ENTERPRISE ARCHITECT

User Guide Series

Systems Modeling Language (SysML)

Author: Sparx Systems
Date: 10/11/2023
Version: 16.1

CREATED WITH ENTERPRISE ARCHITECT

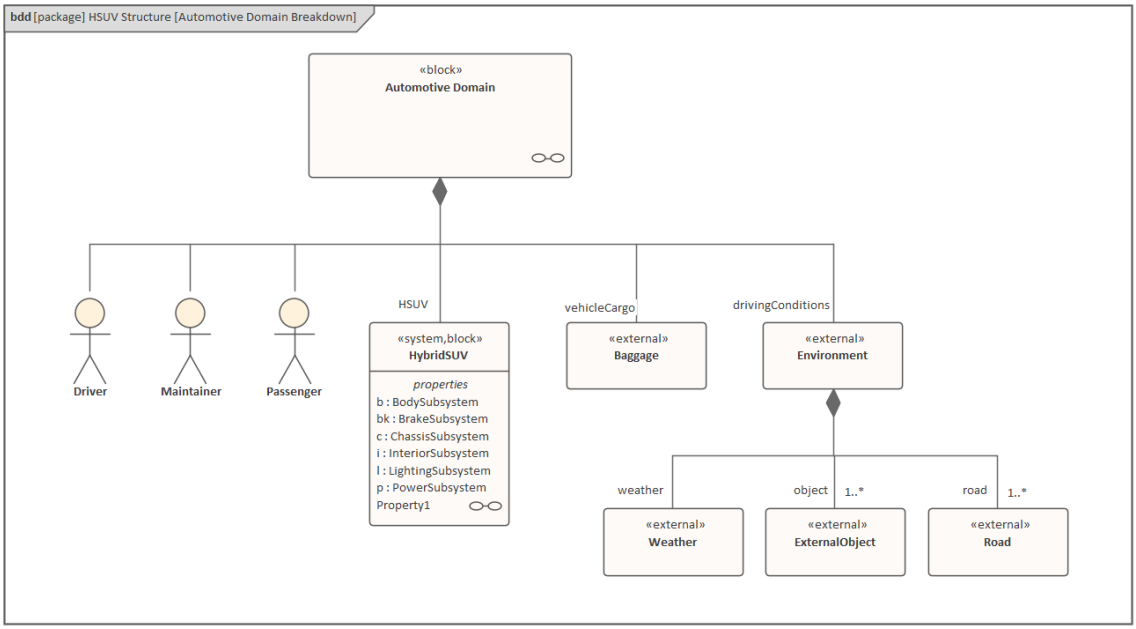
Table of Contents

Systems Modeling Language (SysML)

Enterprise Architect's implementation of SysML 1.5 delivers a multi-featured and rigorous modeling solution for Systems Engineering professionals. This integrated modeling environment helps you to:

- Specify system requirements with effective requirements modeling support
- Design deeply-nested structures of systems and subsystems using Blocks and Block diagrams
- Analyze system-to-system behavior using Interaction diagrams, Activity diagrams and State Charts
- Define system dynamics and enforce correctness with Parametric and ConstraintBlocks.

This example SysML Block diagram can be found in the Enterprise Architect Example model under Systems Engineering > SysML 1.5 Example: HSUV > Modeling Domain > HSUV Model > HSUV Structure.



Using SysML with Enterprise Architect, you can quickly and efficiently specify, design and analyze complex system models and:

- Model with all SysML 1.5 diagrams
- Conduct MDA (Model Driven Architecture) Style Transformations
- Perform simulation of SysML Parametric diagrams with OpenModelica, which supports engineering analysis of critical system parameters including the evaluation of key metrics such as performance, reliability and other physical characteristics
- Visualize and trace Requirements through to model elements throughout the entire development lifecycle
- Use the built-in Discussion Forum to create posts, access discussions and manage threads for team communication
- Use the custom Search Facility to perform complex searches, view SysML Allocations and generate reports from the results

Enterprise Architect supports all versions of SysML, from SysML 1.1 to SysML 1.5.

Access

Ribbon	Design > Package > Model Wizard : <perspective name> button > Systems Engineering > SysML --> SysML Perspective

Context Menu	Right-click on a Package > Add a Model using Wizard > Model Patterns > ... Perspective > Systems Engineering > SysML --> SysML Perspective
Keyboard Shortcuts	Ctrl+Shift+M > Model Patterns > ... Perspective > Systems Engineering > SysML --> SysML Perspective
Other	Click on the  button in the top right corner of the screen, and select the 'Systems Engineering | SysML' Perspective

SysML Integration

Facilities	Detail
SysML In Enterprise Architect	Enterprise Architect's support for SysML provides: • A range of Perspectives and Patterns to generate SysML models, in the Model Wizard (Start Page 'Create from Pattern' tab) • Patterns for each of the nine SysML diagram types, accessed through the 'New Diagram' dialog • A collection of SysML pages in the Diagram Toolbox that contain the SysML elements and relationships for each of the diagram types • SysML element and relationship entries in the 'Toolbox Shortcut Menu' and Quick Linker • A SysML-specific Glossary for the Technology
SysML Toolboxes	Enterprise Architect's support for SysML provides Diagram Toolbox pages for the nine types of SysML diagram, which you can access through the 'Find Toolbox Item' dialog. If you enable SysML as the active technology, you can also open the SysML Toolbox pages by default. See the *SysML Toolboxes* Help topic.
Working with SysML Versions	Enterprise Architect supports these SysML versions: • 1.1 • 1.2 • 1.3 • 1.4 • 1.5 However, SysML 1.5 is virtually identical to SysML 1.4 so the versions are supported and processed as the same thing. You can maintain your models under any of these versions, as necessary, but it is recommended that you only work with one version at a time and disable the others, using the 'MDG Technologies' dialog (select the 'Specialize > Technologies > Manage Technology' ribbon option). You might enable two consecutive versions if you are upgrading your models from the earlier version to the later one. **Please note:** If you select the 'SysML' perspective, you will be using SysML 1.5. If you want to use SysML 1.1, 1.2 or 1.3, select the perspective 'All Systems Engineering' or create your own custom perspective.
Upgrade SysML Models	You can migrate a SysML model (or part of a model) to a later SysML version, using the Automation Interface. See the *Migrate SysML Model to Later SysML*

	Version Help topic.

Notes

- Support for SysML is built in to the Corporate, Unified and Ultimate Editions of Enterprise Architect
- You can purchase an MDG Technology for SysML under a separate licence to use with the Professional Edition of Enterprise Architect
- Support for SysML is provided in Enterprise Architect version 12.1 or higher
- As SysML 1.5 is virtually identical to SysML 1.4, you do not need to upgrade your SysML 1.4 models; references to the latest version of SysML have, however, been updated to '1.5'

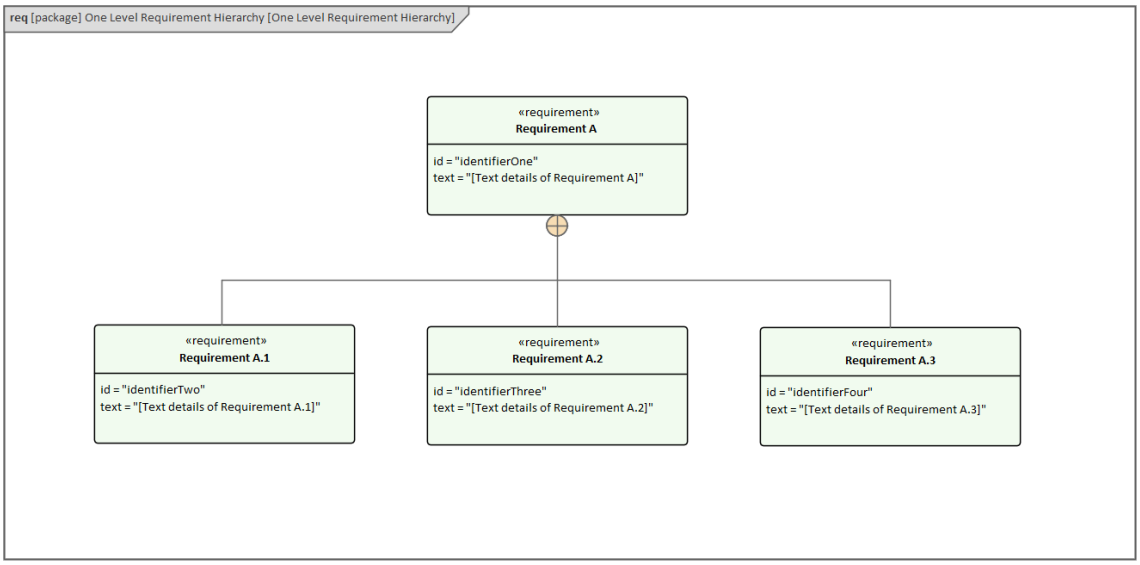
Modeling Systems in Enterprise Architect

Using SysML in Enterprise Architect, the process of developing a model to design or investigate a system is quick and easy, but at the same time versatile and flexible with a full implementation of the SysML specification. An outline of the stages of the process, and the steps for the initial stage, are provided here. Work through these steps to create a model to help engineer your system.

Create a Systems Engineering Model Framework

Follow the step-by-step guide in the table *Create a Systems Engineering Model from a Template* at the end of this topic.

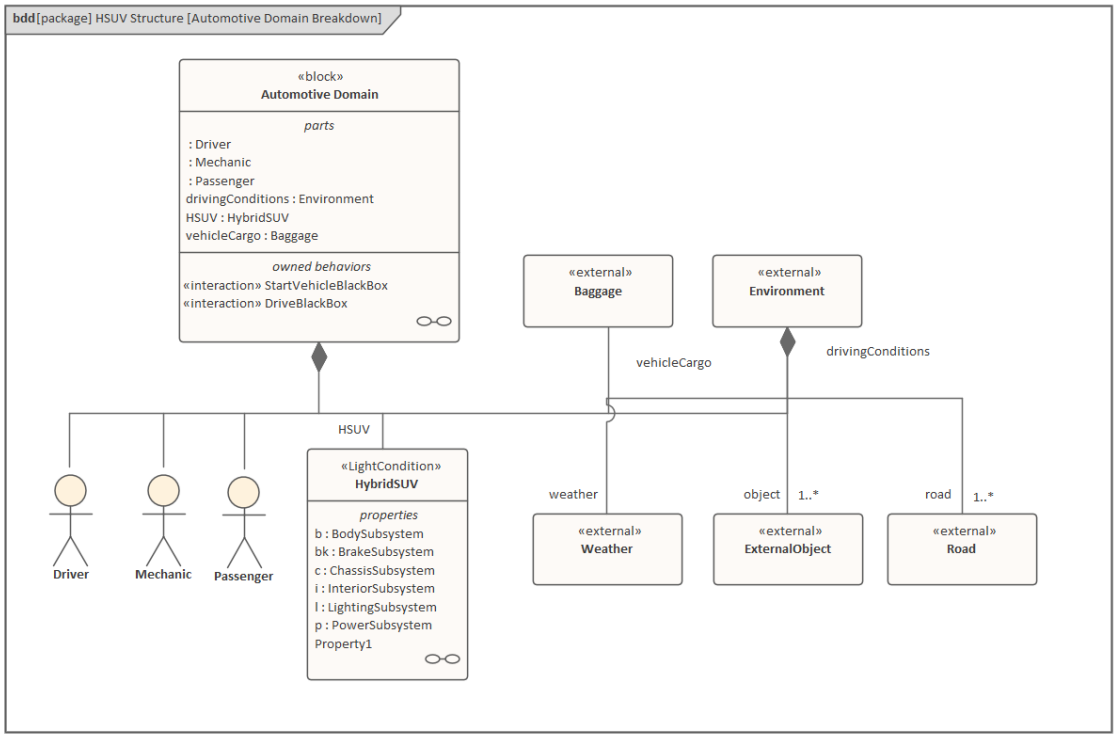
Create a Requirement Model to Define the System's Requirements and Expectations



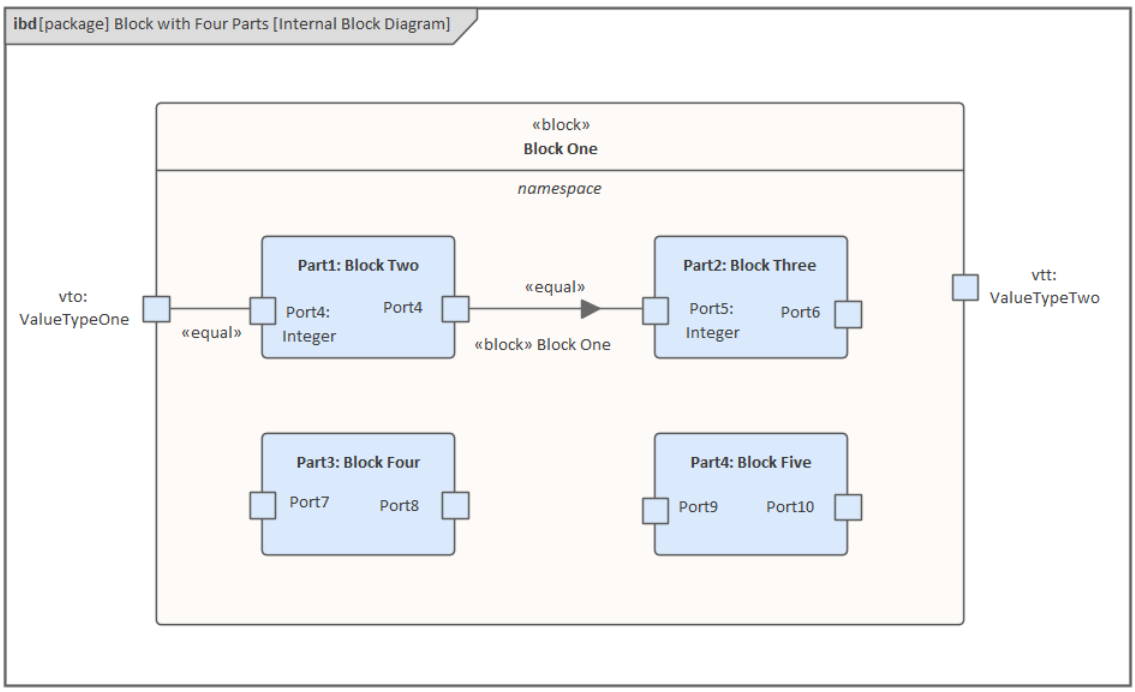
The SysML Requirement Model provides the system requirements, the expected abstract behavior, and the operating constraints that the designed system must conform to (for more information on SysML Requirement modeling, see the *SysML Requirement Modeling* Help topic).

Create an Operational Domain Model

The Operational Domain Model describes the environment that the system operates within, and the entities the system interacts with (for more information on Operational Domain models, see the *A SysML Operational Domain Model* Help topic).



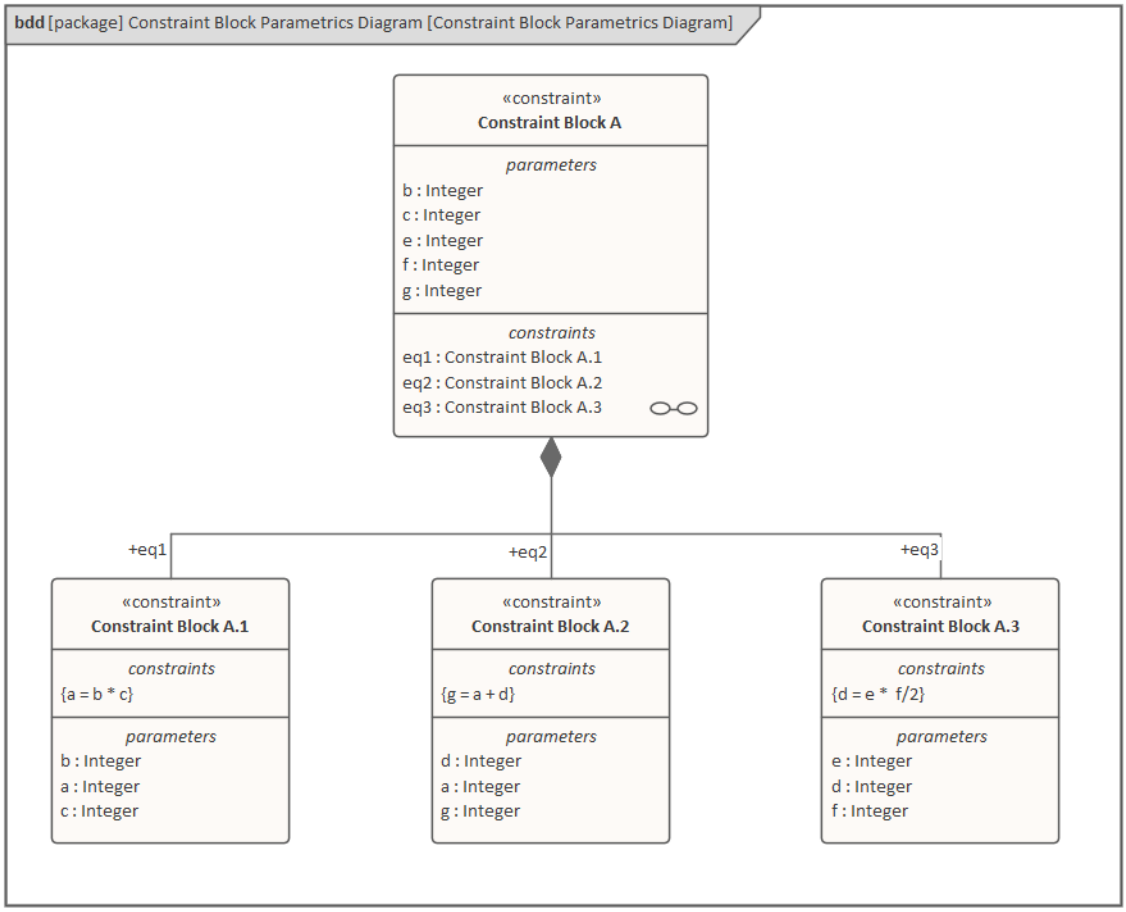
Design the System's Composition Using SysML Blocks and Parts



(For more information on the composition of a system, see the *Compose System Design* Help topic.)

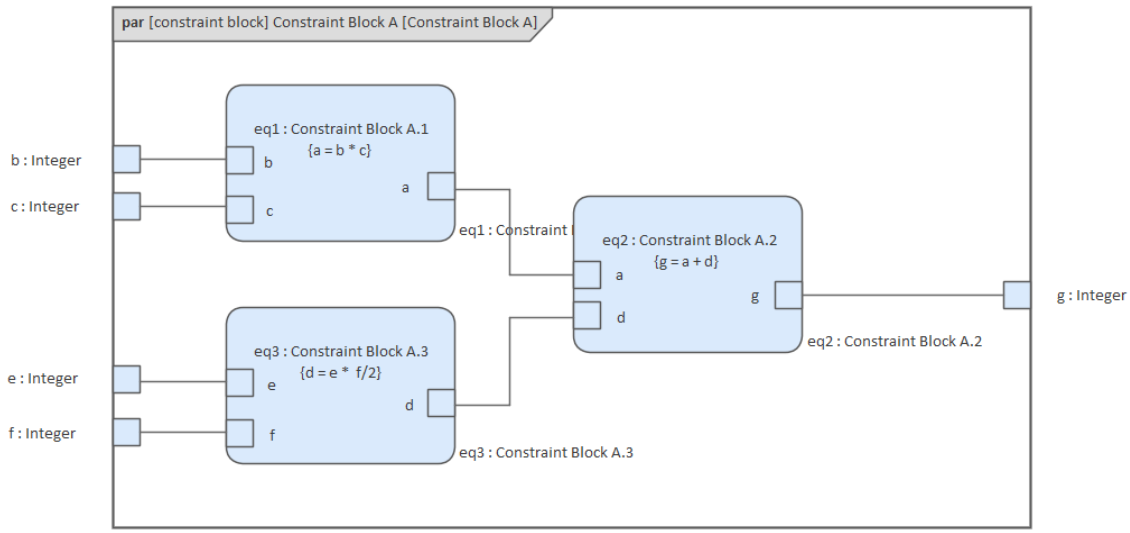
Create Constraint Models

Constraint models describe the system's operating characteristics, using Parametric models (for information on Parametric models, see the *Creating a Parametric Model* Help topic).



