main theme of the Mind Map; you would normally have one or two of these on the diagram, but can add as many as are necessary • Main Topic represents the immediate concepts generated by the Central Topic • Topic represents the larger divisions of a Main Topic • Sub Topic represents the finer divisions of a Topic or Main Topic; you could also have Subtopics of Subtopics to represent increasingly finer distinctions • Relationship represents the connection between any two elements; you can have several Relationships per element Each relationship has three anchor points, so you can curve the lines to develop the flow of concepts more easily

Mind Map Diagram

This is the general appearance of a Mind Mapping diagram.



As the elements can represent any concept, object or relationship, you can use the full range of element properties and features to expand on what the element represents, including adding Note elements. However, to preserve the simplicity and readability of the diagram itself, you cannot display the element compartments on the diagram.

Whiteboards

Scribble and Present Ideas Using an In-Model Electronic Whiteboard

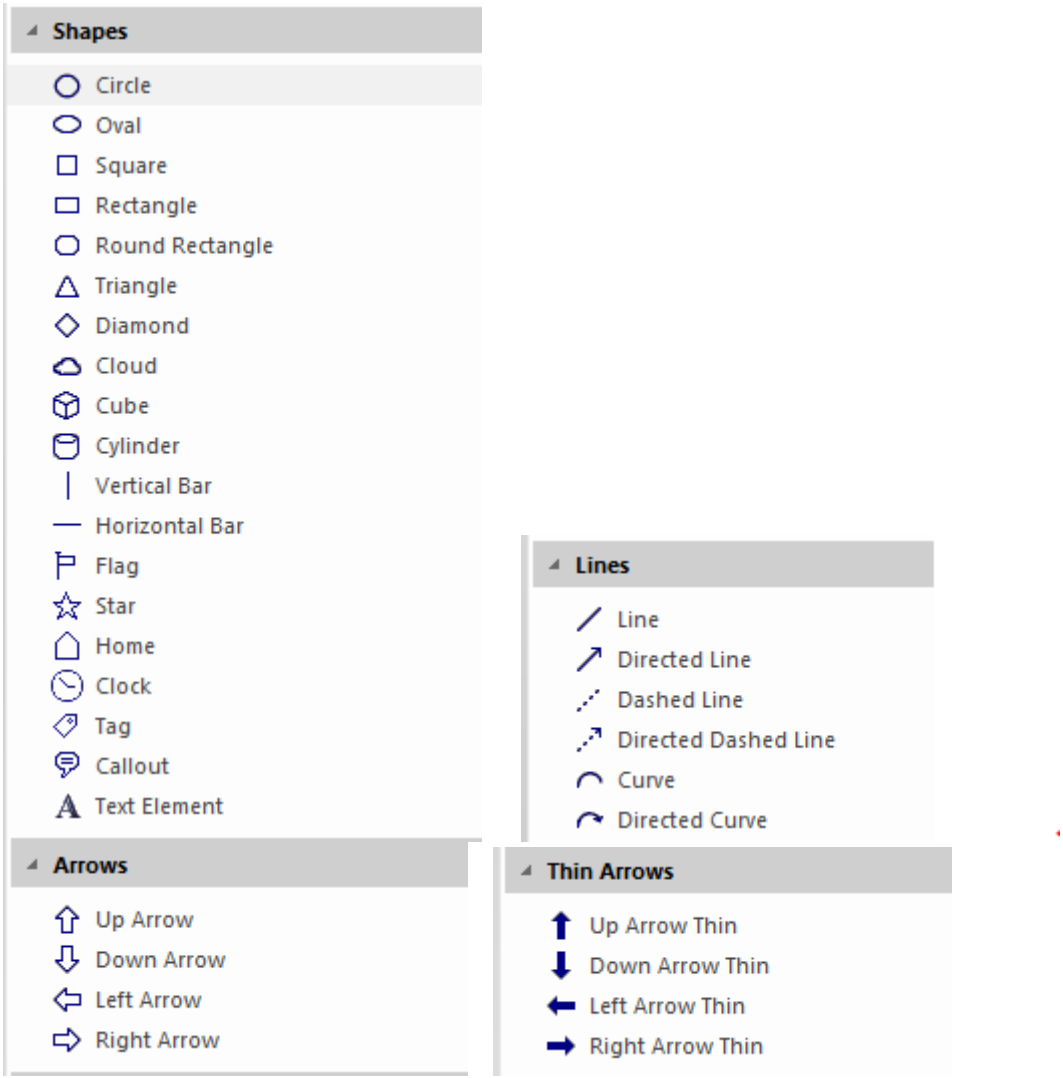
The Whiteboard Technology provides a set of Diagram Toolbox pages with a selection of stereotyped element icons that generate simple shapes suitable for use on hand-drawn and whiteboard diagrams.

Access

Use any of the methods outlined here to display the Diagram Toolbox, then click on  to display the 'Find Toolbox Item' dialog and specify 'Whiteboard'.

Ribbon	Design > Diagram > Toolbox
Keyboard Shortcuts	Ctrl+Shift+3
Other	Click the  icon on the Diagram caption bar to display the Diagram Toolbox

Whiteboard Toolbox Pages



The Whiteboard Toolbox pages provide these shape icons:

- Shapes: Circle, Oval, Square, Rectangle, Round Rectangle, Triangle, Diamond, Cloud, Cube, Cylinder, Vertical Line, Horizontal Line, Flag, Star, Home, Clock, Tag, Callout and Text Element.
- Arrows: Up, Down, Left and Right
- Thin Arrows: Up, Down, Left and Right

Lines: Line, Directed Line, Dashed Line, Directed Dashed Line, Curve, Directed Curve.

Page Appearance	Description

Whiteboard Diagram

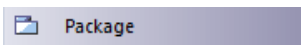
You can create either a Whiteboard diagram or a Hand-drawn diagram. Both of these diagrams, when opened, cause the Whiteboard Toolbox pages to be opened.

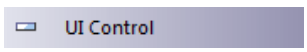
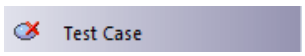
Custom Diagram

A Custom diagram is an extended Class diagram that is used to capture requirements, user interfaces or custom-design models.

Custom models provide a few extensions to the UML model and enable some exploratory and non-rigorous experimentation with model elements and diagrams. You generate Custom diagram elements and connectors from the 'Custom' pages of the Diagram Toolbox.

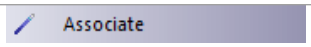
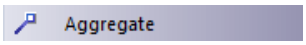
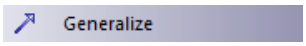
Custom Diagram Element Toolbox Icons

Icon	Description
	Packages are used to organize your project contents, but when added onto a diagram they can be use for structural or relational depictions.
	A Requirement element captures the details of a system requirement.
	An Issue element represents an item of concern (that might or might not occur) or a failure to meet a defined requirement in the current system.

	A Change element represents a change in the defined requirements for the current system and can be used to request and manage the change process.
	A Screen element is used to prototype a User Interface screen flow.
	A UI Control element represents a user interface control element (such as an edit box).
	A Test Case is a stereotyped Use Case element which enables you to give greater visibility to tests.
	An Entity is a stereotyped Object that models a store or persistence mechanism that captures the information or knowledge in a system.

Custom Diagram Connector Toolbox Icons

Icon	Description

	An Association implies that two model elements have a relationship, usually implemented as an instance variable in one or both Classes.
	An Aggregation connector is a type of association that shows that an element contains or is composed of other elements.
	A Generalization is used to indicate inheritance.
	A Realizes connector represents that the source object implements or Realizes its destination object.
	A Nesting Connector is an alternative graphical notation for expressing containment or nesting of elements within other elements.

Data Flow Diagrams

Visualize the Flow of Data and Drill Down from Context to Detail

This text is derived from the Data Flow diagram entry in the online Wikipedia.

'A Data Flow diagram (DFD) is a graphical representation of the 'flow' of data through an information system. A Data Flow diagram can also be used for the visualization of data processing (structured design). It is common practice for a designer to draw a context-level DFD first which shows the interaction between the system and outside entities. This context-level DFD is then 'exploded' to show more detail of the system being modeled.'

'Data Flow diagrams were invented by Larry Constantine ... based on Martin and Estrin's "data flow graph" model of computation. (They) are one of the three essential perspectives of Structured Systems Analysis and Design Method SSADM. The sponsor of a project and the end users will need to be briefed and consulted throughout all stages of a system's evolution. With a Data Flow diagram, users are able to visualize how the system will operate, what the system will accomplish, and how the system will be implemented. The old system's Data Flow diagrams can be drawn up and compared with the new system's Data Flow diagrams to draw comparisons to implement a more efficient system.'

'Developing a DFD helps in identifying the transaction data in the data model.'

Access

Use any of the methods outlined here to display the Diagram Toolbox, then click on  to display the 'Find Toolbox Item' dialog and specify 'Data Flow Diagrams'.

Ribbon	Design > Diagram > Toolbox
Keyboard Shortcuts	Ctrl+Shift+3
Other	Click the » icon on the Diagram caption bar to display the Diagram Toolbox

Data Flow Diagrams in Enterprise Architect

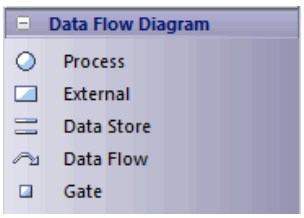
Within Enterprise Architect, you can develop Data Flow diagrams quickly and simply through use of an MDG Technology integrated with the Enterprise Architect installer. The Data Flow diagram facilities are provided in the form of:

- A Data Flow diagram type, accessed through the 'New

Diagram' dialog

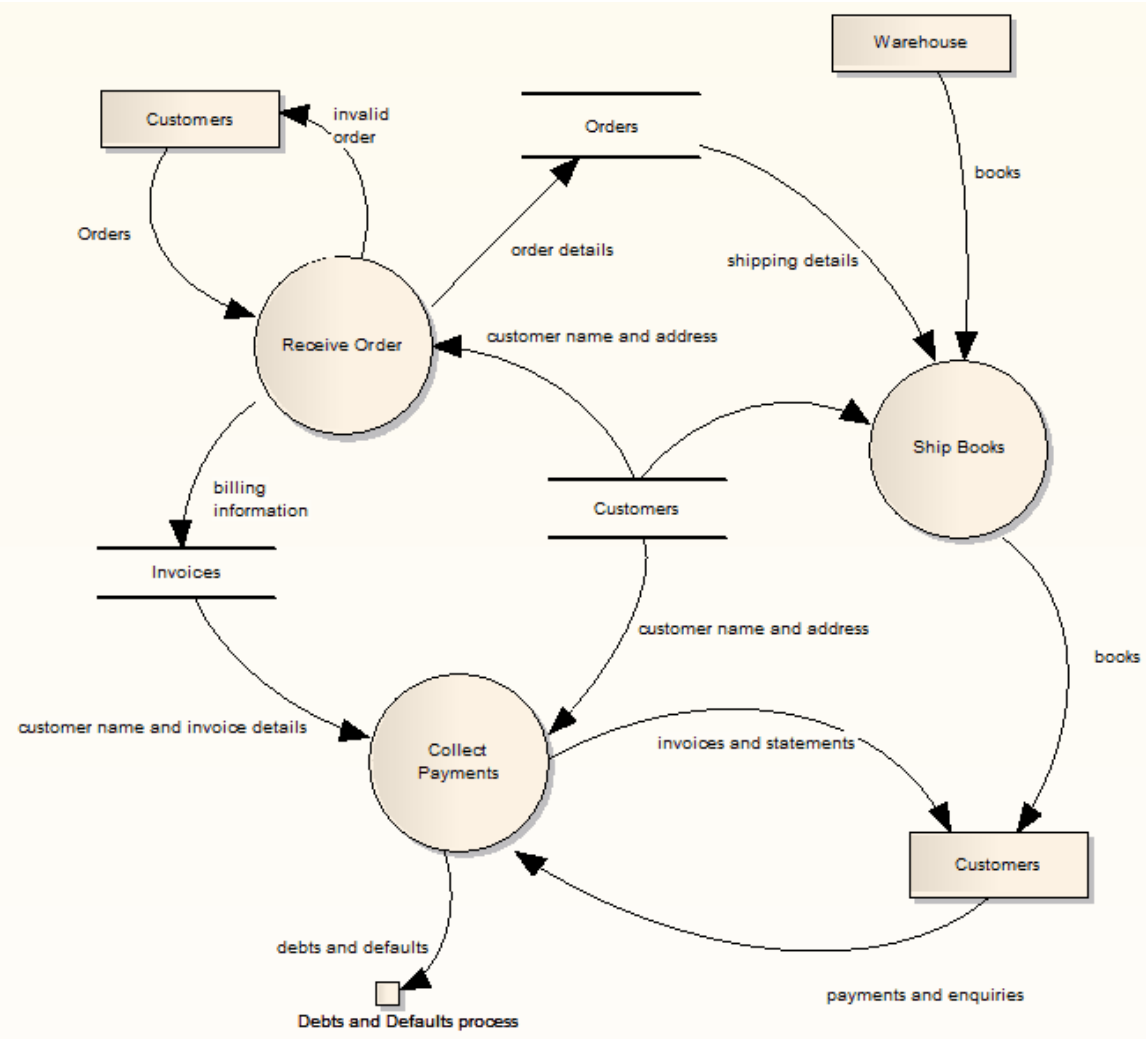
- A 'Data Flow Diagram' page in the Diagram Toolbox
- Data Flow element and relationship entries in the Toolbox 'Shortcut Menu' and Quick Linker