Simulate the Parametric Models

Simulating the Parametric models helps to verify their correctness and obtain the desired characteristics.

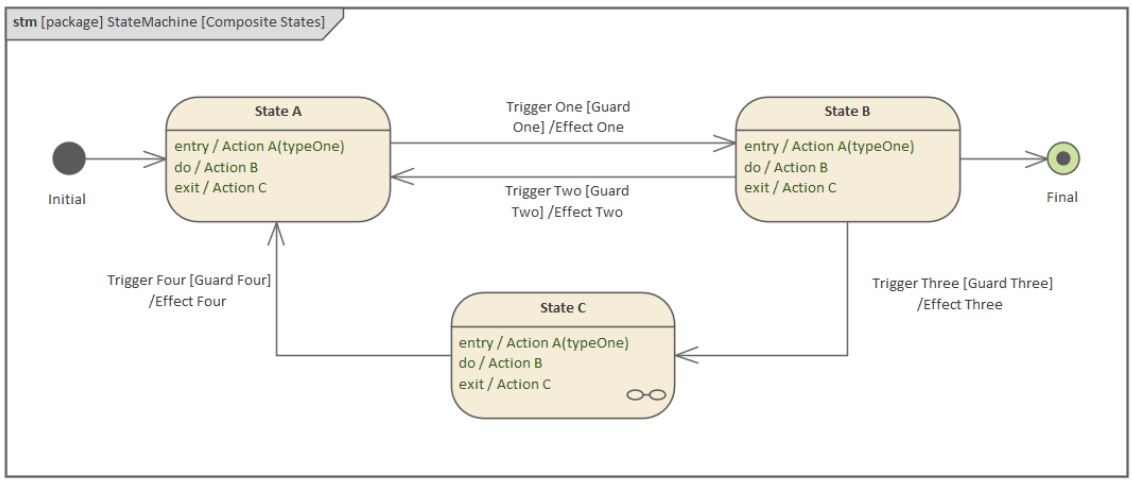


SysML Parametric models support the engineering analysis of critical system parameters, including the evaluation of key metrics such as performance, reliability and other physical characteristics (for more information on simulating

Parametric models, see the *Parametric Simulation* Help topic).

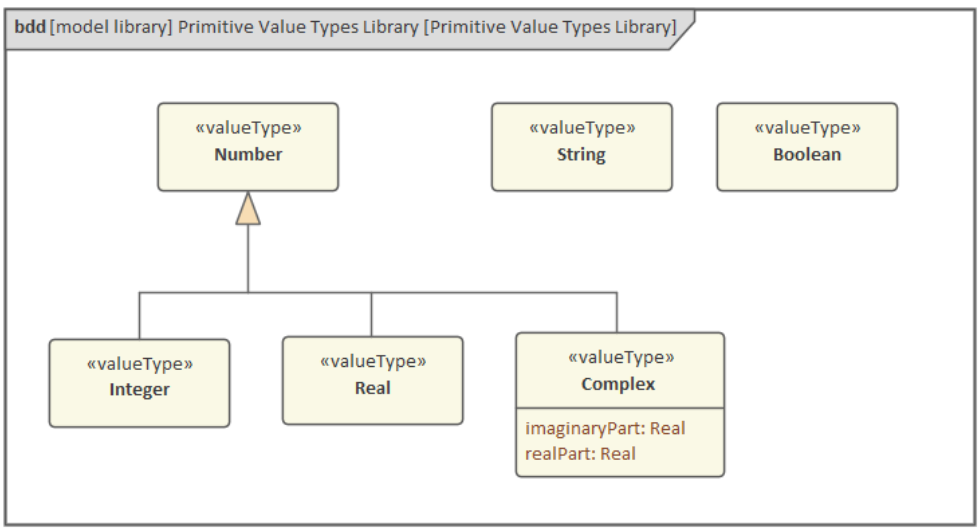
Implement the Embedded Software

You implement the embedded software using UML Classes and Behavioral models (for more information on Behavioral models, see the *Behavioral Models* Help topic).



Create a Library of Reusable SysML Blocks

The reusable SysML Blocks represent subsystems that can be reused on other projects, and other common type definitions (for more information on reusable subsystems, see the *Create Reusable Subsystems* Help topic).

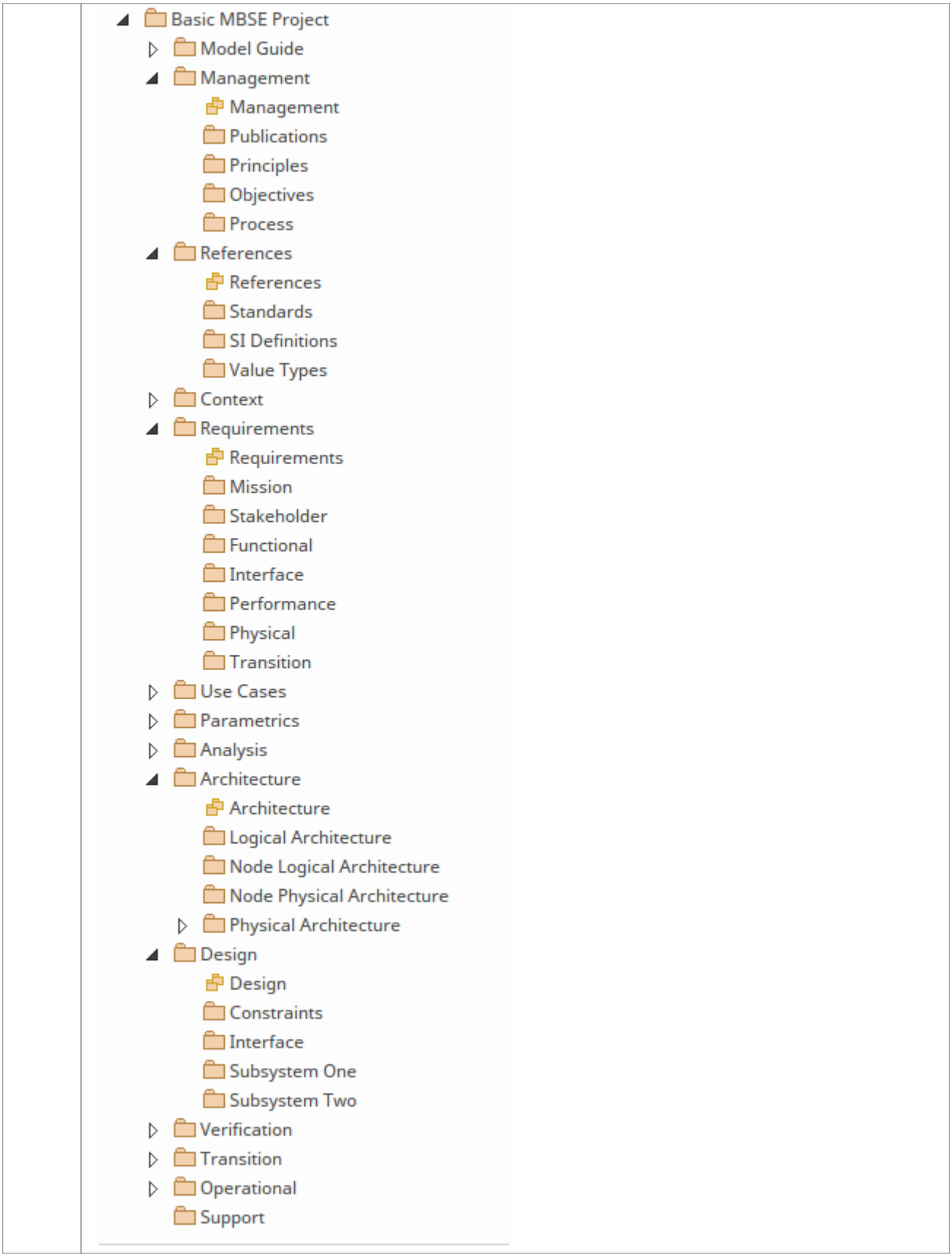


Create a Systems Engineering Model from a Template

Before starting, make sure that the Browser window is displayed (press Ctrl+1).

Step	Action

1	Click on the  icon in the top right of the screen, and from the drop-down menu choose: 'Systems Engineering | SysML'. The Model Wizard (Start Page 'Create from Pattern' tab) displays, showing model patterns from the SysML Perspective.
2	In the left hand panel, expand the 'SysML 1.n Project Structures' group and select the pattern 'Basic MBSE Project'.
3	Click on the Create Model(s) button. This model structure is created in the Browser window:

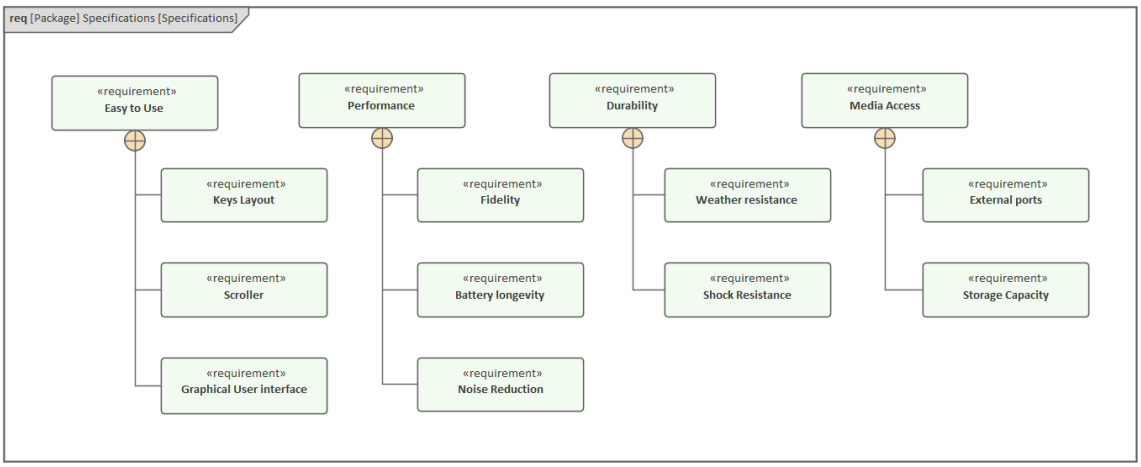

Basic MBSE Project
Model Guide
Management
Management
Publications
Principles
Objectives
Process
References
References
Standards
SI Definitions
Value Types
Context
Requirements
Requirements
Mission
Stakeholder
Functional
Interface
Performance
Physical
Transition
Use Cases
Parametrics
Analysis
Architecture
Architecture
Logical Architecture
Node Logical Architecture
Node Physical Architecture
Physical Architecture
Design
Design
Constraints
Interface
Subsystem One
Subsystem Two
Verification
Transition
Operational
Support

SysML Requirements Modeling

Requirement Engineering is a fundamental aspect of a Systems Engineering model. The discipline focuses on eliciting, analyzing and managing customer requirements early in the process. Once the requirements are understood, trade studies can be conducted to formally assess design options, typically using weighted choices. The requirements are managed as first class citizens and are formally allocated to development items and verification methods.

Enterprise Architect has extensive functionality to assist the modeler with every aspect of the Requirement Engineering discipline, including elicitation, modeling, management and testing. High quality engineering documentation can be generated out-of-the-box using a wide range of built-in templates, carefully crafted to extract the information in the models and present it in visually compelling and high quality documentation, in a wide range of formats including DOCX, PDF and HTML. The documentation engine is highly configurable, and you can generate documentation to match any engineering or organizational standard by creating templates and setting generation options.

The SysML Requirements Model provides the system requirements, the expected abstract behavior and the operating constraints that the designed system must conform to. This diagram shows an example Requirements model for a Portable Audio Player.



This example displays several top level Requirements such as 'Easy to Use' and then breaks those Requirements down into more refined Requirements such as the 'Graphical User interface'.

Building the Requirements Model

Enterprise Architect provides comprehensive support for Requirement Modeling, particularly through the Specification Manager, which you can use to: display existing model diagram elements in text format; edit sets of new requirements imported from a document, spreadsheet or Requirements Management tool; or create new Requirement elements from scratch, building them up from text entries.

You can also generate a starter SysML Requirements model from templates supplied in the 'Create from Pattern' tab (Model Wizard) (Ctrl+Shift+M). These include:

- One Level Requirement Hierarchy
- Two Level Requirement Hierarchy
- Composite Requirement Hierarchy
- Requirements Traceability

On the 'Create from Pattern' tab select the 'SysML' Perspective, then scroll to and expand the 'SysML 1.5 Requirements Diagrams' Pattern group. When you click on a Pattern name in the left hand panel of the tab, a full description of what the pattern provides is displayed in the right hand panel.

Elements

The main elements that can appear in Requirement diagrams are:

- Requirement
- Test Case

The Requirement elements contain two Tagged Values that you can set to be displayed on the diagram, using Ctrl+Shift+Y > Show Element Compartments > Tags.

- ID - the element or reference name of the Requirement
- Text - the description of the Requirement (its definition, purpose or, if at the top of a hierarchy of Requirements, the generic term for the group)

Connectors

The main connectors that can appear in Requirement diagrams are:

- Containment
- Trace
- Copy
- Derive
- Verify
- Refine
- Satisfy

Notes

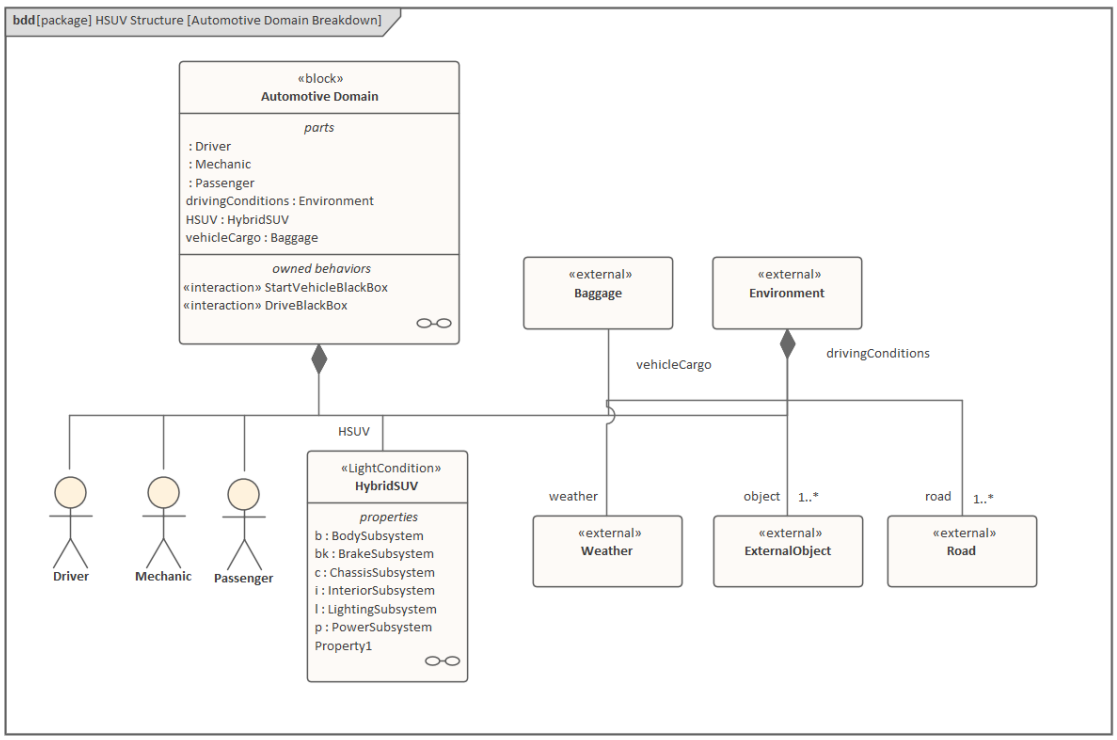
- The standard Enterprise Architect Requirement-type is interchangeable with the SysML requirements, meaning that the general Requirement Management features such as the Specification Manager, the Traceability view and Relationship Matrix can all be used in SysML Requirements Management
- With the Copy connector, when there is a value in the 'Text' Tagged Value on the target Requirement, the text is copied to the 'Text' Tagged Value on the source Requirement; the source 'Text' Tagged Value is set to read-only

A SysML Operational Domain Model

Systems Engineering is an interdisciplinary field of engineering that takes a whole-of-system view of a problem and its solution. The Operational Domain model is a central part of any model-based approach, describing the system in the context of its environment. This includes the humans that are intended to operate and interact with the system, external objects that might influence the system, and environmental elements that could impact the system. The Operational Domain model is a useful starting point to get an overview of a system and how it will operate.

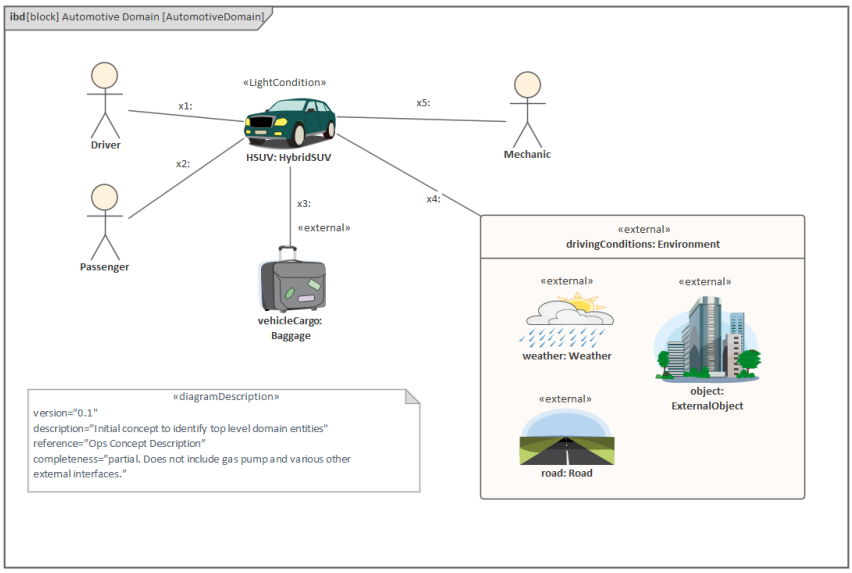
Enterprise Architect provides a range of features that help the engineer to construct an Operational Domain model, including standard SysML Block Definition diagrams (BDDs) and Internal Block diagrams (IBSs), and the ability to include pictorial representations of elements that make the diagrams more compelling. The elements can also be hyper-linked, enabling the viewer to use a diagram as a launching pad to more detailed models and diagrams.

The SysML Operational Domain model defines the system's operating environment, which describes the conditions that the system is intended to operate under. This diagram shows an example Operational Domain model for a Hybrid Sports Utility Vehicle; the SysML Block Definition diagram describes the Operational Domain (in this example - the Automotive Domain) as a system composition.



In the example, the Automotive Domain is defined as a system containing other subsystems; the domain contains subsystems that define the Driver (that is, User), the Hybrid SUV, Baggage (which is carried in the vehicle), and the External Environment.

Aspects of the Automotive Domain system are further detailed in the Automotive Domain's Internal Block diagram:



In the example, the Automotive Domain system's detailed composition shows how the Hybrid SUV and other subsystems fit together to form the Automotive Domain; it also describes the Binding relationships between the parts, which define how the parts are functionally bound to one another.

Block Definition Diagrams (BDDs)

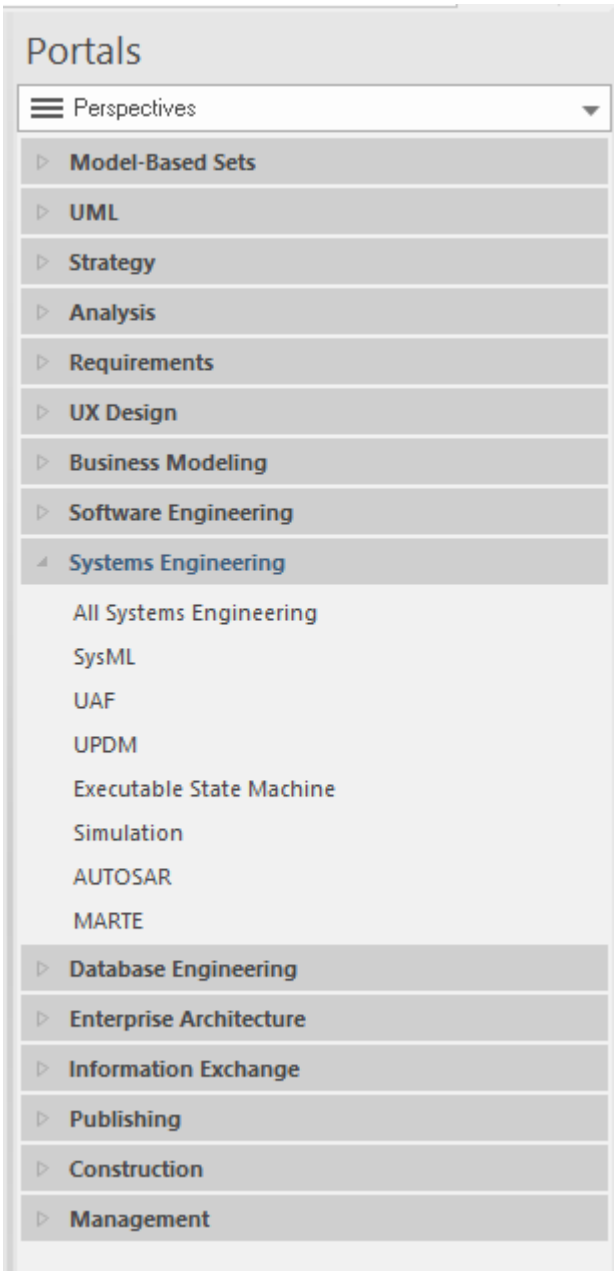
A Block defines a collection of features used to describe a system, subsystem, component or other engineering object of interest. These features can include both structural and behavioral features, such as properties, operations and receptions, that represent the state of the system and the behavior that the system might exhibit.

Getting Started with Blocks

A SysML Block Definition diagram is the starting point for describing your system structure. Using Blocks, you can model your system hierarchy and the relationships between systems and subsystems.

Setting the Perspective and Workspace

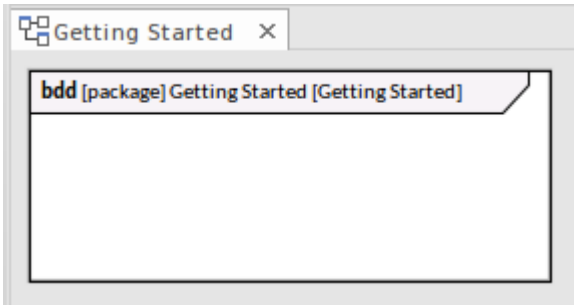
Systems Engineers who are experienced in using Enterprise Architect will generally select a perspective from the Systems Engineering Perspective Set; typically this will be the SysML perspective, giving them access to patterns and Toolbox pages tailored for creating SysML diagrams such as Block Definition and Internal Block diagrams.



Create a Block Diagram

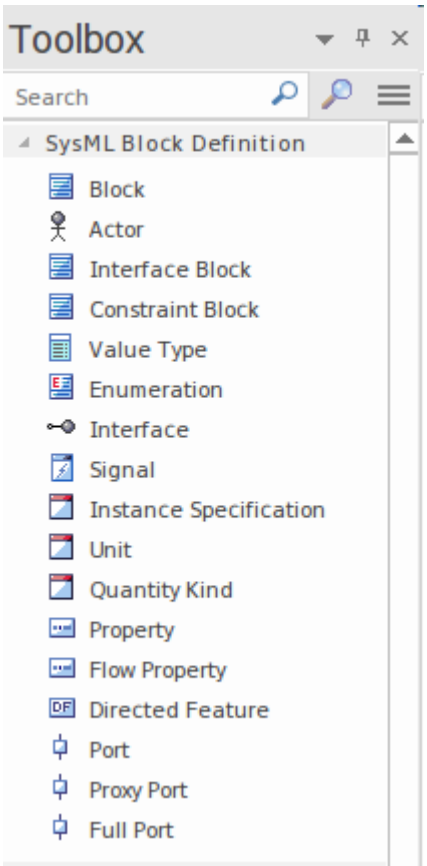
A Block diagram can be created within a selected Package using any of these options:

- The Browser window context menu (Right-click on a Package and choose 'Add Diagram')
- The 'Create from Pattern' tab (Model Wizard) on the Start Page (Ctrl+Shift+M)
- The New Diagram dialog (Ctrl+Insert)



Creating a Block Element

Block elements can be created using the 'Add Element' option on a Package context menu, or by using the SysML Block Definition toolbox to place a Block on a Block Definition diagram (BDD).



It is common for Blocks to appear on multiple BDDs, where each diagram is designed to address the concerns of a particular stakeholder or stakeholder group.

Blocks are discrete modular units that provide the foundations for system description. A Block models a collection of features that are used to define an aspect of a system or the system itself.

Block features are of two fundamental types: structural features and behavioral features - what a Block consists of and what it does.

Structural features can be further categorized into three sub-types:

- Parts - that describe the composition of a Block; for example, a vehicle is composed of two axles and four wheel assemblies
- References - that describe the Block's relationship with other Blocks (including itself); for example, that a metropolitan train has a relationship to a station and to an overhead wiring system

- Values - that describe quantifiable aspects of a Block; for example, dimensions, temperature and luminosity

Behavioral features can be subdivided into two subtypes:

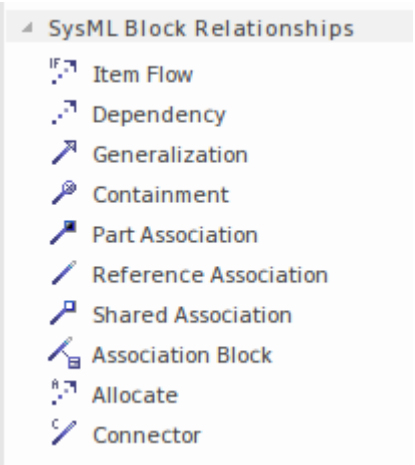
- Operations - typically representing synchronous requests
- Receptions - representing asynchronous requests

Block Relationships

A Block's relationships to itself, to other Blocks and to other types of element help to describe the structure of a system, subsystem or component.

The core Relationships used in modeling Blocks include:

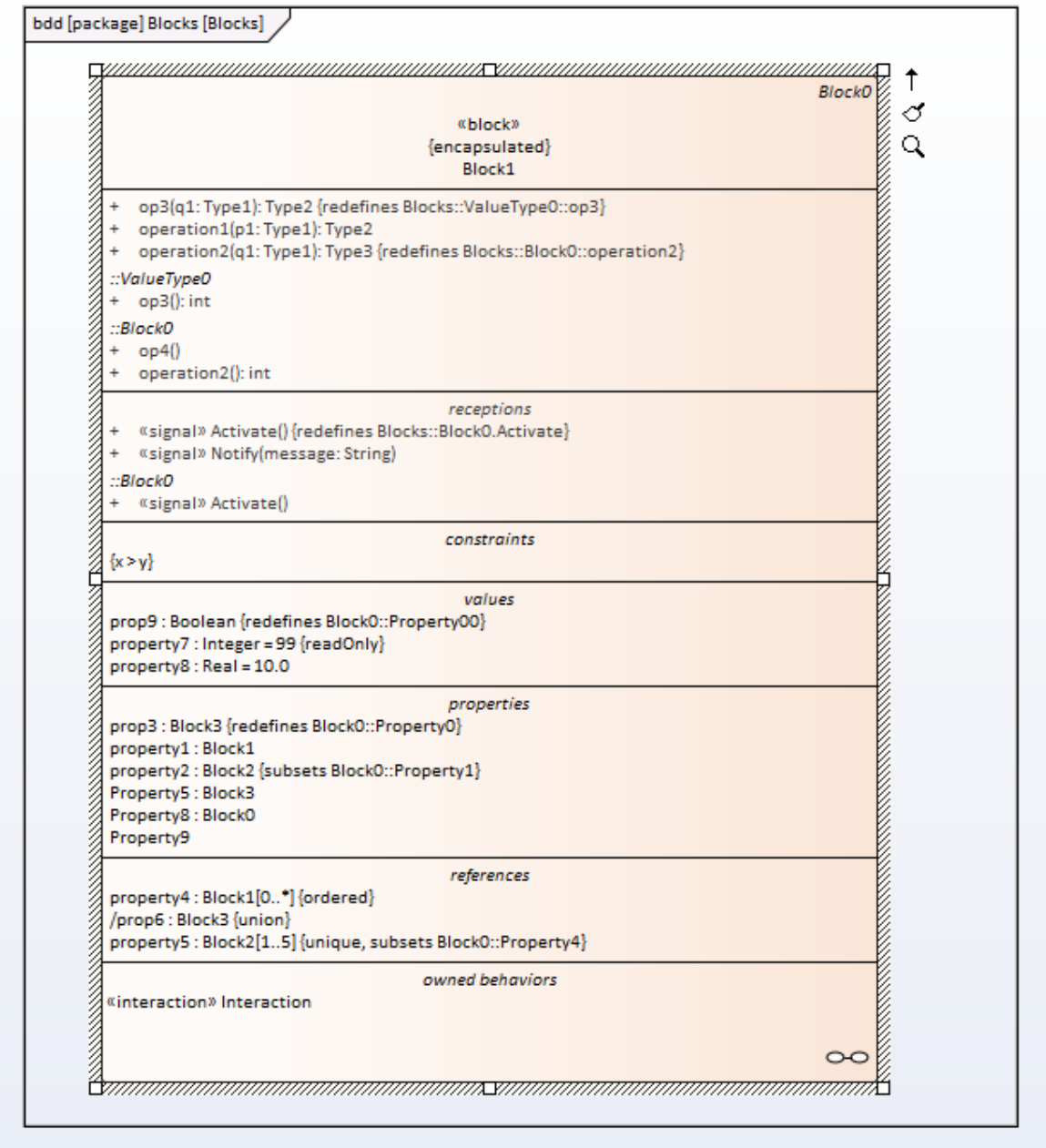
- Item Flow
- Generalization
- Part Association
- Association
- Association Block



Block Definition diagrams are often the starting point for creating other diagrams, such as Internal Block diagrams, Parametric diagrams and Activity diagrams. Features that appear on the Block Definition diagram, such as Parts and Ports, typically form the basis for modeling in these other diagrams. Enterprise Architect's *Synchronize Structural Elements* feature can be used to populate Internal Block diagrams and Parametric diagrams using information from your Block Definition diagram.

Block Element Compartments

SysML elements such as Blocks and ConstraintBlocks can show compartments that list child elements and related elements. These compartments help you to easily identify the types of property owned by a Block, and to see how other elements are linked to the Block.



Compartments that show child elements are visible by default, whereas compartments that show related elements (elements linked by connectors) are hidden by default. You can toggle between displaying and hiding each of the compartments using the:

- 'Compartment Visibility' dialog (press Ctrl+Shift+Y on the parent element on a diagram) or
- 'Compartments' tab of the docked Properties window for diagrams (Ctrl+2) or
- 'Properties' dialog for the diagram, on the 'Element' page ('Design > Diagram > Manage > Properties' ribbon option)

Note that:

- Elements are listed in compartments only if they are not already rendered as elements on the diagram

- A compartment is displayed only if at least one matching element exists for it; so, for example, a 'flowPort' compartment will be displayed only when:
  - the parent element owns at least one Port that has a «flowPort» stereotype, and
  - the Port is not on the diagram
- If a compartment is not shown, it might be necessary to locate and remove from the diagram the corresponding related or child elements, save the diagram, and reload the diagram to refresh the display of compartments

The tables *SysML Block - Child Element Compartments* and *SysML Block - Related Element Compartments* each provides a list of compartments, identifying which element/connector type has to exist in order for a given compartment to be displayed.

SysML Block - Child Element Compartments

These compartments are displayed when the Block owns one or more of the appropriate child elements, and those elements are not already rendered on the diagram.

Compartment Name	Child Element Types Listed
adjunct	Lists Parts that have the «AdjunctProperty» stereotype.
bound reference	Lists Parts that have the «BoundReference» stereotype.
classifier behavior	Identifies the behavioral classifier (Interaction, StateMachine or Activity), if set.
constraints	Lists Parts that have the «constraintProperty» stereotype.
directed features	Lists Parts that have the «DirectedFeature» stereotype.
flow ports	Lists Ports that have the «flowPort» stereotype.
flow properties	Lists Parts that have the «flowProperty» stereotype.
full ports	Lists Ports that have the «fullPort» stereotype.
owned behaviors	Lists Behavioral elements (Interactions, StateMachines and Activities) owned by this Block.
parameters	Lists Ports and Parts that have the «constraintParameter» stereotype.
participants	Lists Parts that have the «participantProperty» stereotype.
parts	Lists Properties created by adding a Part Association connector between Blocks.
ports	Lists any Ports with other stereotypes or no stereotype, not listed in the other compartments.
properties	Lists Parts that do not have a stereotype.
proxy ports	Lists Ports that have the «proxyPort» stereotype.
references	Lists Parts for which the isReference Tagged Value is set to true.
«stereotype»	Lists Parts that have a stereotype other than those identified in this table (each

	stereotype has its own compartment with the same name as the stereotype).
values	Lists Parts that are typed by a «valueType» element.

SysML Block - Related Element Compartments

These compartments are displayed based on the relationships between a Block and other elements.

Compartment Name	Object Displayed
allocatedFrom	Identifies the source element of a connector that has the «allocate» stereotype.
allocatedTo	Identifies the target element of a connector that has the «allocate» stereotype.
derived	Identifies the target element of a connector that has the «derivereqt» stereotype.
derivedFrom	Identifies the source element of a connector that has the «derivereqt» stereotype.
master	Identifies the target element of a connector that has the «copy» stereotype.
refinedBy	Identifies the source element of a connector that has the «refine» stereotype.
satisfiedBy	Identifies the source element of a connector that has the «satisfy» stereotype.
tracedTo	Identifies the source element of a connector that has the «trace» stereotype.
verifiedBy	Identifies the source element of a connector that has the «verify» stereotype.
refines	Identifies the target element of a connector that has the «refine» stereotype.
satisfies	Identifies the target element of a connector that has the «satisfy» stereotype.
tracedFrom	Identifies the target element of a connector that has the «trace» stereotype.
verifies	Identifies the target element of a connector that has the «verify» stereotype.

SysML Constraint Block Element Compartments

In addition to the compartments that a Block element can display, a Constraint Block can also display:

Compartment Name	Child Element Types Listed
parameters	Identifies any Part that does not have a stereotype (such as those without a «constraintProperty» or «objectiveFunction» stereotype; those that have the stereotype are listed in a compartment having the same name as the stereotype).

Create a Constraint Block from Equations

This feature is available from Enterprise Architect Release 14.1.

When developing an engineering solution, it is a common requirement to reflect factors determined by calculation using mathematical equations, such as Force = Mass x Acceleration (or f=m*a). The equation is represented by a Constraint, and the elements of the equation - in this case *f*, *m*, and *a* - are the parameters of the constraint.

You can model one or more calculated constraints as a SysML Constraint Block element using the 'Edit Constraint Block' dialog, through which you parse the constraints and extract the parameters from each of constraints. You can apply any equations that are appropriate to your model, whether they be international standard formulae or those you have derived yourself within the domain of your work.

Access

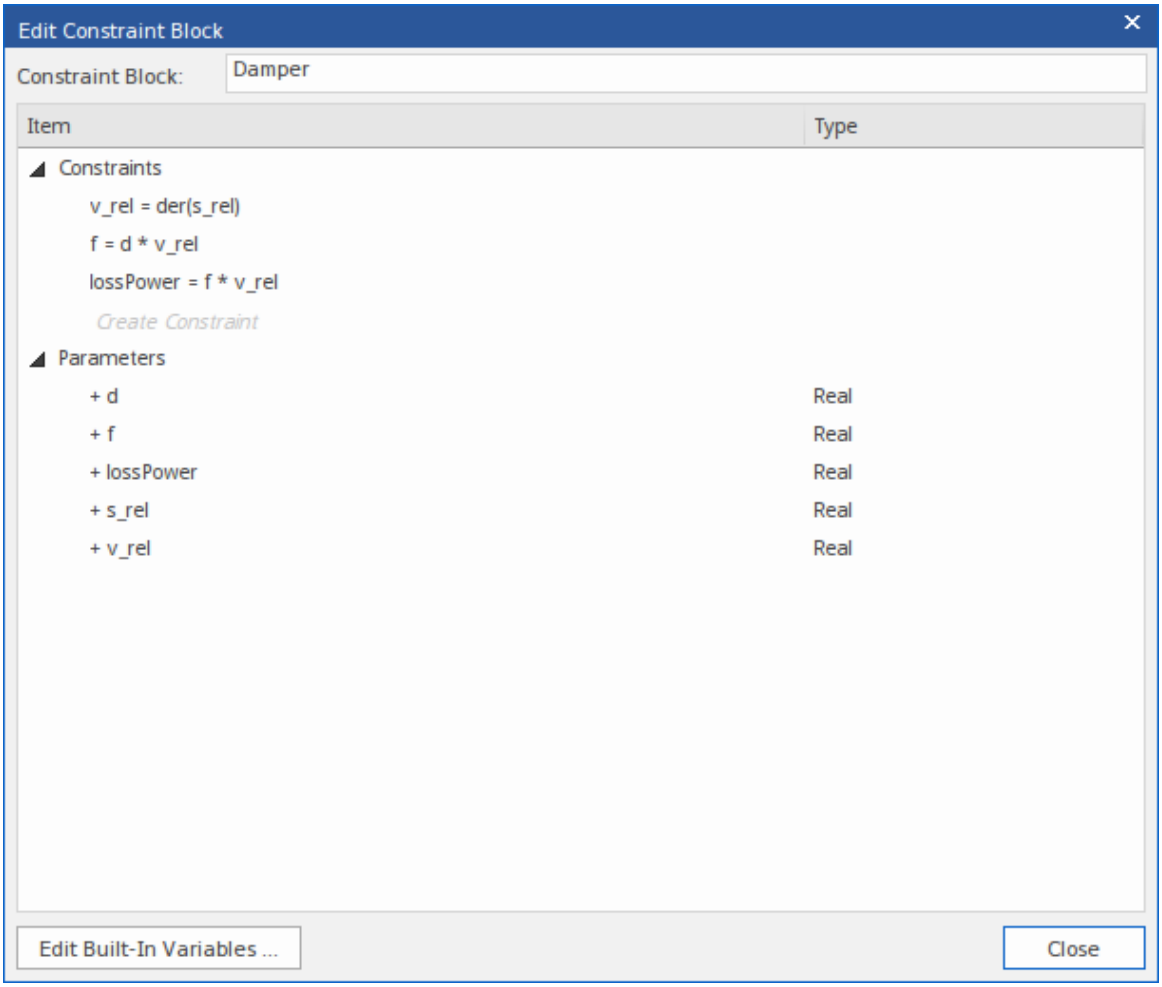
Context menu	Right-click on a Constraint Block | Edit ConstraintBlock
Other	Diagram Toolbox, SysML Block Definition page | Drag a Constraint Block icon and drop it on a Block Definition diagram

Parse Equations and create Parameters

Suppose we have a Constraint Block named 'Damper', containing these three equations as constraints:

- v_rel = der(s_rel)
- f = d * v_rel ('d' is an incorrect symbol for 'Damping Coefficient'; this is deliberate, to allow correction in a later step)
- lossPower = f * v_rel

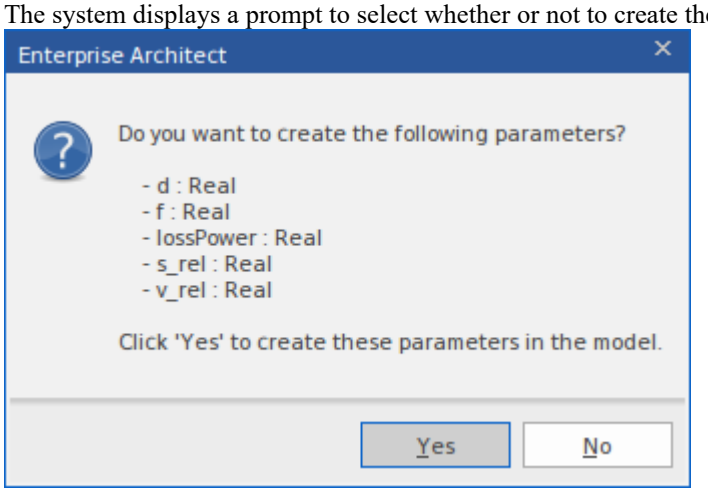
The three constraints are entered into the dialog (by overtyping the *Create Constraint* text) and from these constraints five parameters are automatically extracted.