Data Flow Diagram Toolbox Page

Page Appearance	Description
	Process is a process or activity in which data is used or generated. External represents an external source, user or repository of the data. Data Store represents an internal physical or electronic repository of data, into and out of which data is stored and retrieved. Data Flow (connector) represents how data flows through the system, in physical or electronic form. Gate represents the termination point of incoming and outgoing messages on a lower level diagram (that is, messages to and from processes depicted elsewhere).

Data Flow Diagram Appearance

When dragged onto a Data Flow diagram, the elements and relationships have these appearances:



To preserve the simplicity and readability of the diagram, you cannot display element compartments on the diagram.

Context Diagram

A Context diagram is a top-level Data Flow diagram that has just one Process element representing the system being modeled, showing its relationship to external systems.

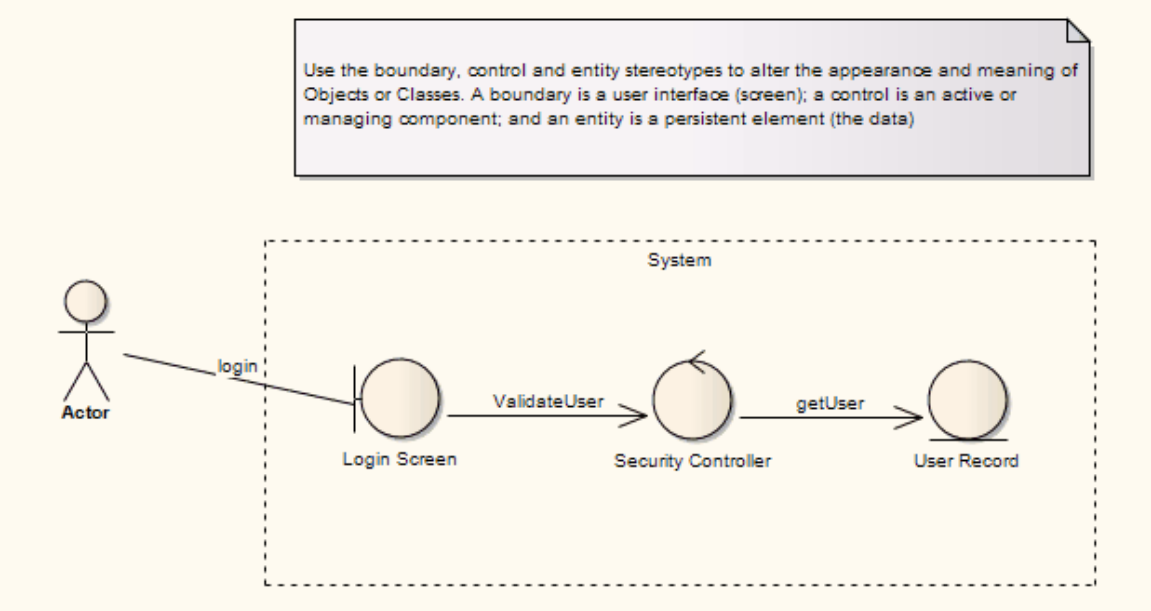
Analysis Stereotypes

Enterprise Architect provides some built in stereotypes that you can assign to an element during analysis. The effect of each of these stereotypes is to display a different icon from the normal element icon, giving a visual key to the element's purpose. The main types of inbuilt icon for elements created using the stereotypes include:

- Boundary - for a system boundary (for example, a Login screen)
- Control - to specify an element is a controller of some process (as in the Model-View-Controller Pattern)
- Entity - the element is a persistent or data element

The elements created using these stereotypes are illustrated in this Robustness diagram. Also see the Business Modeling elements, used in Business Modeling diagrams and Business Interaction diagrams.

Example of Analysis Stereotypes in use


Use the boundary, control and entity stereotypes to alter the appearance and meaning of Objects or Classes. A boundary is a user interface (screen); a control is an active or managing component; and an entity is a persistent element (the data)
System
Actor
login
Login Screen
ValidateUser
Security Controller
getUser
User Record

Analysis Diagram

An Analysis diagram is a simplified Activity diagram, used to capture high level business processes and early models of system behavior and elements. It is less formal than some other diagrams, but provides a useful means of capturing the essential business characteristics and requirements.