The '+' sign preceding each parameter indicates that it does not yet exist in the model. To create the parameter in the model:

1. Click on the Close button.

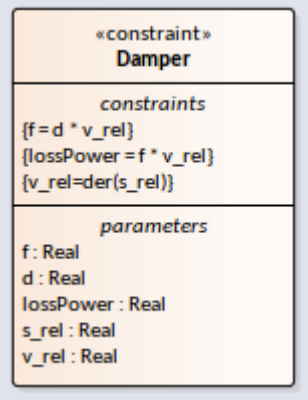
   The system displays a prompt to select whether or not to create the parameters.



2. Click on the Yes button.

Alternatively, in the 'Edit Constraint Block' dialog, you can right-click on a new parameter and choose the 'Create Parameter' context menu option. In this way, you can create a single parameter.

This image shows the resulting Constraint Block *Damper*.



Deleting Parameters

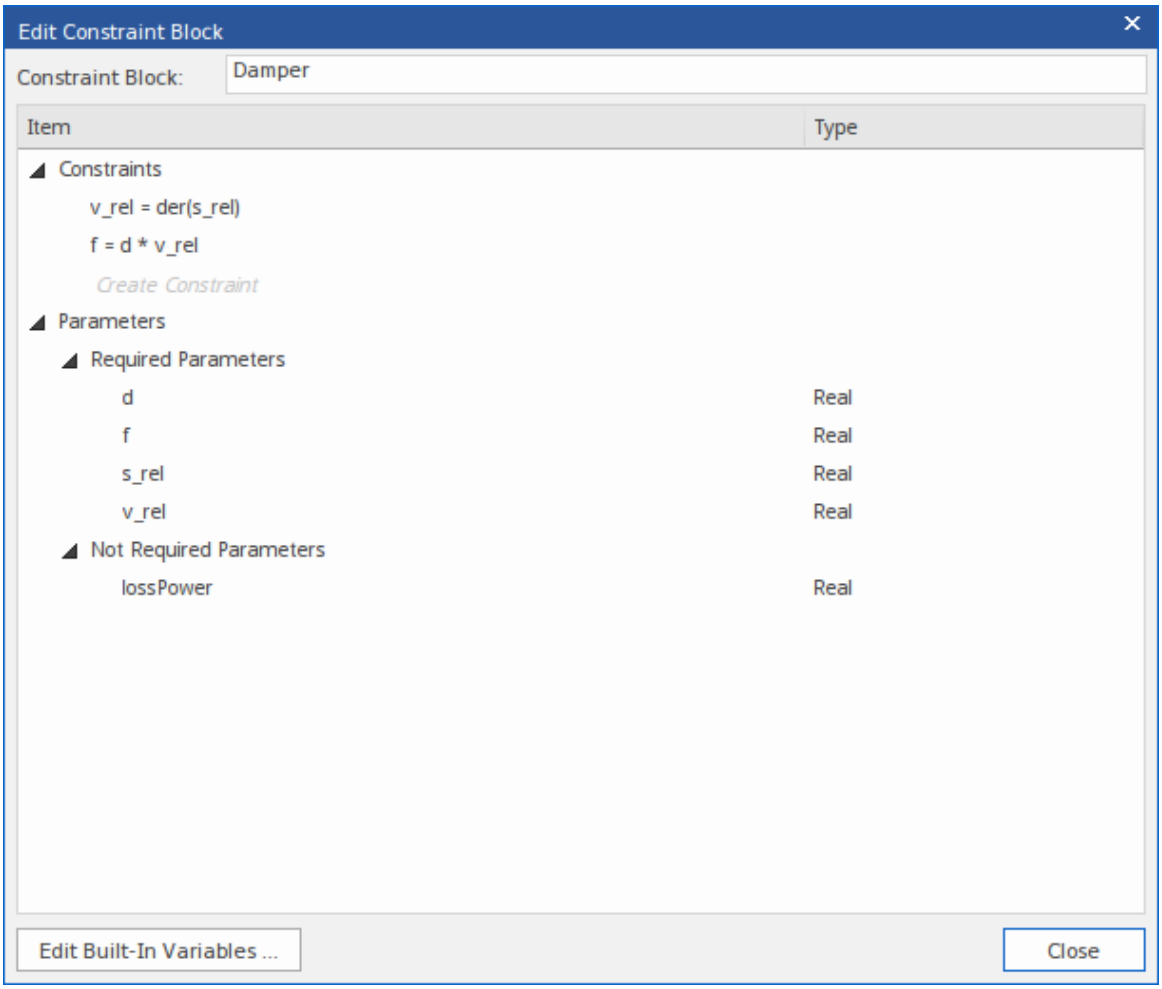
The 'Edit Constraint Block' dialog can also be used to remove a constraint and its associated parameters from a Block.

In our example, suppose we open the dialog and delete the constraint:

lossPower = f * v_rel

(Right-click on the constraint and select the 'Delete' option.)

Parameters unique to the constraint (in this case the 'lossPower' parameter) will be moved under the heading 'Not Required Parameters'.



You can now:

- Right-click on the parameter and choose the 'Delete Parameter' context menu option, or
- Right-click on the 'Not Required Parameters' heading and choose the 'Delete All Not-Required Parameters' option

Rename existing parameters

In our example, suppose we open the dialog and change the constraint:

f = d * v_rel

to

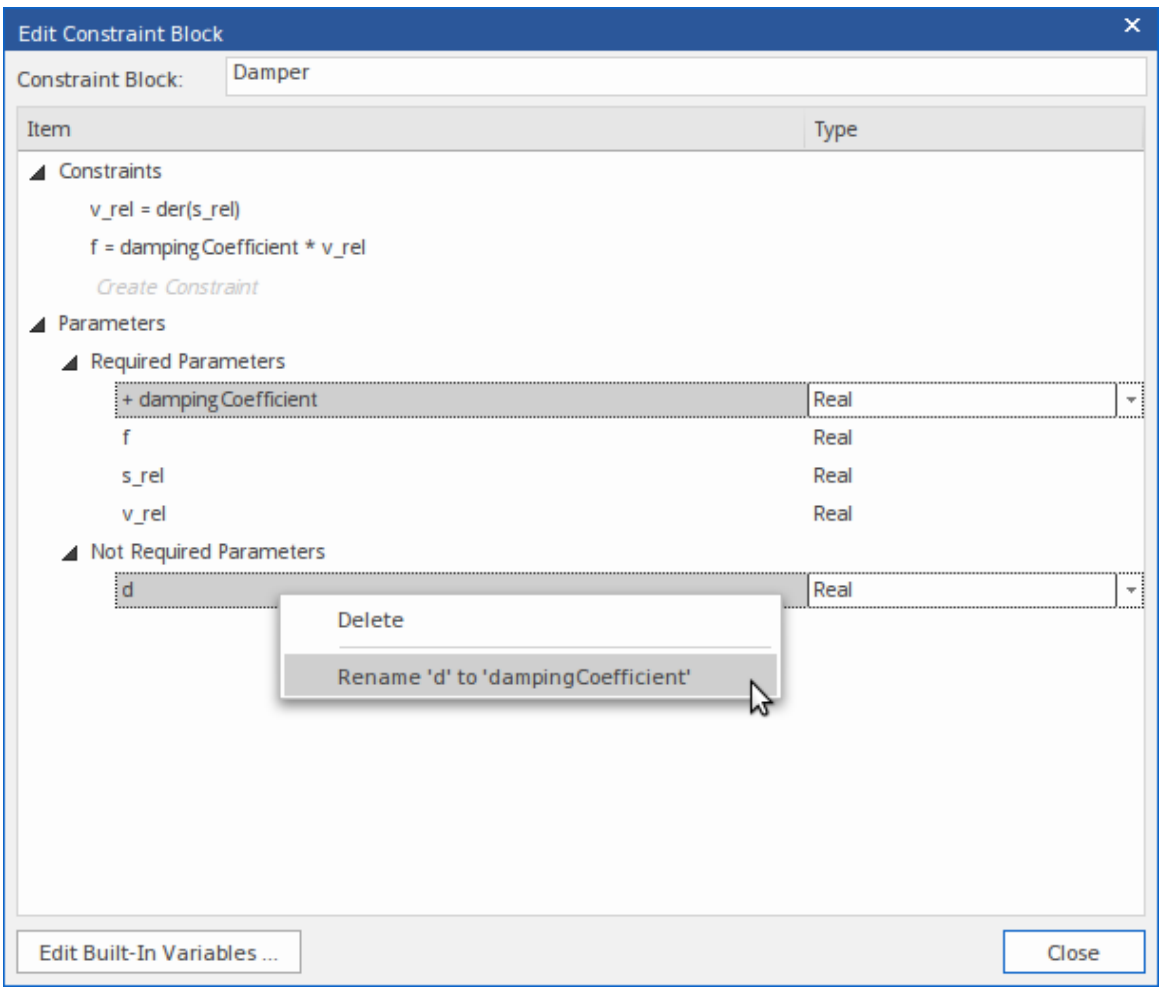
f = dampingCoefficient * v_rel

(Click on the constraint, click on the point in the constraint to begin editing, and overtype or delete the text. Then click off the constraint.)

These changes will occur:

- Parameter 'dampingCoefficient' is extracted and reported as a new parameter (it does not exist in the current model)
- Parameter 'd' is reported as 'Not Required'

We can delete 'd' and create 'dampingCoefficient', as explained earlier; however, there might be binding connectors connecting to parameter 'd' and all we want is to rename parameter 'd' to 'dampingCoefficient'. Therefore, a better solution is to right-click on the 'Not Required' parameter 'd' and choose the menu option *Rename 'd' to 'dampingCoefficient'*.



(For information on binding parameters, see the *Bind Parameters of a ConstraintProperty* Help topic.)

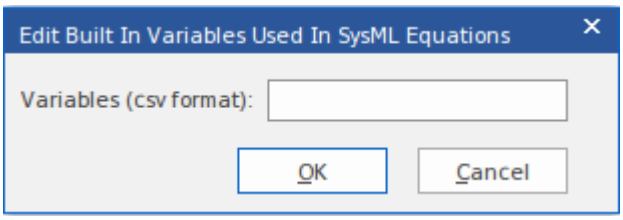
Mathematical Functions

The equation parser supports the use of mathematical functions (such as *der(s_rel)*, earlier in this topic) within your constraint equations. When specifying a function, there should be no spaces between the function name and the opening parenthesis. The function parameters will be extracted as new constraint parameters, but the function name will not.

Built-In Variables

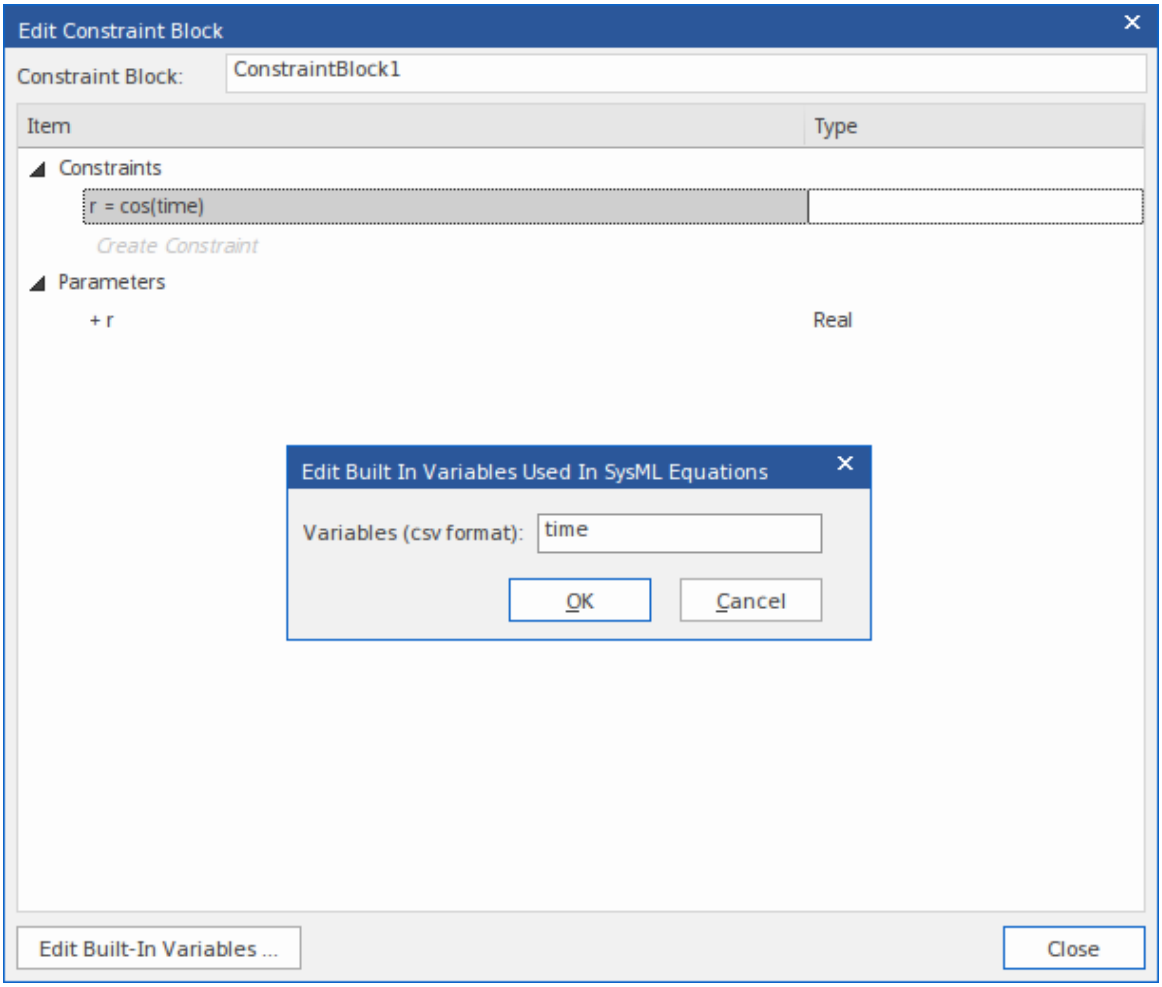
An equation could contain variables that you prefer not to extract as constraint parameters. For instance, the simulation environment OpenModelica provides a number of built-in variables, so where the model is to be simulated under OpenModelica you would not want to duplicate those variables as extracted parameters. You can identify the built-in variables to the 'Edit ConstraintBlock' functionality, so that they are **not** extracted from your constraint equations, thus avoiding potential conflicts.

To define a list of variables that should not be extracted as parameters, on the 'Edit ConstraintBlock' dialog click on the Edit Built-In Variables... button.



Then enter or add to the comma-separated list of built-in variable names.

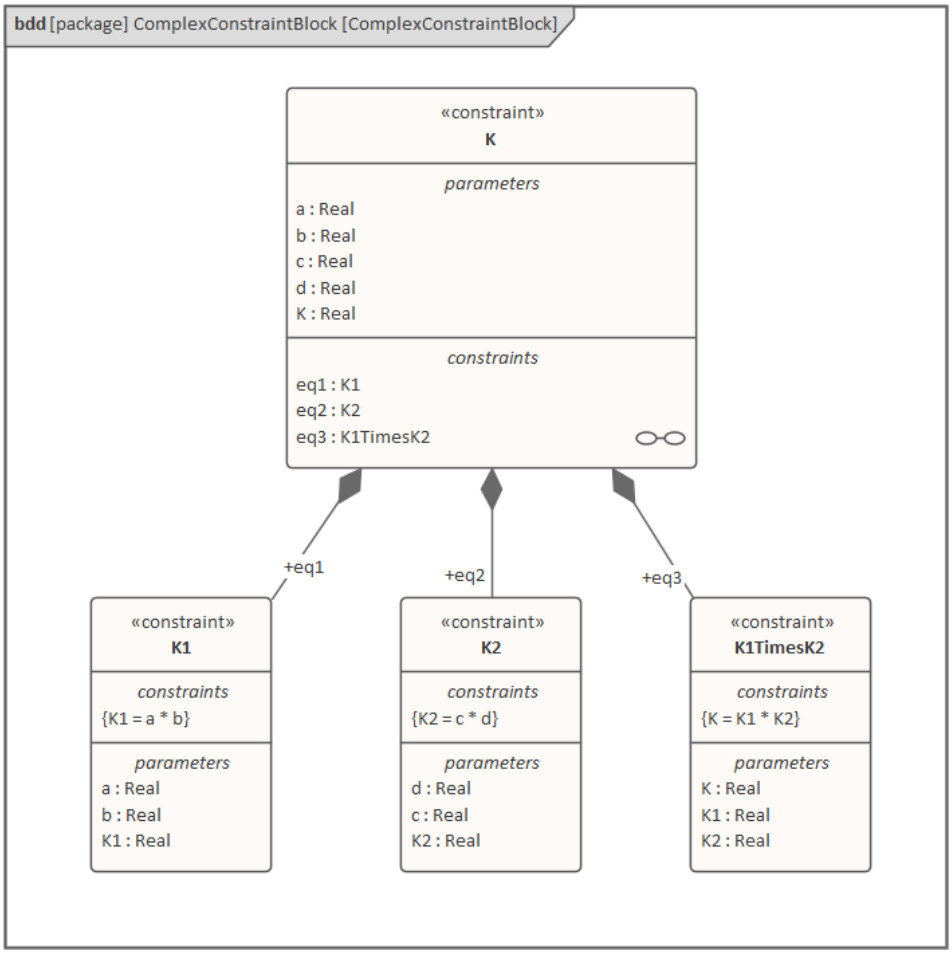
For example, OpenModelica defines 'time' as a built-in variable, so we would add 'time' to the list for the ConstraintBlocks. Then when we enter the equation 'r = cos(time)' only the parameter 'r' will be extracted.



Composite ConstraintBlock

In the development of more complex and/or sequential calculated constraints, you can construct composite ConstraintBlocks to break down and sequence the calculations.

In this example, the ConstraintBlock *K* does not itself define any constraints, but as a composite inherits them from its three component ConstraintBlocks *K1*, *K2* and *K1TimesK2*. ConstraintBlock *K* inherits the five parameters a, b, c, d and K.



Rather than extract the value of K from one calculation, the system will first calculate K1, then K2, and finally the product of K1 and K2, giving the value of K.

This structure also makes it easier to modify the input of certain elements of the calculation without affecting the other elements.

Creating Ports and Parts

The set of features that a Block element defines can include Port and Part (or Property) elements. When you initially create a Port or Part on a Block Definition diagram it is rendered as an object on its parent Block element, but the object is usually then removed from the diagram and represented by a text string in a labeled compartment of the Block.