Robustness diagrams, which are used extensively in ICONIX, can also be created as Analysis diagrams.

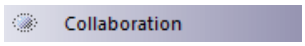
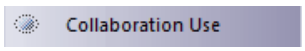
You generate Analysis diagram elements and connectors from the 'Analysis' pages of the Diagram Toolbox.

Example Diagram

Example Analysis Diagram

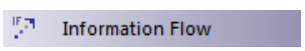
Analysis Diagram Element Toolbox Icons

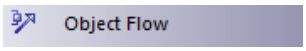
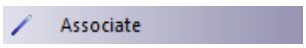
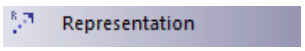
Icon	Description
	An Actor is a user of the system; 'user' can mean a human user, a machine, or even another system or subsystem in the model.

	An Object is a particular instance of a Class at run time.
	A Process is an Activity element with the stereotype process, which expresses the concept of a business process.
	A Collaboration defines a set of cooperating roles and their connectors.
	A Collaboration Use element allows for a Pattern defined by a Collaboration to applied to a specific situation.
	The Send element is used to depict the action of sending a signal.
	A Receive element is used to define the acceptance or receipt of a request.
	An Information Item element represents an abstraction of data, which data can be conveyed between two objects.
	A Decision is an element that indicates a point of conditional progression: if a condition is true, then processing continues one way; if not, then another.

	A Merge Node brings together a number of alternative flow paths in Activity, Analysis and Interaction Overview diagrams.
	A Boundary is a stereotyped Object that models some system boundary, typically a user interface screen.
	A Control element represents a controlling entity or manager that organizes and schedules other activities and elements.
	An Entity is a stereotyped Object that models a store or persistence mechanism that captures the information or knowledge in a system.

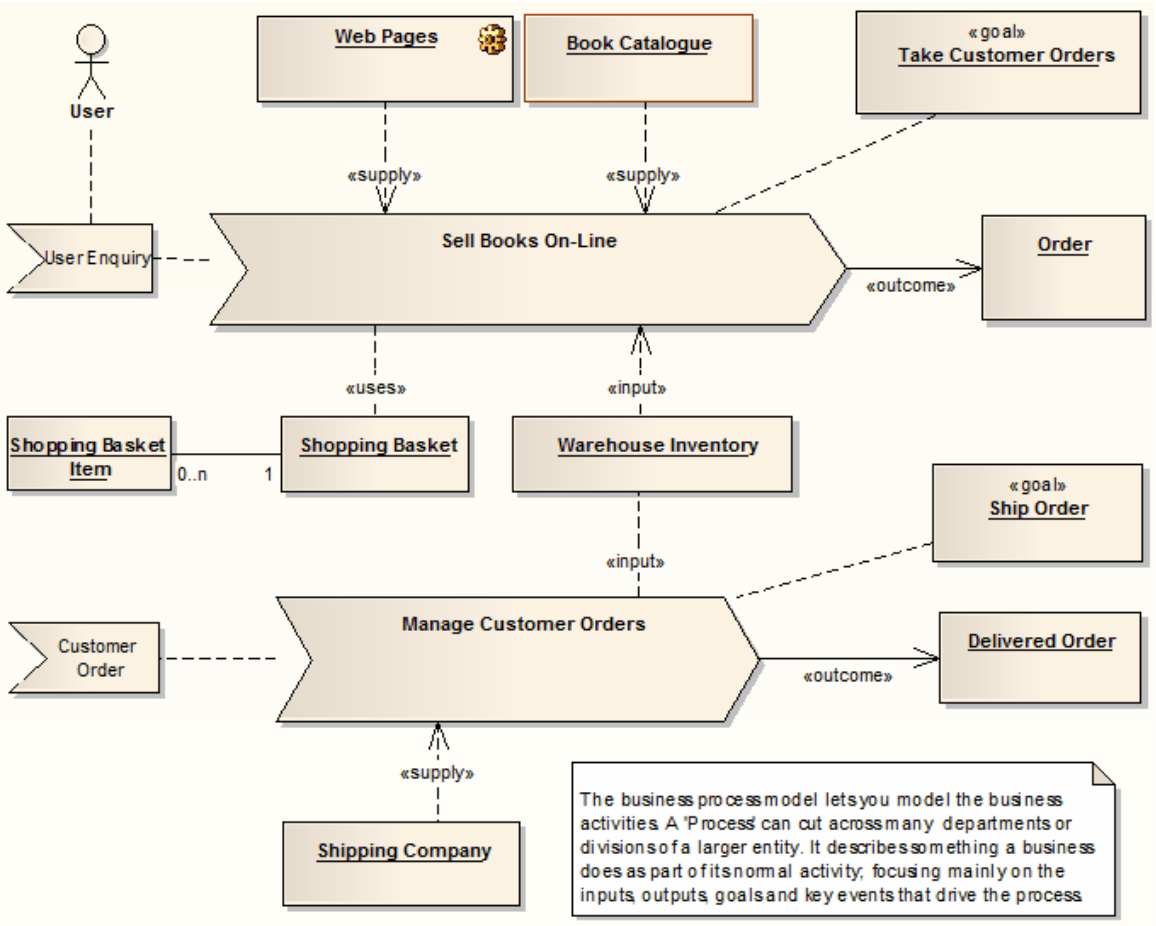
Analysis Diagram Connector Toolbox Icons

Icon	Description
	An Information Flow represents the flow of Information Items (either Information Item elements or classifiers) between two

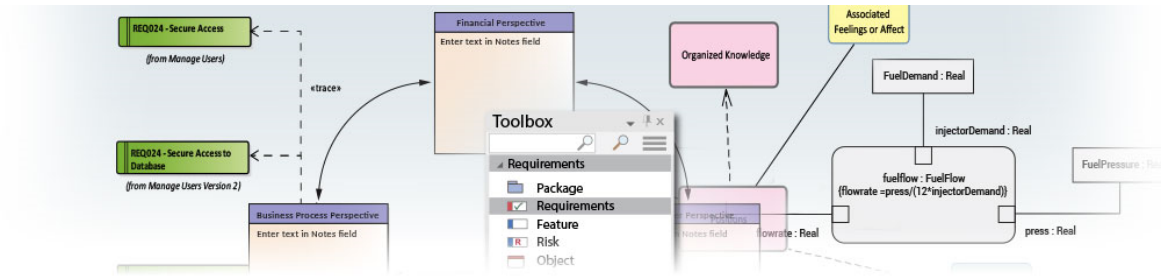
	elements in any diagram.
	An Object Flow connects two elements, with specific data passing through it, modeling an active transition.
	An Association implies that two model elements have a relationship, usually implemented as an instance variable in one or both Classes.
	A Realizes connector represents that the source object implements or Realizes its destination object.
	The Representation relationship is a specialization of a Dependency, connecting Information Item elements that represent the same idea across models.

Example Analysis Diagram

Enterprise Architect supports some of the Eriksson-Penker Business Extensions that facilitate business process modeling. You can also load the complete Eriksson-Penker Business Extensions UML Profile into Enterprise Architect to create detailed process models.



More Domain Models



Enterprise Architect is a multi-featured and comprehensive analysis and design, execution and testing tool that helps you to create models of enterprise, business, engineering, real-time and software systems. Its feature set is so rich that no matter what you want to do there is almost always a tool feature that will help you achieve the task. Enterprise Architect supports a wide range of frameworks, modeling languages and technology platforms and languages. Every discipline will benefit from using the tool, and when an entire team is using the tool the models created by each group can be seamlessly integrated into a whole that will bring great clarity and efficiency to the project or program of work and substantially reduce the risk of failure.

High level managers can create Mind Maps, and strategic thinkers can create stakeholder profiles and define business goals and drivers; Enterprise Architects can create road maps and application inventories, Analysts can describe current and future process models, and Requirements Analysts can create requirement models. On the solution side of the model, a Solutions Architect can detail the interfaces between applications, developers can create code stubs, and Testers can define test cases, just to detail a few

possibilities.

Enterprise Architect allows you to follow any project methodology, whether it is an Agile process or a formal military process or one of the standard frameworks such as TOGAF or Zachman. You are also free to set up your own modeling process, and Enterprise Architect has built in features for defining software development processes. Modeling can start at any point and in real projects it typically doesn't follow the sequences described in text books; Enterprise Architect allows you to work in a flexible way but supports you with a versatile set of tools.

Enterprise Architect has a flexible feature that helps you to create Profiles of the UML so even if you cannot find a suitable modeling solution built into the core product, you can extend the tool by creating your own profile.

Domain Based Diagrams

Enterprise Architect supports a wide range of modeling languages, such as UML, SysML and BPMN, but in addition to the diagrams that are defined as part of these languages Enterprise Architect has a rich set of additional (extended) diagrams, including Mind Maps, User Interface diagrams and Data Modeling diagrams; there is even a general purpose Custom diagram. This allows several specialists such as Strategic Thinkers, User Experience Designers and Scientists to contribute to the models and to create a repository of articulated knowledge that has not been possible before.

Domain Based Diagram Types

Diagram Type	Detail
Analysis Diagram	An Analysis diagram is a simplified Activity diagram, which is used to capture high level business processes and early models of system behavior and elements.
Custom Diagram	A Custom diagram is an extended Class diagram that is used to capture

	requirements, user interfaces or custom-design models.
Requirements Diagram	A Requirements diagram is a Custom diagram used to describe a system's requirements or features as a visual model.
Maintenance Diagram	A Maintenance diagram is a Custom diagram used to describe change requests and issue items within a system model.
User Interface Diagram	User Interface diagrams are Custom diagrams used to visually mock-up a system's user interface using forms, controls and labels.
Data Modeling Diagram	A Data Modeling diagram is a Class diagram used for representing database schemas.
Documentation	Virtual documents enable you to structure and filter your document and web reports by selecting, grouping and ordering individual Packages independent of the organization of the Browser window.
Business Modeling	Business Modeling diagrams and Business Interaction diagrams enable you

and Business Interaction	to model both the structure and behavior of a business system. Business Modeling diagrams are based on a Class (UML Structural) diagram, whilst Business Interaction diagrams are based on a Sequence (UML Behavioral) diagram.

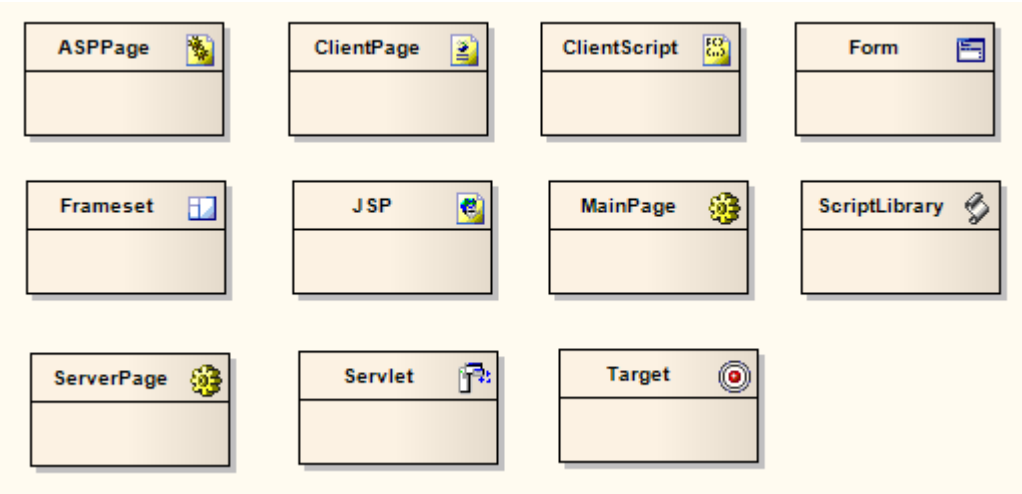
Modeling Disciplines

Modelers from all Disciplines and Domains can Create Industry Standard Models

Regardless of the discipline or your specific field in the system development life-cycle you will find modeling tools and options to create expressive and industry standard models. Regardless of whether you are creating new system components or selecting a commercially available product, or if you work in a strategic, business, technology or engineering division of an organization Enterprise Architect allows you to create collaborative models that show how all facets of the product fit together. The collaboration features ensure that requirements, design and implementation choices are all resolved as a solution is developed.