Access

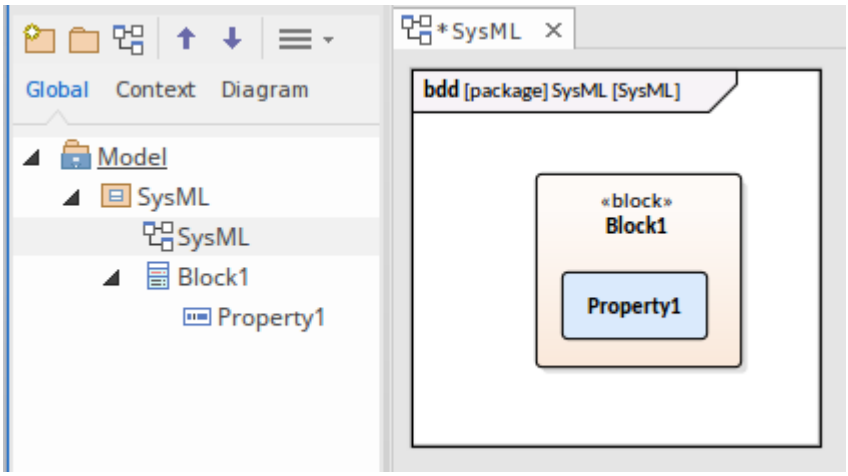
Other	Select or create the required Block Definition diagram, which will open the 'SysML Block Definition' pages of the Diagram Toolbox. Select or create the appropriate Block element in the diagram.

Creating Ports and Parts from the Toolbox

To create a Port or Part:

1. Click on the 'Port' or 'Property' icon in the SysML Block Definition Toolbox.
2. Click within a Block on the diagram.

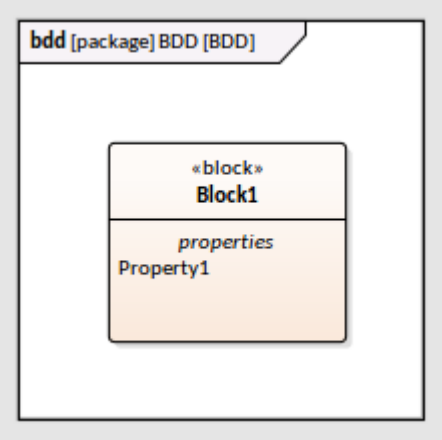
The new element is depicted as a Port or Property object on the diagram. Use the Browser window to confirm that the object has been created as a child of the Block element.



You can leave the object rendered as a graphic on the diagram, or you can remove it from the diagram and reference it as text in a compartment of the Block element.

To represent the object as text in a compartment:

1. Select the Property/Port in the diagram.
2. Press the Delete key to remove the object from the diagram.
   The name of the Property or Port will immediately appear in the appropriate compartment of the Block element.

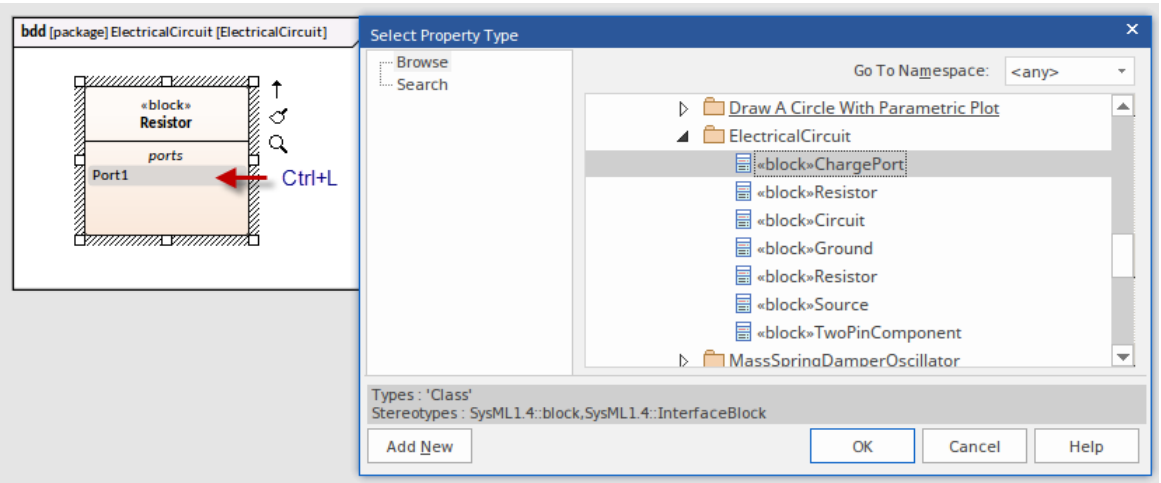


Specifying the Type of a Port or Part

You might need to set a classifier as the Type of a Port or Part. To do this, you display the 'Select Property Type' dialog and browse or search for the appropriate classifier.

To display the 'Select Property Type' dialog, either:

- Click on the object element or the object name in the compartment and press Ctrl+L, or
- Right-click on the object in the diagram, and select the 'Advanced | Set Property Type...' context menu option

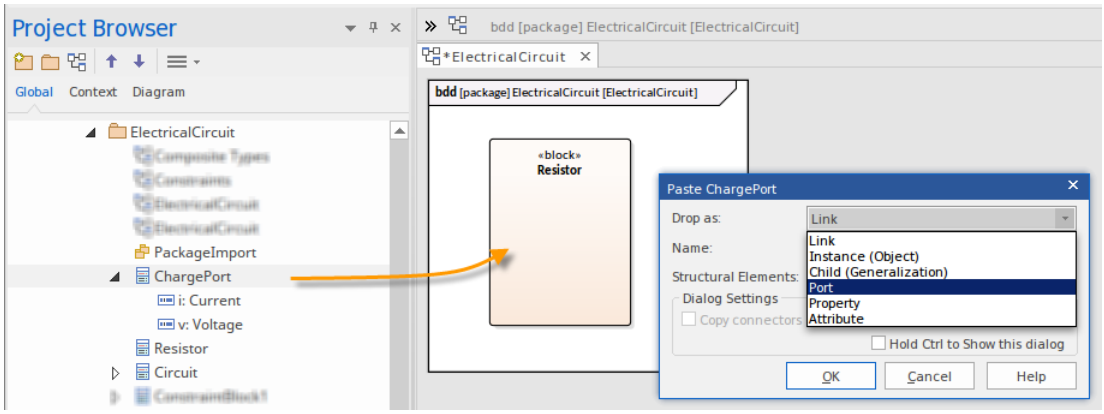


Creating a Part or Port as an Instance of an Existing Block

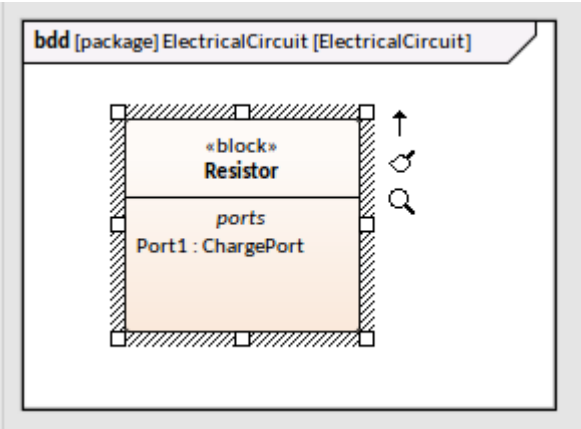
Where you have existing Block definitions in the model, you can create an instance of one of those Blocks as a Part or Port within another Block.

With the parent Block showing in a diagram:

1. In the Browser window, select the Block that will be used as the Part or Port.
2. Press the Ctrl key and drag that Block from the Browser window, dropping it onto the parent Block on the diagram.



3. In the 'Paste item' dialog, click on the 'Drop as' drop-down arrow and select the required option (a 'Port' in the illustration) from the list.
4. Click on the OK button.
5. The new Port is created within the parent Block, as an instance of the dragged Block.
6. If you prefer to set the Port to show as text in a compartment, click on the Port and press the Delete key.



Generate Parts From Block Associations

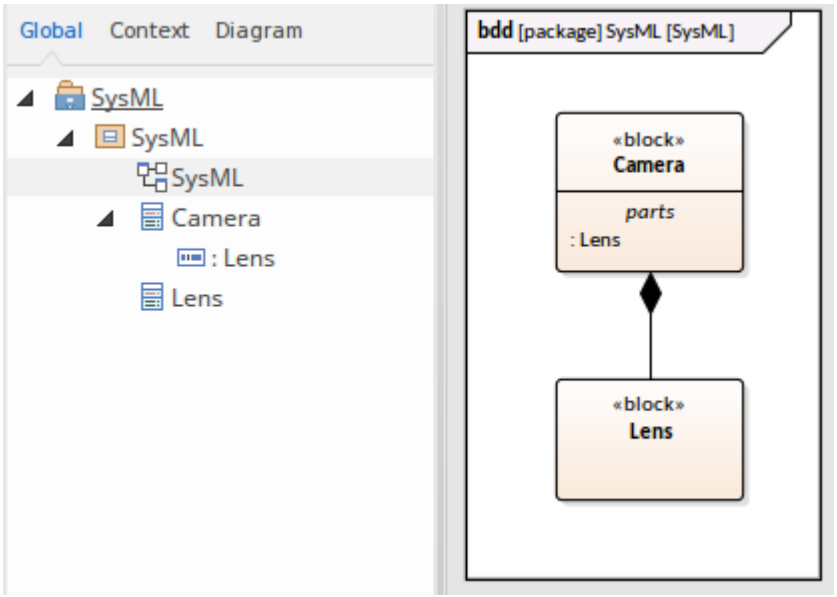
On a SysML Block diagram, the ends of an Association relationship between two Block elements can represent SysML Properties. If an Association End is navigable, the Property that it represents is owned by the Block element at the other end of the Association.

In Enterprise Architect you can automatically generate Part elements from the Association Ends to more visibly represent these owned Properties, using any of the methods described here. The Part is bound to the Association End – they represent the same Property, so changing one updates the other, either automatically or at the next synchronization; that is, if you change the Association Source Role name, multiplicity or Aggregation setting, the Part name, multiplicity and isReference setting are updated; if you change the Part details, the Association End properties are updated.

Generate from Part Association

Click on the 'Part Association' icon in the Diagram Toolbox and drag the cursor between two Block elements.

An anonymous Part property is generated on the target Block element.

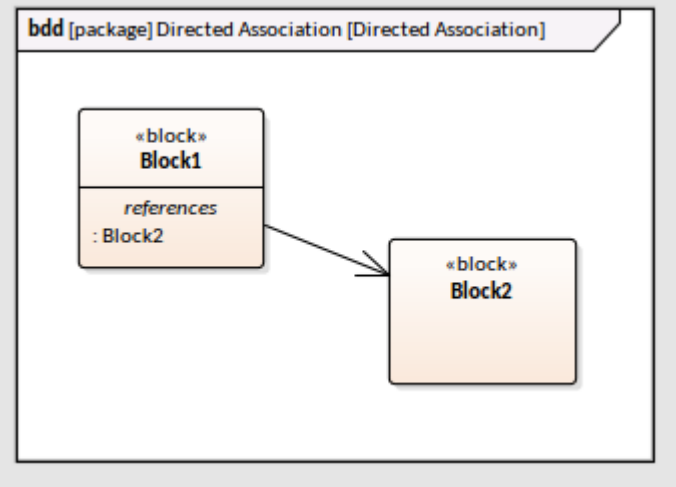


Generate from Directed Association

Firstly, ensure that Associations you create will default to 'directed' (select the 'Start > Appearance > Preferences > Preferences > Links' option and select the 'Association default = source --> target' checkbox).

Create an Association relationship between two Block elements, using either the 'Reference Association' icon in the Diagram Toolbox or dragging the Quick Linker arrow and selecting 'Association'.

An anonymous Reference property is generated on the source Block element.



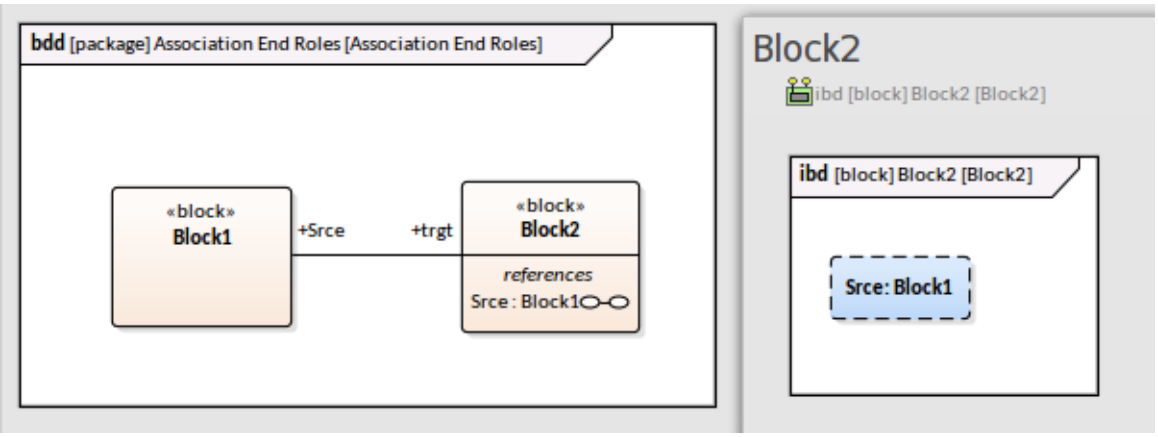
Generate from Internal Block Diagram

Create an Association between two Blocks and give one of the Association end roles a name.

Open the Internal Block diagram for the Block at the opposite end of the connector, right-click on it and select the 'Synchronize Structural Elements' option.

The property is generated on both that Block and its Internal Block diagram.

For example, name the target end role, open the Internal Block diagram of the source element, and select the 'Synchronize...' option to create the property on both the source Block and its IBD.



Change Property-Association Binding

You can, if necessary, change the binding of a Property to an Association End, or bind existing Association Ends and Properties that are not yet bound to each other.

Right-click on the Property in:

- The Browser window, and select the 'Add | Bind to Connector Role' option, or
- The Internal Block diagram and select the 'Advanced | Bind to Connector Role' option

In each case the 'Choose Connector Role to Bind' dialog displays, listing the Associations issuing from the parent Block element.

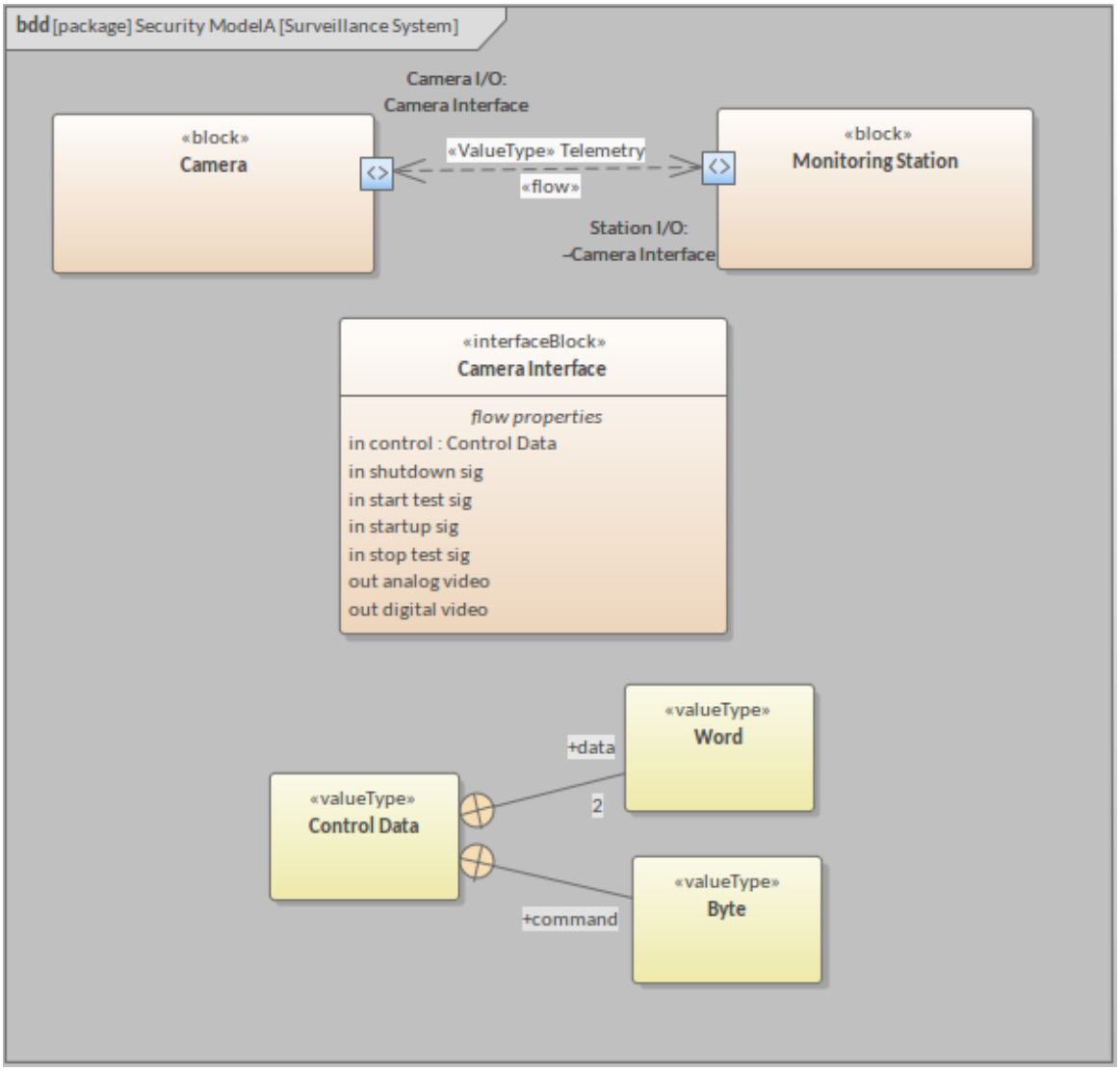
- Select the Association to bind the Property to
- Click on the OK button

If you subsequently delete an Association that is bound to a Property, when you save the diagram you are prompted to confirm whether to also delete the Property or keep it, unbound to a connector.

If the Property element is locked, it cannot be deleted.

Show Direction on SysML Ports

In SysML Block Definition diagrams you can represent the direction of flow through Ports, Full Ports and Proxy Ports by generating direction arrows on the Ports, as illustrated by the 'Camera I/O' and 'Station I/O' Ports in this Block diagram.



The direction of flow is defined by one or more Flow Property elements contained in an Interface Block element that 'types' the Port. (The direction is set in the 'direction' Tagged Value of each Flow Property element.) Where one Port references several Flow Properties, the direction arrows reflect all of the direction values used in the properties. For example, in the diagram, some Flow Properties have the direction 'in' and some 'out', so the Port displays both 'in' and 'out' arrows.

Ports that exchange the Flow Property items are usually in reciprocal pairs. That is, the items flow out from one Port and in to another. Both Ports would reference the same Interface Block and Flow Properties, but the flow through one would have to be set to the inverse of the defined direction(s). You do this by selecting that Port's 'Conjugated' property. When this is done, on the diagram a tilde (~) is displayed against the Port's reference to the Interface Block name. On the example diagram, this is shown in front of 'Camera Interface' under the 'Station I/O' Port name, indicating that the Port references the Flow Properties of Camera Interface but with the inverse direction values.

Each FlowProperty identifies something that is passed into or out of an element. You can have a single Flow Property for a discrete object that is passed through the Port, such as 'electric current', or a number of Flow Properties to identify the components of a package of items, such as data items, as shown in the diagram.

The item identified by a FlowProperty is defined through a series of Value Type elements and Unit and Quantity Type objects that are ultimately derived from your SysML Model Library (see the *Creating Model Library Definitions* Help

topic). On our diagram, this is illustrated by the Control Data, Word and Byte elements that are used to define the 'control' FlowProperty; the definition of each of the other FlowProperties in the diagram would be drawn from a similar arrangement of Value Type elements.

If the Block element has an Internal Block diagram or Parametric diagram, the direction arrows on the Port are automatically shown in that diagram.

Add direction arrows to a Port

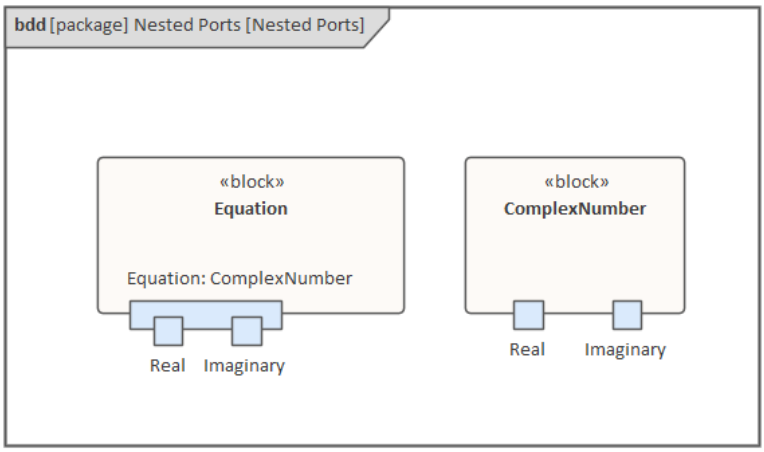
Step	Action
1	In necessary, on a Block Definition diagram create the Port, Full Port or Proxy Port element on a Block element, using the 'SysML Block Definition' pages of the Diagram toolbox. Do this for as many Block elements and/or Ports as are required.
2	If necessary, create an Interface Block either on the same diagram or on a convenient reference data diagram, again using the 'SysML Block Definition' pages of the Diagram toolbox. From the same pages, drag a Flow Property element onto the Interface Block for each object or quantity that is passing through a given Port.
3	For each Flow Property, in the Properties window type a relevant name for the element, and in the 'Tags' tab of the window set the 'direction' tag to the appropriate value. Also, as required, define the parameters of the property by first setting the type (press Ctrl+L and select the appropriate Value Type element) and quantity type and unit (set the Tagged Values in the 'Tags' tab).
4	For each Port in turn, click on the Port and press Ctrl+L, then from the Select Property Type window select the Interface Block element containing the required Flow Properties. Notice that the Port now displays one or two arrowhead symbols (<,>) reflecting the directions defined on the Flow Property elements (assuming they are not all set to <none>).
5	If you need to set a Port to reciprocate another Port, display the Properties window, click on the Port, select the 'Property' page for the Port and select the 'Conjugated' check box (in the expanded 'Property' category). This applies the reciprocal direction of the Flow Properties to the Port, and displays the tilde character (~) in front of the Interface Block reference under the Port name.

Nested Ports in SysML

This statement is derived from the SysML 1.5 Specification:

'Ports nest other Ports in the same way that Blocks nest other Blocks. The type of the Port is a Block (or one of its specializations) that also has Ports.'

For example, a complex number is made up of two members (of type Real) - the real value and the imaginary value. This is represented by the ComplexNumber Block with two Ports, Real and Imaginary. The Equation Block has a Port that takes a ComplexNumber, but we might want to connect the real and imaginary portions of that variable to different sources. Therefore we have to show them, on the Equation Port, in order to connect them.



Once created and nested, the child Ports are always bound to the edge of the Parent Port on which they exist, during a resize or move of the parent element.

This facility is available in Enterprise Architect for all versions of SysML and for UML.

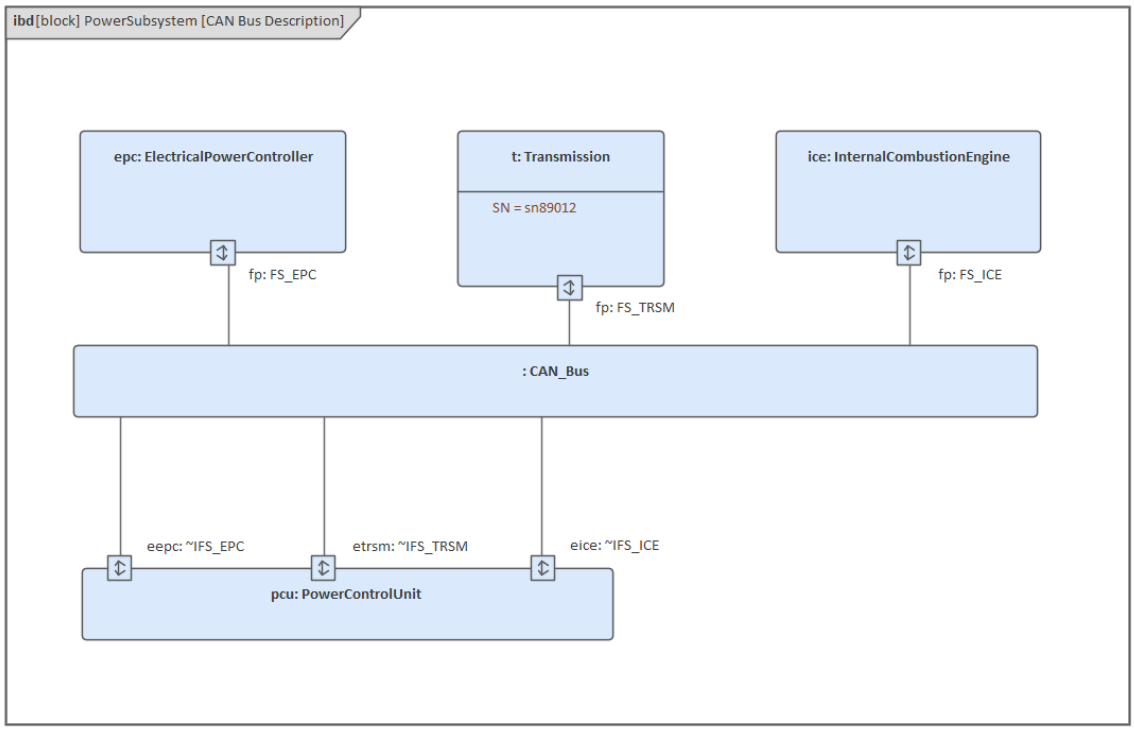
Create Nested Ports in SysML

In describing this procedure, we use a representation of the earlier example as a framework for the steps.

1. Create a SysML Block named 'Equation' with a Port also named 'Equation'.
2. Create a SysML Block named 'ComplexNumber' with Ports named 'Real' and 'Imaginary'.
3. Display the Properties window (press Ctrl+2) for the 'Equation' Port and select the 'Property' page.
4. In the 'Type' field, click on the drop-down arrow and select the 'Select Type' option, then locate and select the 'ComplexNumber' Block element; this sets the 'Type' field to 'ComplexNumber'.
5. Right-click on the 'Equation' Port and select the 'Features | Interaction Points' option. The Features window displays at the 'Interaction Points' tab.
6. Select the 'Show Owned/Inherited' checkbox. The 'Real' and 'Imaginary' Ports are shown in the list panel.
7. Select the checkboxes against 'Real' and 'Imaginary'. The two Ports are now nested in the 'Equation' Port.

Internal Block Diagrams

An Internal Block diagram (IBD) captures the internal structure of a Block element, in terms of its properties (Ports and Parts) and the connections between those properties. The IBD is an instance of the Block element, and the Block is the classifier for the IBD.



The elements in the IBD are enclosed in a frame representing the parent Block element. The name of the parent Block is displayed in both the diagram title and in the frame label; in the example diagram, the Block is called 'PowerSubsystem' and its IBD is called 'CAN Bus Description'.

If necessary you can create more than one IBD for a Block. As an IBD is its Block's composite child diagram, if you have more than one IBD you specify which one is the active child of the Block.

Whilst the IBD defines the structure of a Block, the broader context and usage of that Block is defined in a Block Definition diagram.

IBD Ports

Ports on an IBD can be set to show compartments containing the features and characteristics of the element, such as Tagged Values, Constraints and Attributes. To set which compartments to show, right-click on the Port and select the 'Compartment Visibility' option (for full details, see the *Feature Visibility* Help topic).

To show the compartments, right-click on the Port and select the 'Advanced | Show Compartments' option.

The Ports in the IBD can also be set to show the direction of flow into and out of the Block (by associating them on the Block with a Flow Property). See the *Show Direction on SysML Ports* Help topic.

Model Elements

The Model Elements for Internal Block diagrams are available through the 'SysML Block Internal' pages of the Diagram Toolbox.

The elements that you can use in Internal Block diagrams are:

- Property
- Connector Property
- Distributed Property
- Flow Property
- Participant Property
- Directed Feature
- Adjunct Property
- Bound Reference
- End Path Multiplicity
- Classifier Behavior Property
- Signal
- Port

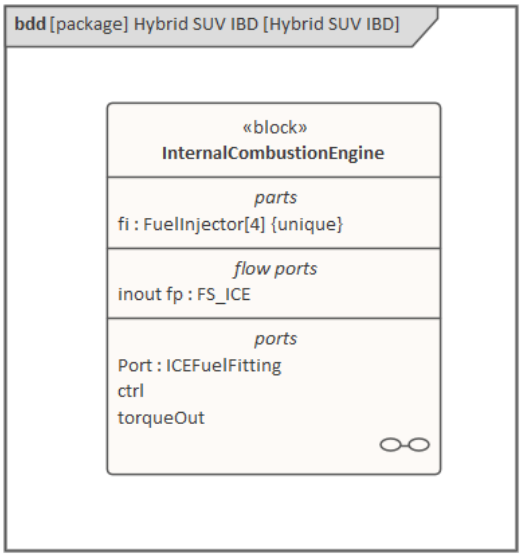
The connectors that you can use in Internal Block diagrams are:

- Item Flow
- Connector
- Binding Connector
- Dependency

Synchronize Structural Elements - Internal Block

If you have already defined the Parts and Properties of a Block element, you can automatically display them on the element's child Internal Block diagram and/or Parametric diagram using a simple context-menu option on the child diagram.

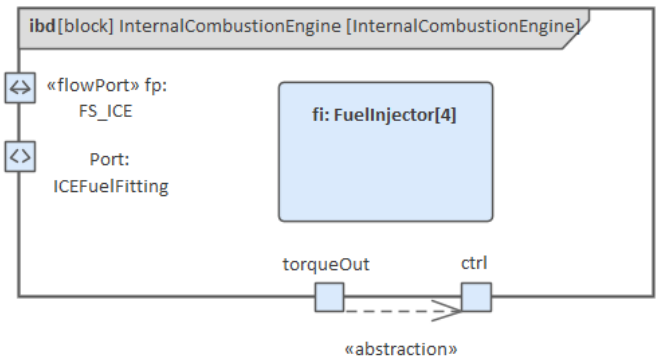
On an existing Block Definition diagram, such as this one:



1. Select the Block element.
2. Right-click and select 'New Child Diagram | Internal Block Diagram' or 'Parametric Diagram'.
3. On the new diagram, external to the diagram frame, right-click and select the 'Synchronize Structural Elements' option.

These elements are added to the Internal Block or Parametric diagram, as linked elements within the diagram frame:

- Each structural element (such as Ports and Parts) owned by the Block element
- Properties defined by existing Association connectors on the Block element
- Edge mounted Ports for any Ports defined in the Block

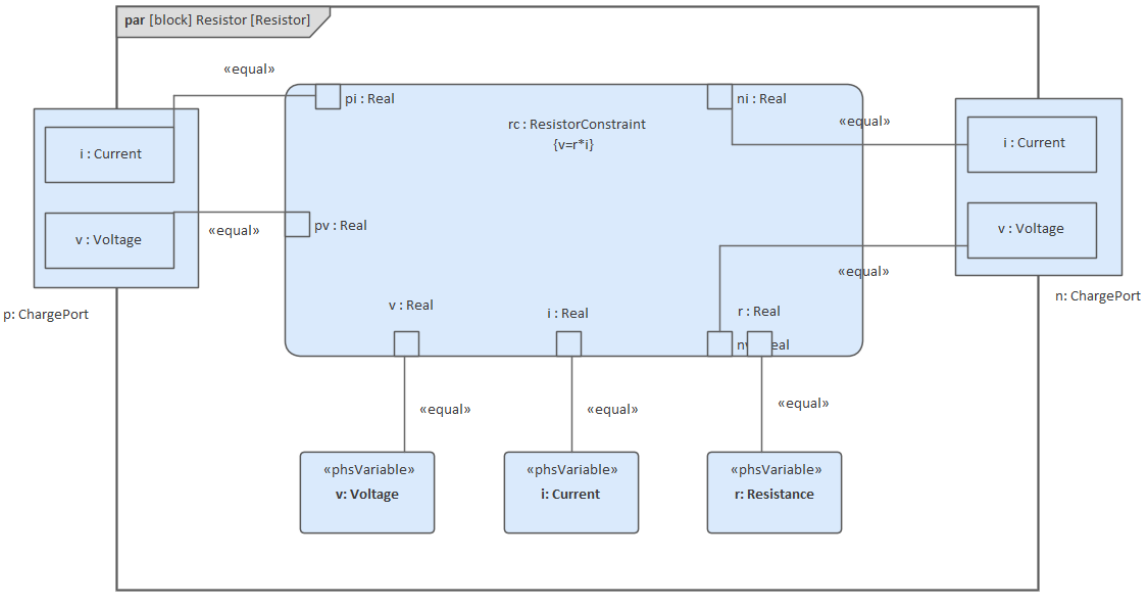


Parametric Diagrams

SysML Parametric models support the engineering analysis of critical system parameters, including the evaluation of key metrics such as performance, reliability and other physical characteristics. These models combine Requirement models with system design models, by capturing executable constraints based on complex mathematical relationships, which can be defined and calculated using the integrated facilities of Math Simulation tools such as MATLAB Simulink, OpenModelica and Octave, under the SysML Extension for Physical Interaction and Signal Flow Simulation (SysPhS) standard. The Mathematical Simulation tools are discussed in the *Mathematical Simulations* chapter.

Parametric diagrams are specialized Internal Block diagrams that help you, the modeler, to combine behavior and structure models with engineering analysis models such as performance, reliability, and mass property models.

SysML Parametric diagrams are dependant on Block definitions being created in the model. The parametric definitions apply equations as constraints on the properties of these Blocks. The equations have parameters that are bound to the properties of the system. Parametric diagrams use ConstraintBlocks to define these constraints. These can be derived from the Block Definition or Internal Block model.

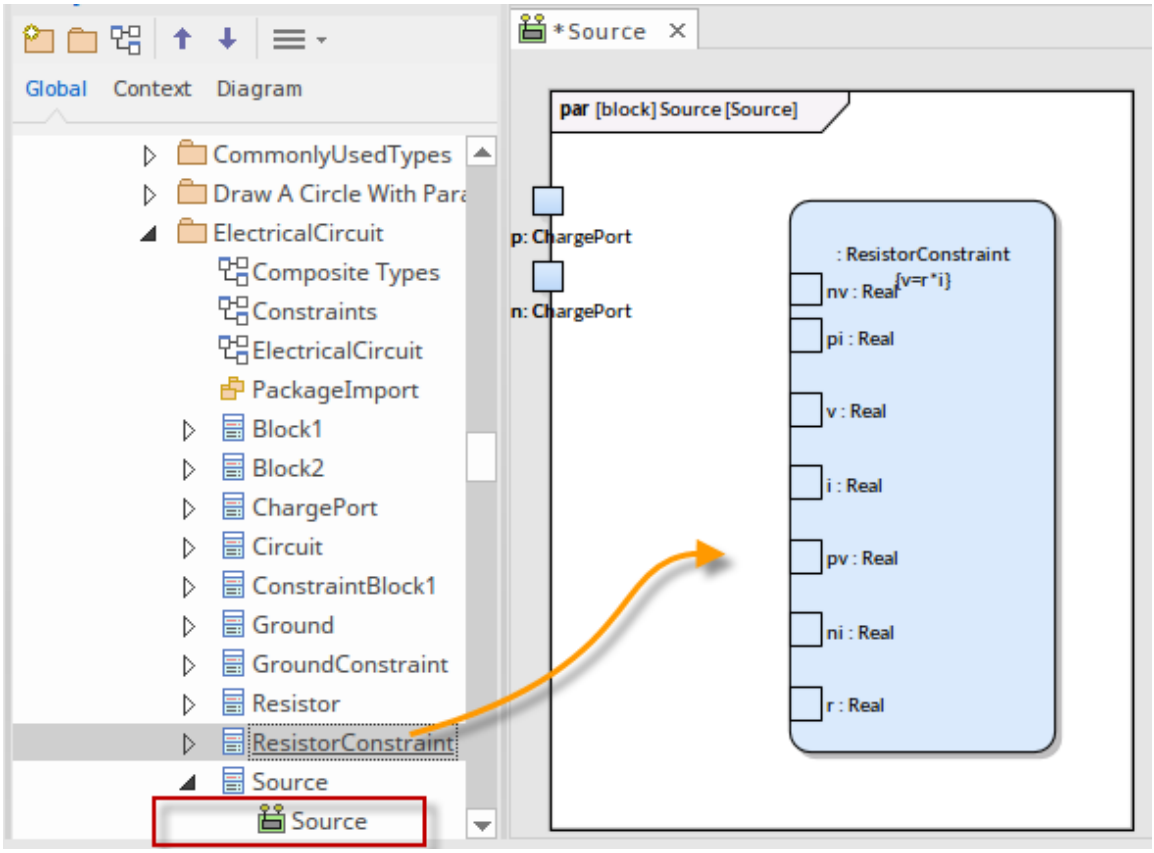


A typical system can contain multiple Parametric diagrams, each defining a specific engineering analysis of specific parts of the system.

Create a Parametric diagram

To quickly set up a ConstraintProperty in a Parametric diagram, containing the equation and the parameters defined in the Constraint Block, simply:

- Create your Parametric diagram (as a child of a Block)
- Drag the 'Constraint Block' icon from the Diagram Toolbox onto the diagram



SysML Parametrics Toolbox page

When you are constructing SysML Parametric models, you can populate the Parametric diagrams with Constraint Block elements using the icons on the 'SysML Parametrics' pages of the Diagram Toolbox.

The Block that owns the Parametric diagram is automatically represented by a diagram frame enclosing the Parametric diagram elements. You can:

- Hide the frame (right-click on the diagram and select the 'Hide Diagram Frame' option) and show it again (select the 'Show Diagram Frame' option)
- Make the frame selectable to move or resize it (right-click on it and select the 'Selectable' option)
- Create Ports and Parts on the frame and create connectors between them

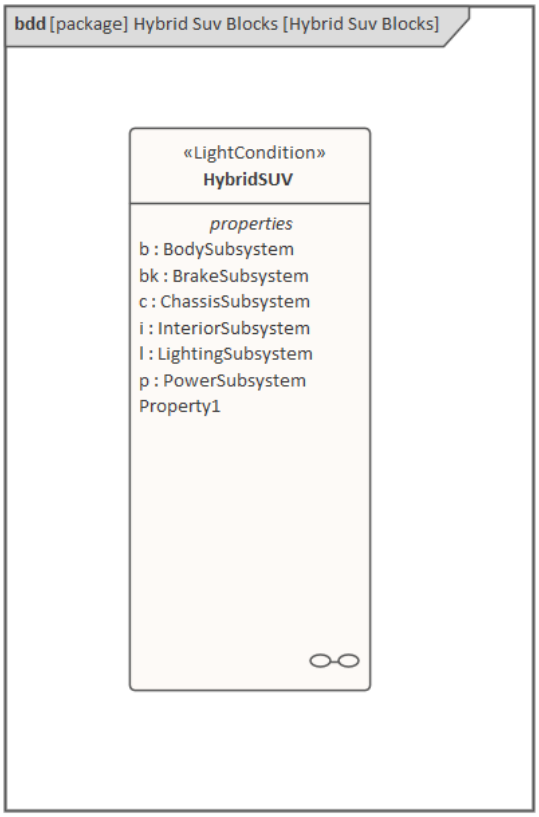
If set to 'Non-selectable', the frame will auto-resize to fit the bounds of the diagram, expanding from its default size but not shrinking smaller.