Web Stereotypes

Enterprise Architect supports a number of stereotypes for web page modeling, the graphical elements for which display with a graphical icon instead of the usual «stereotype» format. These stereotypes are only supported for Class elements. These are the various graphical icons and their associated stereotypes:



A similar set of web modeling elements and their relationships are also available through dedicated 'Web Modeling' pages in the Diagram Toolbox.

Set a web icon

Step	Action
1	Create a new Class element in a diagram.

2	Display the Class 'Properties' dialog.
3	In the 'Stereotype' field, either type in the required stereotype name or click on the drop-down arrow and select the required stereotype (as named previously).
4	Click on the OK button. The Class displays as in one of the examples shown.

User Interface Diagrams

User experience and user interface design have traditionally been modeled in a variety of tools that are separate from other disciplines, leading to a disconnect between these models and the other analysis and technology models.

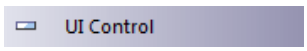
Enterprise Architect allows you to model a wide range of user interfaces and platforms, including client software, web sites and pages, and mobile devices such as phones and tablets. It uses compelling representations of the physical devices and the platforms to make these models appealing and useful for walks-through with users. The elements in these models can also be traced to other elements in the repository, including design principles, requirements, use cases and user stories, stakeholders' concerns, information models, architecture and design models. StateMachine diagrams can also be created to represent the important states of the user interface, and these can be traced to testing models.

The User Interface diagram is an extended diagram type that provides a set of wire framing toolboxes with a rich palette of user interface elements for Android and Apple devices, as well as for web pages and dialogs. There is also a facility for modeling Win32® user interfaces, with a toolbox containing a wide range of controls such as Check Boxes, Spin Controls, Tree Controls and many more.

Example Diagram

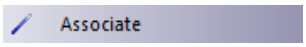
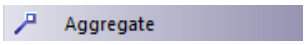
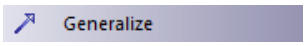
Example User Interface Diagram

User Interface Diagram Element Toolbox Icons

Icon	Description
	Packages are used to organize your project contents, but when added onto a diagram they can be use for structural or relational depictions.
	A Screen element is used to prototype a User Interface screen flow.
	A UI Control element represents a user interface control element (such as an edit box).
	An Object is a particular instance of a Class at run time.

User Interface Diagram Connector Toolbox

Icons

Icon	Description
	An Association implies that two model elements have a relationship, usually implemented as an instance variable in one or both Classes.
	An Aggregation connector is a type of association that shows that an element contains or is composed of other elements.
	A Generalization is used to indicate inheritance.
	A Realizes connector represents that the source object implements or Realizes its destination object.

Notes

- Using stereotyped Classes, you can model the design of a web page user interface

- The Enterprise Architect Professional, Corporate, unified and Ultimate editions also include the MDG Win32 UI Technology, with which you can design user interface components that render more precisely as Win32 ® User Interface elements

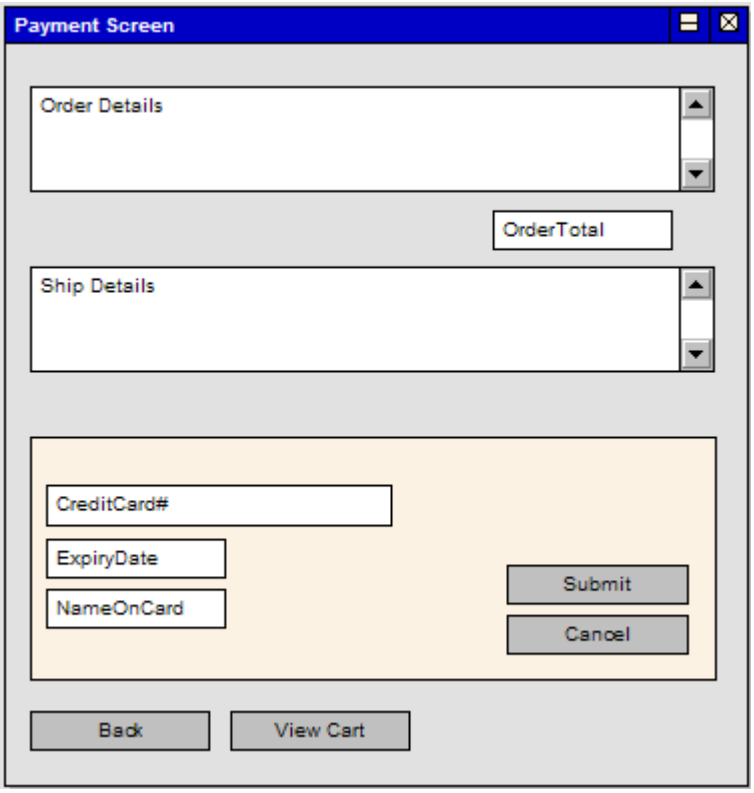
Screen

A Screen is used to prototype User Interface screen flow. By using UML features such as Requirements, Constraints and Scenarios against User Interface diagram elements, you can build up a solid and detailed understanding of user interface behavior without having to use code. This provides an excellent means of establishing the precise behavior of the system from a user's perspective and, in conjunction with the Use Case model, defines exactly how a user gets work done.

Web pages can also be prototyped and specified rigorously using Enterprise Architect's custom interface extensions.

Example

This example diagram illustrates some features of Enterprise Architect's screen modeling extensions that support web page prototyping. By adding requirements, rules, scenarios and notes to each element, a detailed model is built up of the form or web page, without having to resort to GUI builders or HTML.

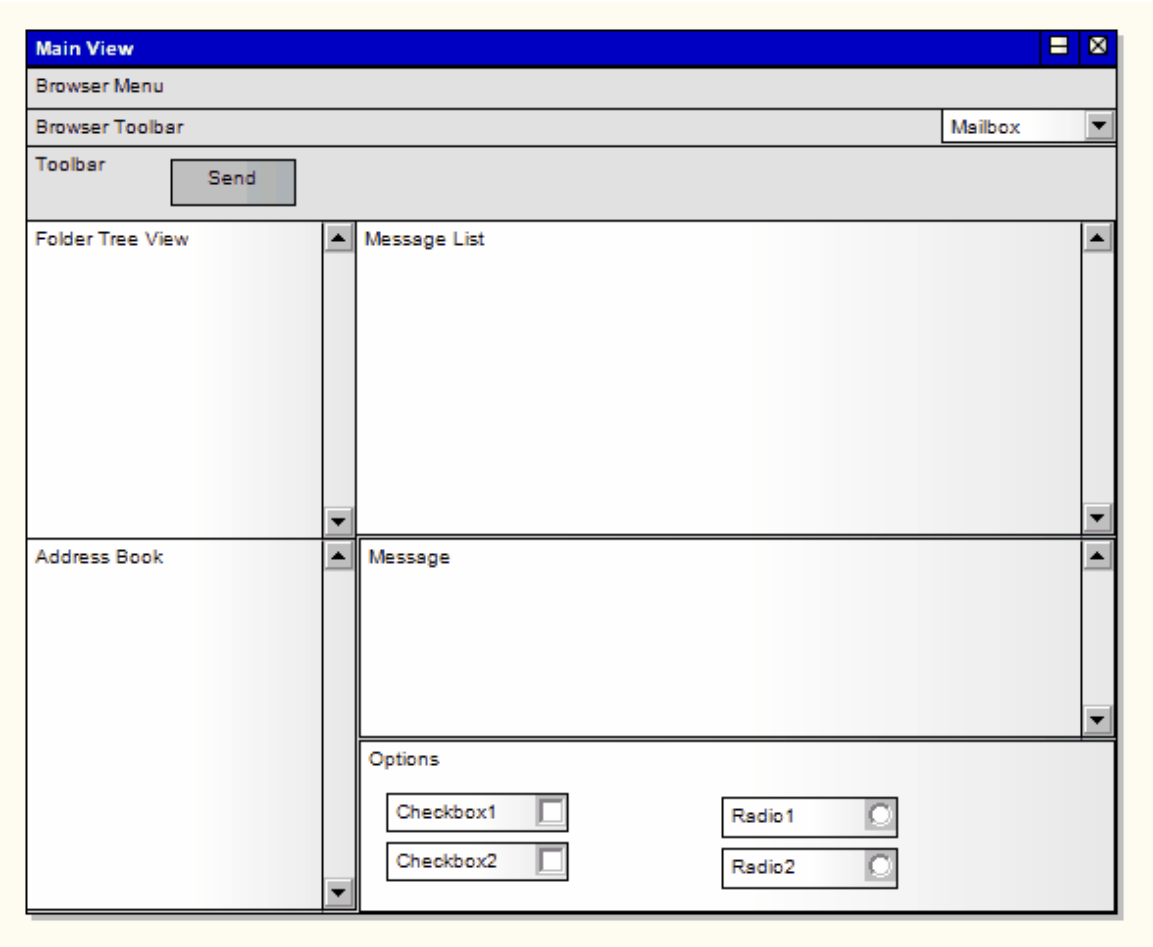


Enterprise Architect displays UI Controls as a range of special icons, depending on the stereotype used; for example, a Control stereotyped as a «list» displays with a vertical scroll bar.

Toolbox icon

Example User Interface Diagram

In this example User Interface diagram, forms, controls and labels are arranged on the diagram to define the appearance of a user interface screen and controls. UI Control elements can also be traced to other model elements linking the UI with the underlying implementation.



Notes

- The Screen element is the parent of all the UI Control elements it contains; in the Browser window, expand the

Screen element to list its child UI elements

- If you are designing more than one screen, and you want to move a UI Control element from one screen to another, you can do this in the Browser window - click on the UI Control element and drag it underneath the target Screen element; on the User Interface diagram, the UI Control element is removed from the source Screen and displayed in the target Screen