Note that diagrams showing Diagram Frames applied using release 14.0 or later of Enterprise Architect will draw the parent object on the diagram when opened in a release of Enterprise Architect earlier than release 14.0.

Synchronize Structural Elements

Where the Parts and Properties of a parent Block have already been defined you can display them on the Internal Block Diagram (IBD) with a simple context-menu option on the new IBD.

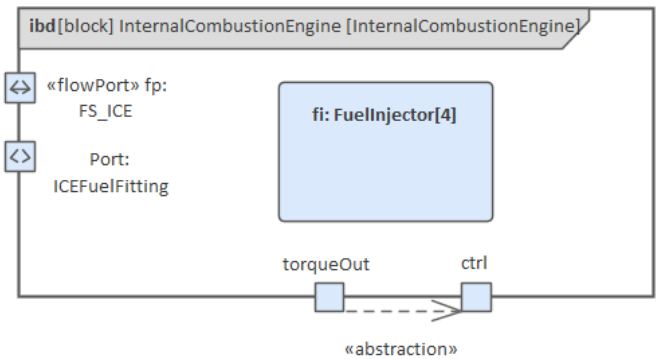
Given an existing Block Definition Diagram (BDD):



- Select the Block in the Browser window
- Right-click on the Block and select the 'New Child Diagram | Internal Block Diagram' option
  This creates the new IBD frame
- External to the IBD frame - right-click on the IBD and select the 'Synchronize Structural Elements' option

This places as element-links all the structural elements (such as Ports and Parts) relating to the Block that owns this diagram.

This command will also generate Properties defined by existing Association connectors.



This will include edge mounting any Ports defined in the Block.

Resizing Ports

Having created a Port on your Parametric diagram, you can resize it to accommodate any text it contains. You have two options:

- Right-Click - Advanced | Port Size Customizable

- Right-Click - Advanced | Bind to Connector Role

Parametric Diagram Modeling Assistant

This feature is available from Enterprise Architect Release 14.1.

Enterprise Architect provides a set of convenient tools to help you create ConstraintBlocks and parameters, by parsing mathematical equations to create parameter binding connectors and using a hierarchical element picker to create embedded elements.

Bind Parameters of a ConstraintProperty

When you create a ConstraintProperty, you define an equation or expression as the constraint. You can then bind the parameters of the constraint to properties to identify what the parameter is and where its values come from.

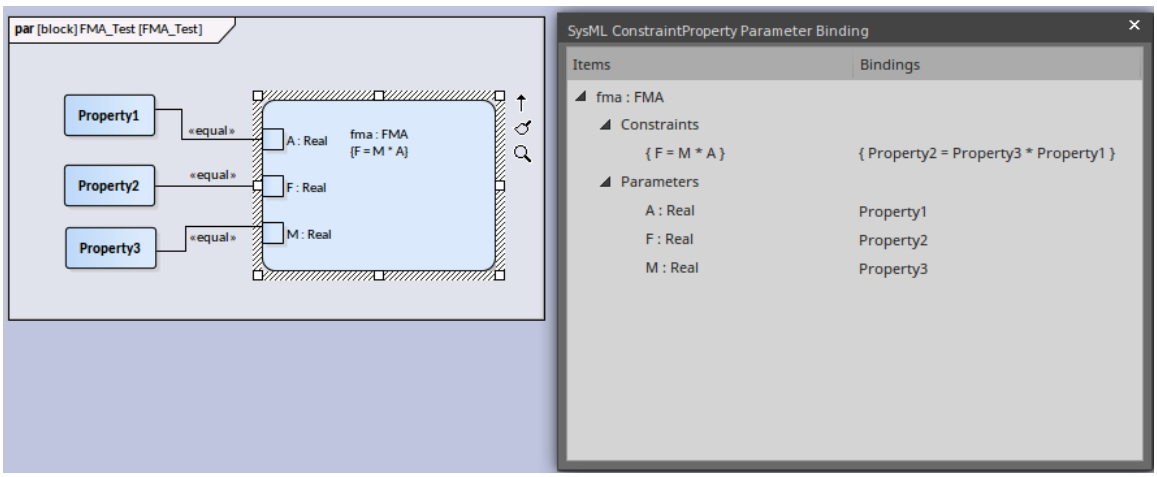
Access

Context menu	In a diagram, right-click on an existing ConstraintProperty | Edit Constraint Property...
Other	In Browser window | Drag a Constraint Block and drop it on a Parametric diagram A ConstraintProperty with all the parameters will be created

Bind parameter to properties in context

In this example, we create the Constraint Block 'FMA' and use it as a ConstraintProperty 'fma' in the context of a Block 'FMA_Test', which contains three properties: 'Property1', 'Property2' and 'Property3'.

Right-click on ConstraintProperty 'fma' and select the 'Edit Constraint Property...' context menu option to open the 'SysML ConstraintProperty Parameter Binding' dialog.

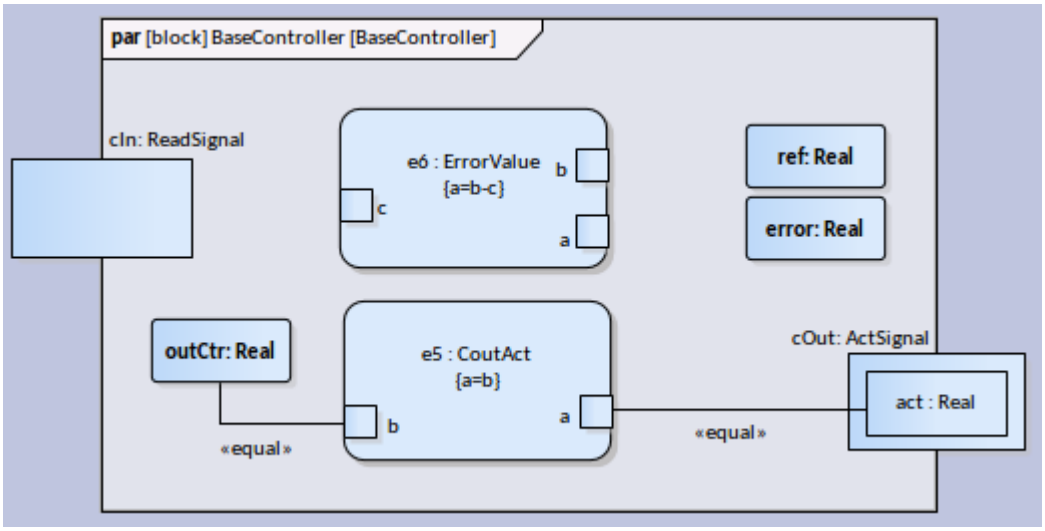


Click on the ... button in the same row as a parameter to open the 'Hierarchy Properties Picker' dialog; choose a property to bind to the parameter.

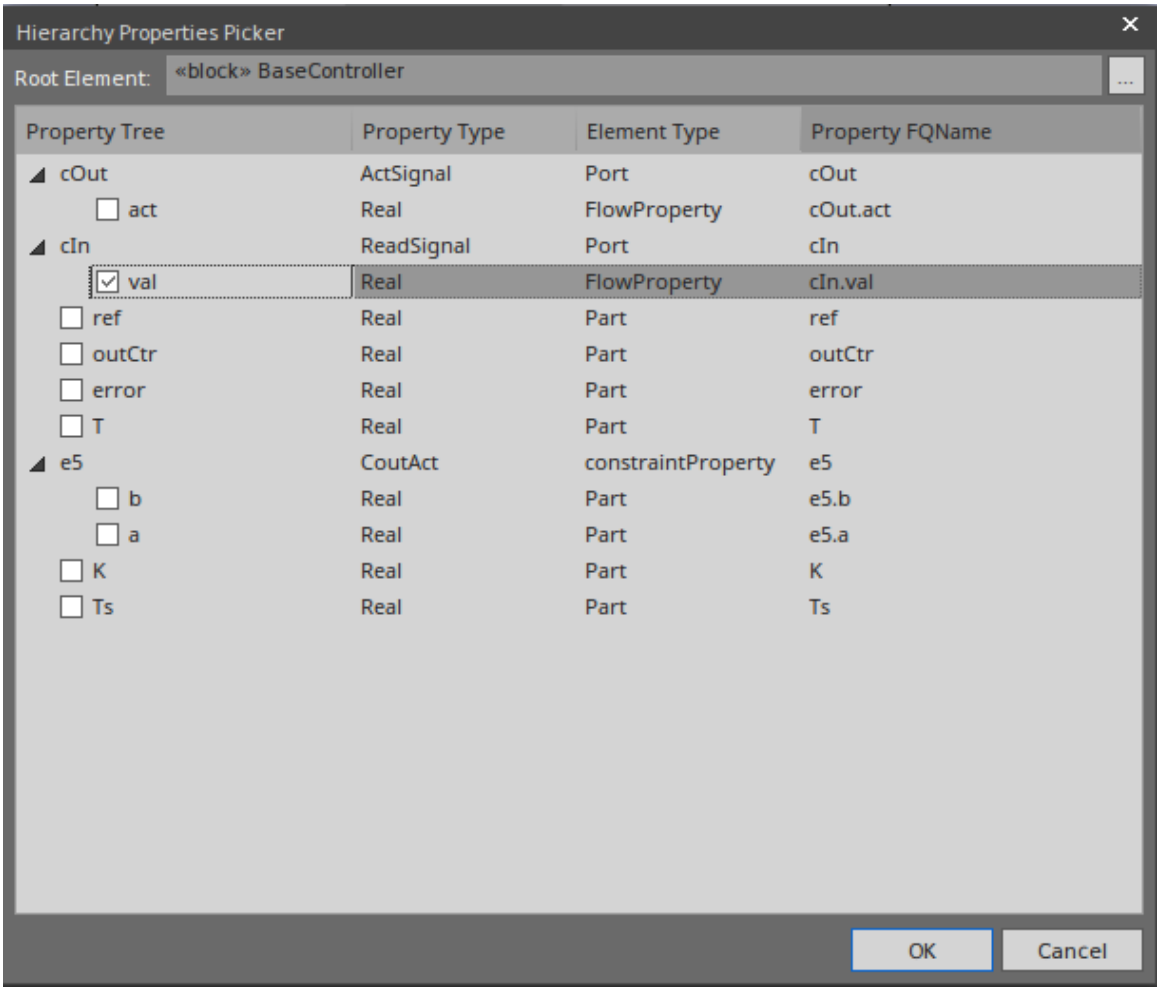
After binding, the property will be shown on the diagram and a connector will connect it to the parameter of the ConstraintProperty. The equation 'F =M*A' becomes 'Property2 = Property3 * Property1' after binding.

Hierarchy Element Binding

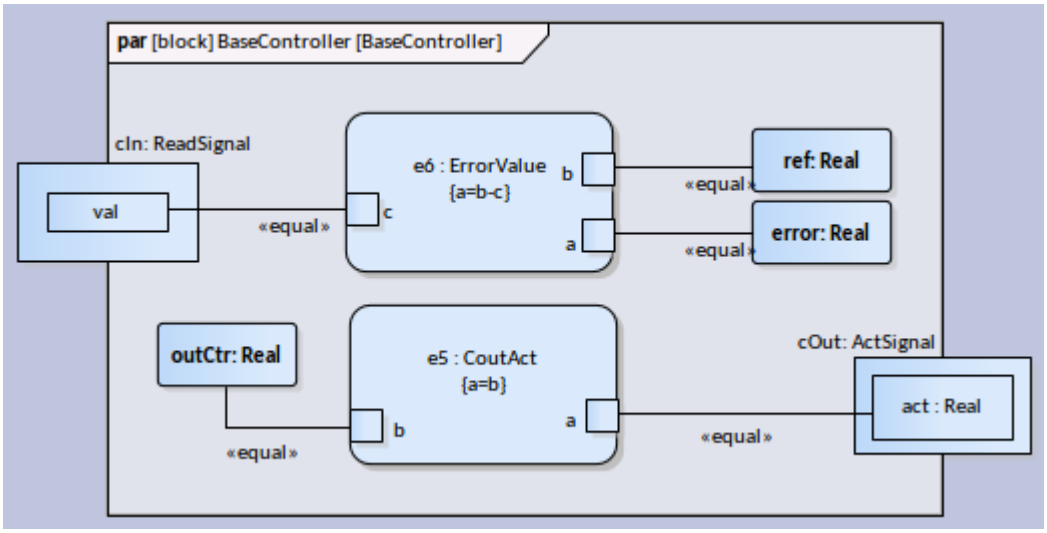
In this example, Block *BaseController* has a ConstraintProperty *e6* with parameters a, b and c. Now we want to bind the parameters to the Block's properties. Specifically, we want to bind parameter *e6.c* to *cIn.val*, which is a Flow Property defined in Block *ReadSignal*; *cIn* is the Port defined on *BaseController*.



This figure shows the hierarchy of properties defined in *BaseController*. The properties with matching type to the binding parameter will be shown with a checkbox for selection.

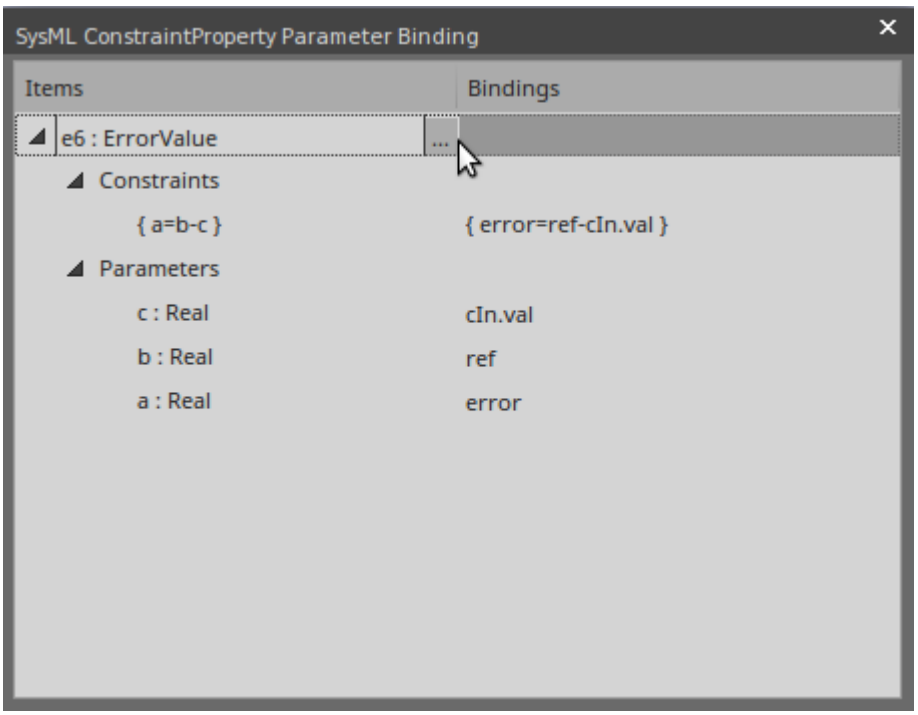


We select property *val* under *cIn* and click on the OK button. The property *val* will be created on the diagram inside the Port *cIn*, and a Binding connector between *cIn.val* and *e6.c* will be created. After binding parameter *e6.a* to property *error*, and *e6.b* to property *ref*, the diagram will resemble this:

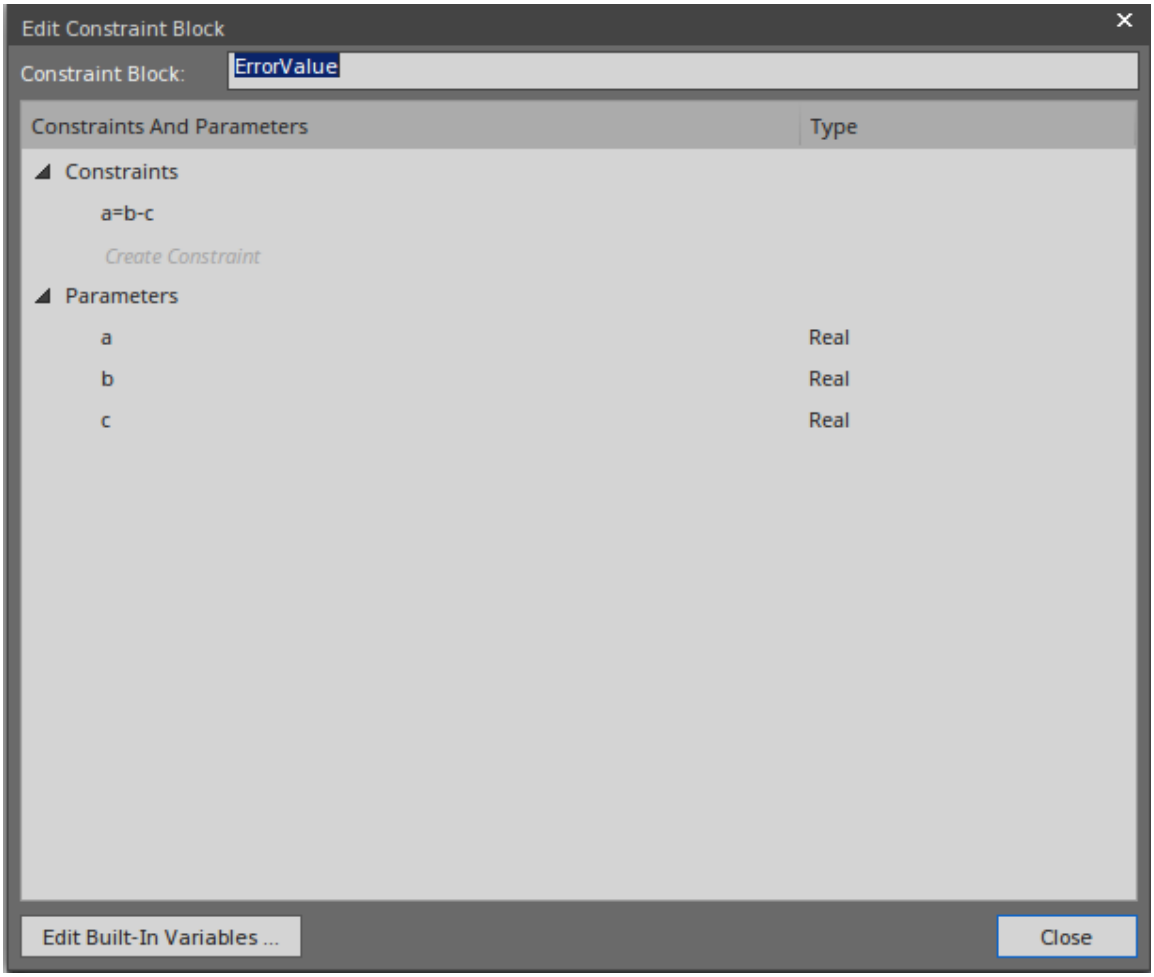


Navigate to ConstraintBlock

Select the cell of the ConstraintProperty.



Click on the ... button on the right to edit the typing Constraint Block. The 'Edit Constraint Block' dialog displays.