UI Control Elements

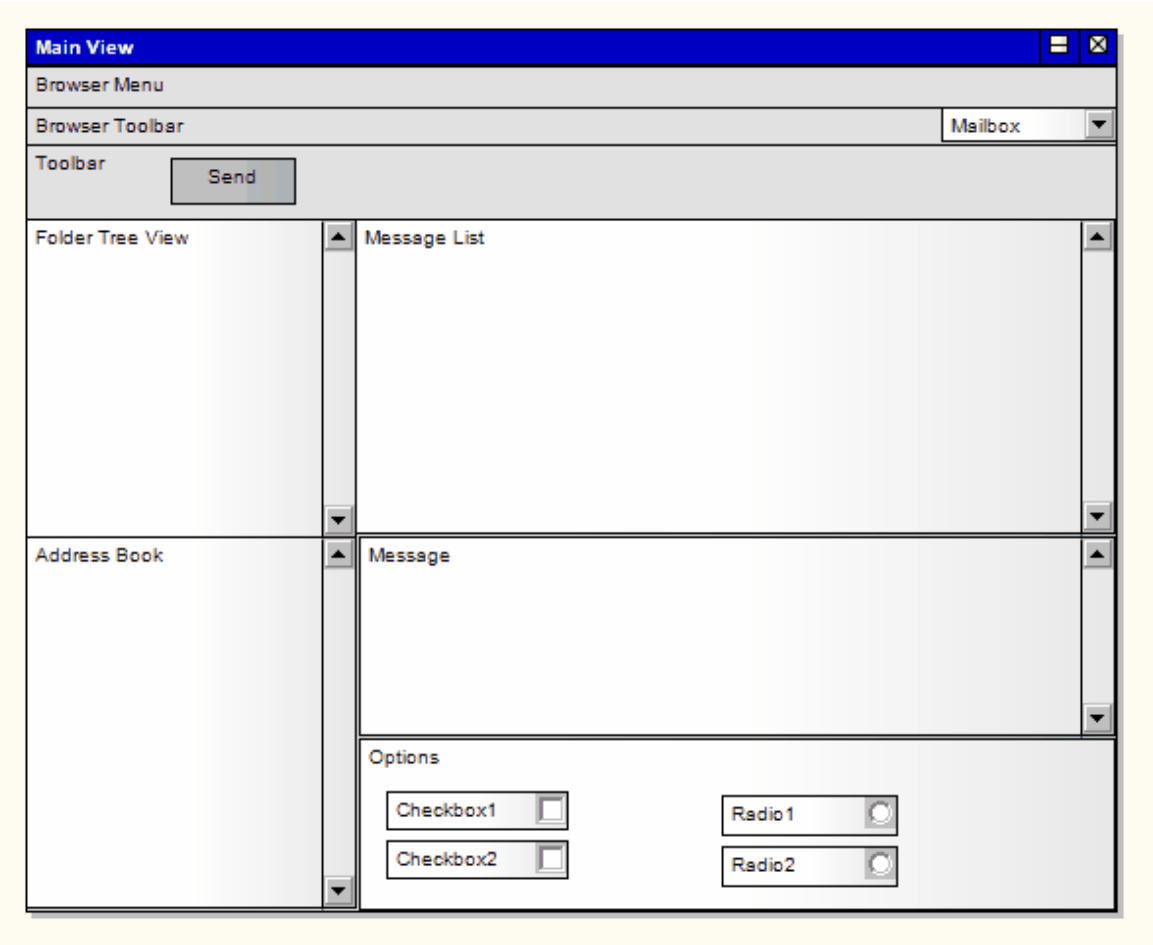
A UI Control element represents a user interface control element (such as an edit box). It is used for capturing the components of a screen layout and requirements in a Custom or User Interface diagram.

There are a number of UI Control elements available in the 'User Interface' page of the Toolbox. These include:

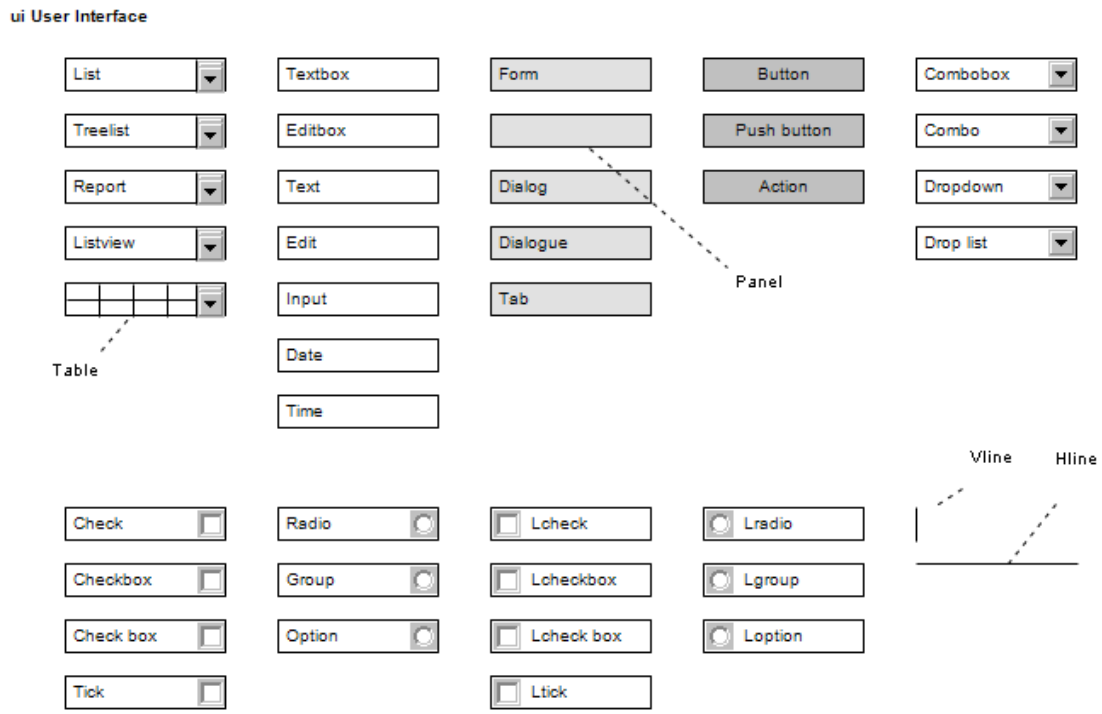
- List
- Table
- Text Box
- Label
- Form
- Panel
- Button
- Combobox
- Checkbox
- Checkbox (left hand side)
- Radio button
- Radio button (left hand side)
- Vertical Line
- Horizontal Line

Example

The icons can be combined on a 'Screen' icon to represent the appearance of a user interface screen, as shown:



You can also extend the available icons by selecting other stereotypes in the 'UI Control Element Properties' dialog. The full set of available stereotypes is shown here; type or select the text in the 'Stereotype' field to create the corresponding icon.



Toolbox icon

(where UI Control is the name of the user interface element type)