Edit Constraint Block
Constraint Block:
ErrorValue
Constraints And Parameters
Type
Constraints
a=b-c
Create Constraint
Parameters
a
Real
b
Real
c
Real
Edit Built-In Variables ...
Close

Compose System Design

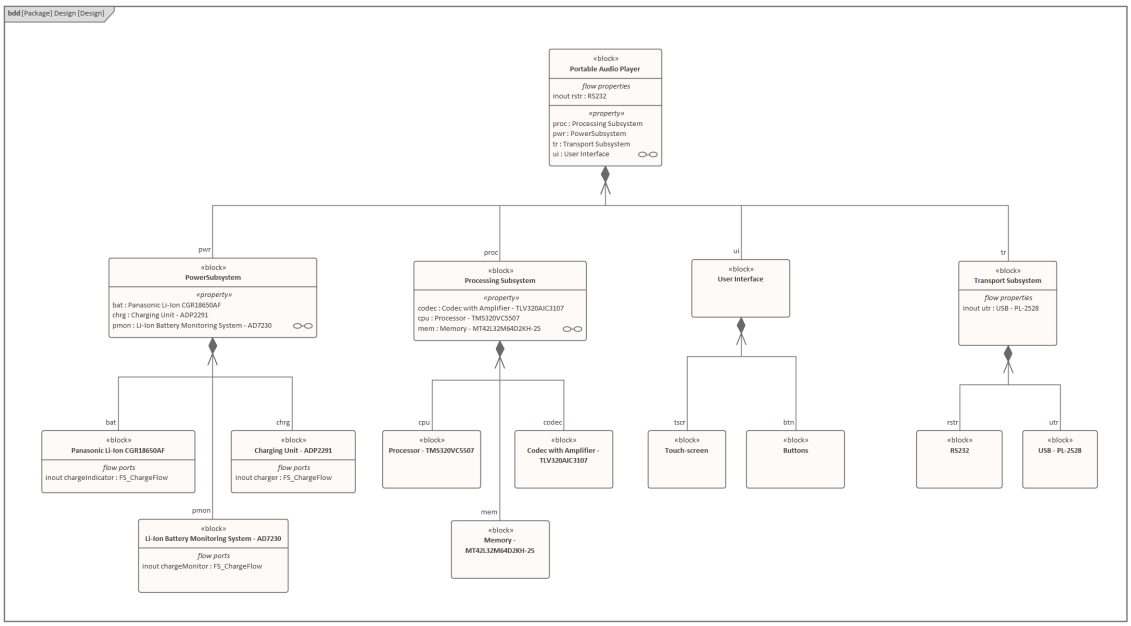
The SysML systems engineering modeling language has a strong focus on design. Once the requirements have been elicited, modeled and analyzed, the attention of the engineer turns to design. Systems are typically complex and must be broken down into a number of subsystems that will interact with each other through known and published interfaces.

A Block Definition diagram can be created in Enterprise Architect to model the decomposition of a system into a hierarchy of subsystems. Subsystems can be hyper-linked to more detailed diagrams, which allow the viewer to click through from the system level through to all its constituent parts. The subsystems can also be linked back to the requirements they are implementing, and then to the stakeholders who own the requirements.

The SysML Design Model

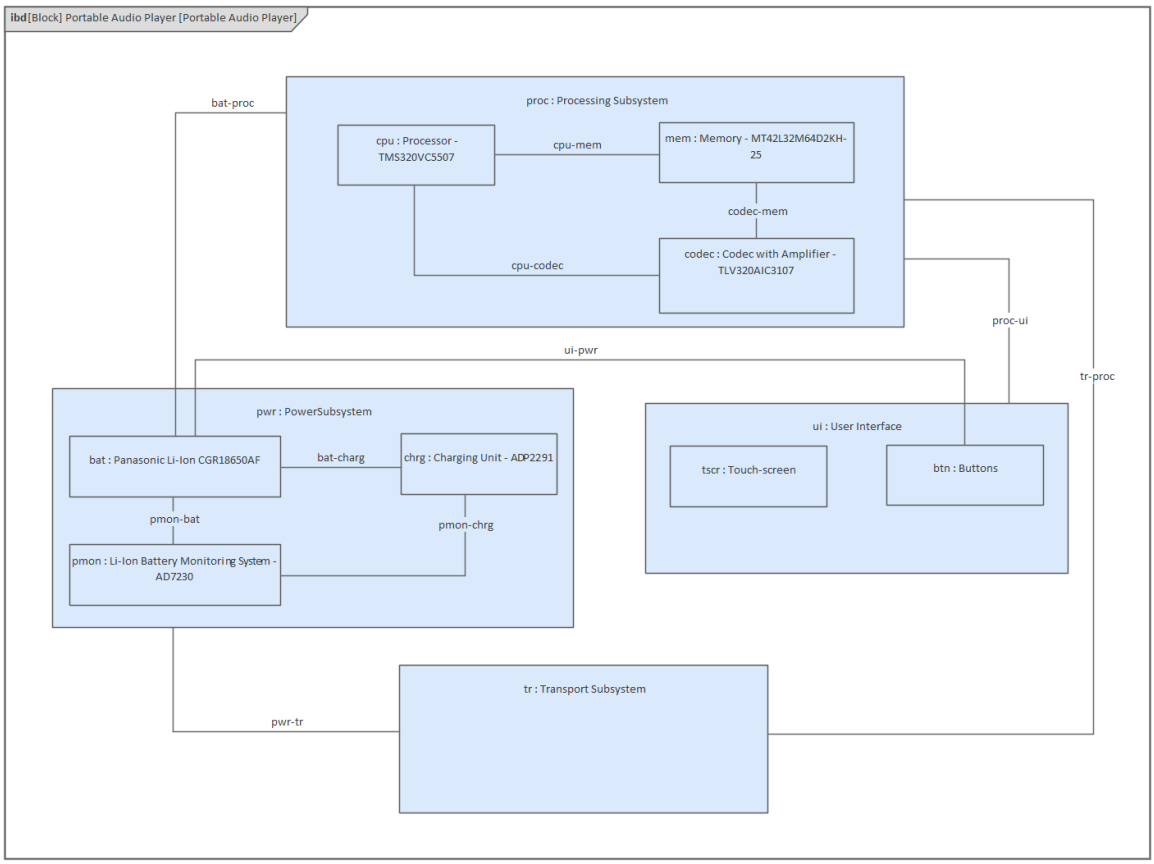
The SysML Design Model contains the Blocks that define the system's composition; it describes the manner in which reusable subsystems fit together to fulfill the design requirements.

This diagram shows an example Design Model for a Portable Audio Player; the SysML Block Definition diagram describes the Portable Audio Player as a composition of various reusable off-the-shelf subsystems and in-house designed ones.



In this example, the Portable Audio Player is defined in SysML as a system composed of subsystems that perform specific tasks; the design shows subsystems for supplying power, performing playback and audio processing, and interfacing with other devices and the user interface.

The Portable Audio Player's composition is described in greater detail in the Portable Audio Player's Internal Block diagram:



The example describes the Portable Audio Player's composition, detailing how each of the subsystems is structured. The example also describes the relationships between the parts, defining how they are functionally bound to one another; for example, the CPU, Memory and Codec are interfaced together in the Processing Subsystem.

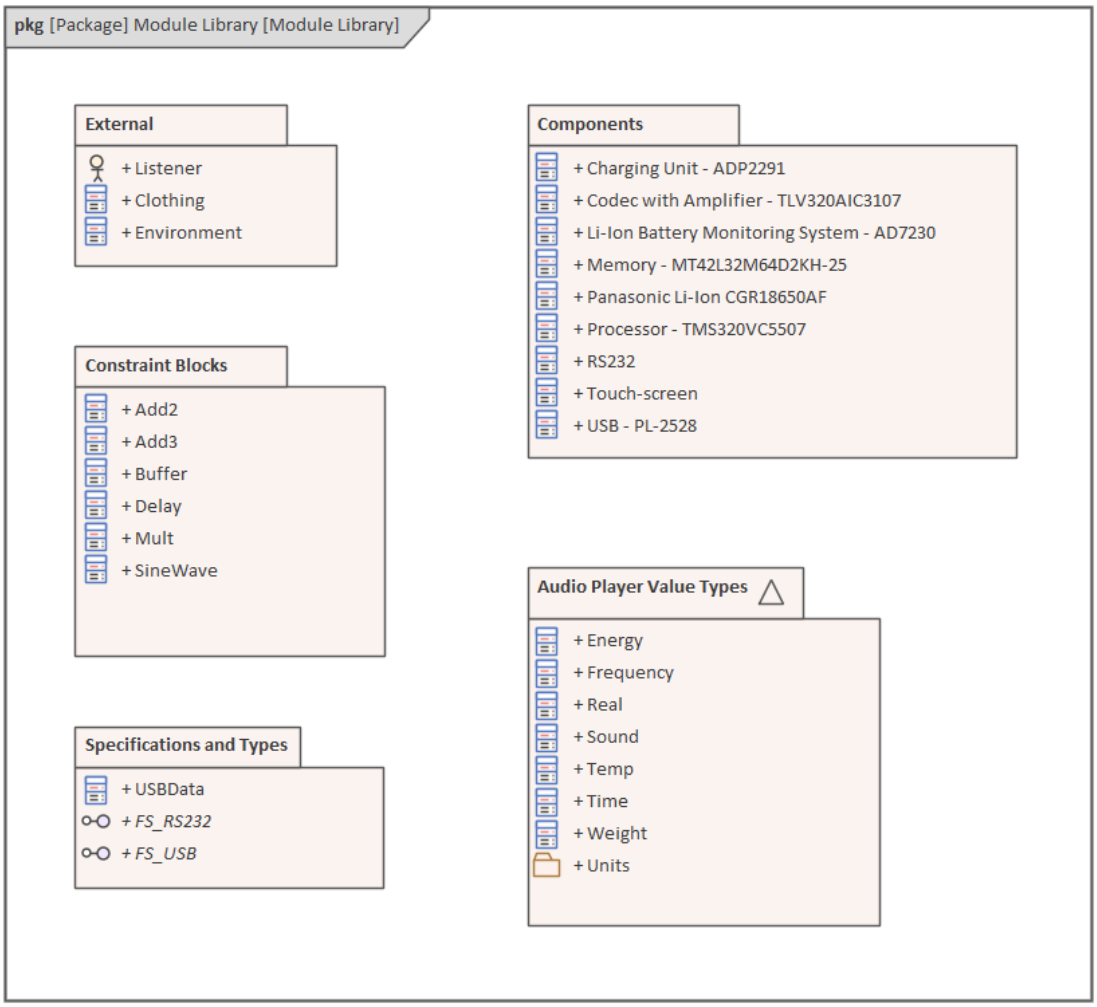
Create Reusable Subsystems

Model-based Systems Engineering provides the flexibility and expressiveness to define complex systems quickly and effectively, by reusing common entities across design projects. Before the model based approach became prevalent, systems were defined using document based methods with little opportunity for re-use. The SysML contains a series of reusable libraries such as the SI Definitions and the SI Value Types, but also supports a modeler in creating additional domain or technology-specific libraries that could be reused within an organization, or published for wider use by a community of users or an entire industry.

Enterprise Architect provides a range of functionality to assist in creating, discovering, visualizing and reusing libraries of elements such as subsystems, parametric constraints, common data types, common value types, dimensions and units. The Reusable Asset Service could be used to store these assets in order to provide a canonical set of libraries governed by the respective standards agencies.

SysML Design Model

A Library is a Package containing many reusable subsystems, parametric constraints, common data types and common value types, dimensions and units. This diagram shows an example library model:



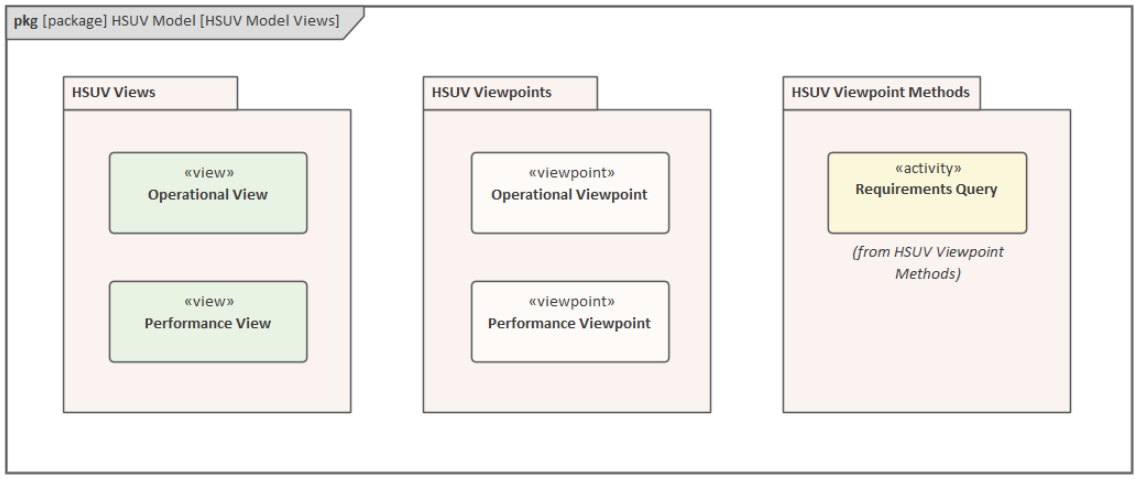
In the example Library, each of the child Packages contains child models that capture these reusable entities:

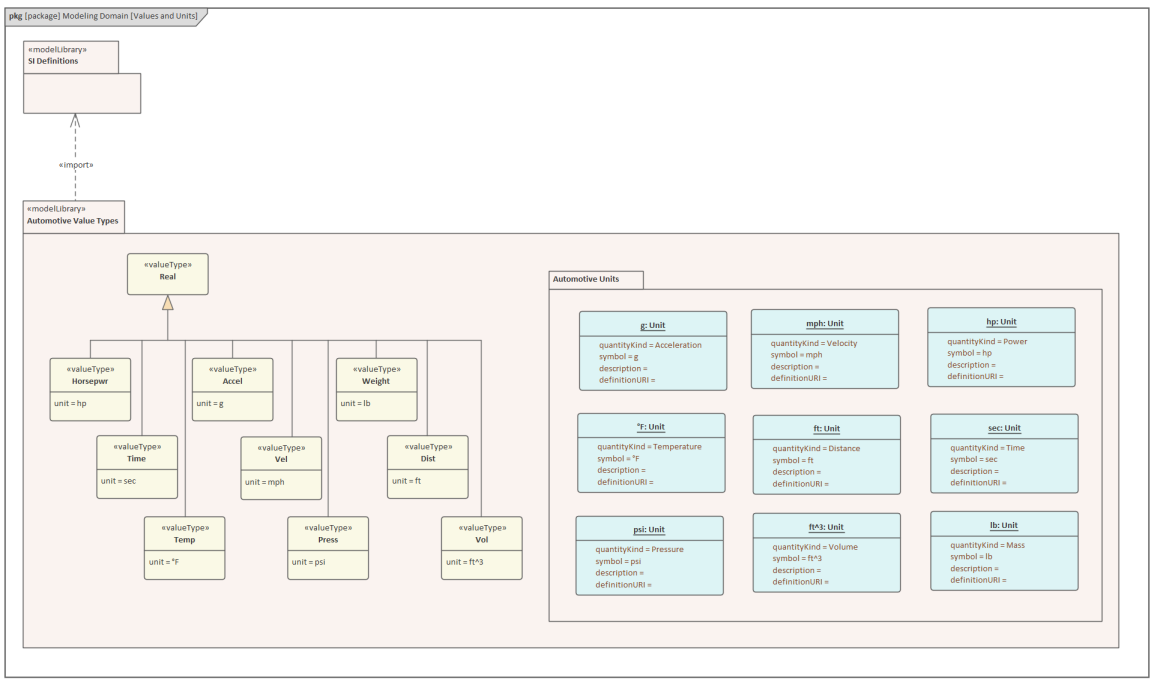
- Blocks defining systems such as those listed in the Components Package, or those defined in the External Package
- ConstraintBlocks defining parametric constraints for use in parametric models
- Value Types describing quantities, expressed as measurable dimensions in specific units

- Data Types and Flow Specifications describing data structures and Flows

SysML Package Diagram

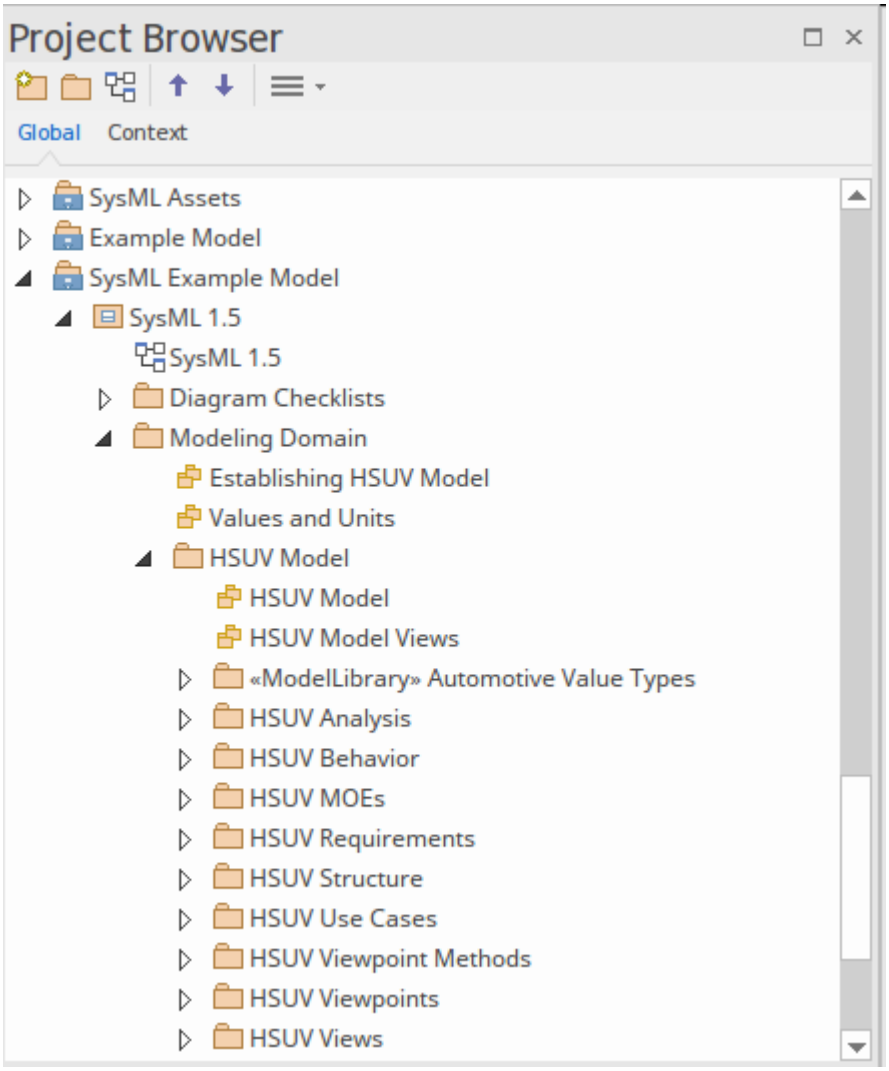
A SysML Package diagram provides a means of visualizing the organization of a complex model into recognizable containers, which helps you to group the structures of the model and define high level relationships between these groupings. The structures can include name-spaces and their sub-Packages, and other less formally defined groups of elements. The basis of allocating structures to Packages could be, for example, access control, configuration Management, ease of navigation, or dependency level.



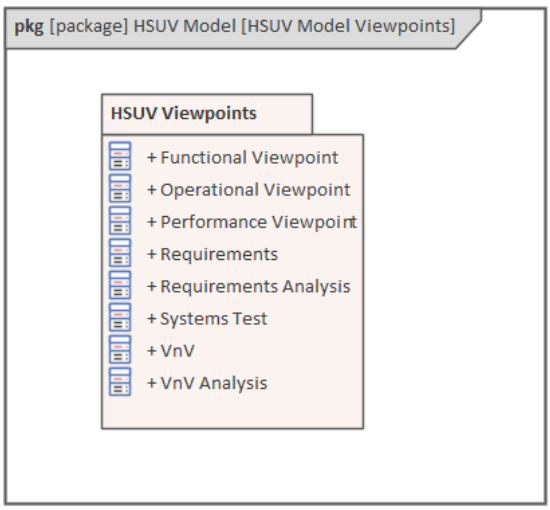


The main element represented in the Package diagram is the Package itself, which can represent a complete Model, a Model Library of commonly-used objects (such as definitions and values), or a Package as a container. The diagram can also show two stereotyped Class elements - a SysML View, which defines an aspect of a system, from a perspective defined by a SysML View Point. The diagram can contain no relationships, simply indicating how objects are grouped, or it can show a number of relationships to indicate how the Packages are interlinked. For details of the element and connector types, see the *SysML Model Toolbox* Help topic.

The Packages that appear in diagrams can also be viewed in the Browser window, and their hierarchy can be navigated by expanding and collapsing the tree.

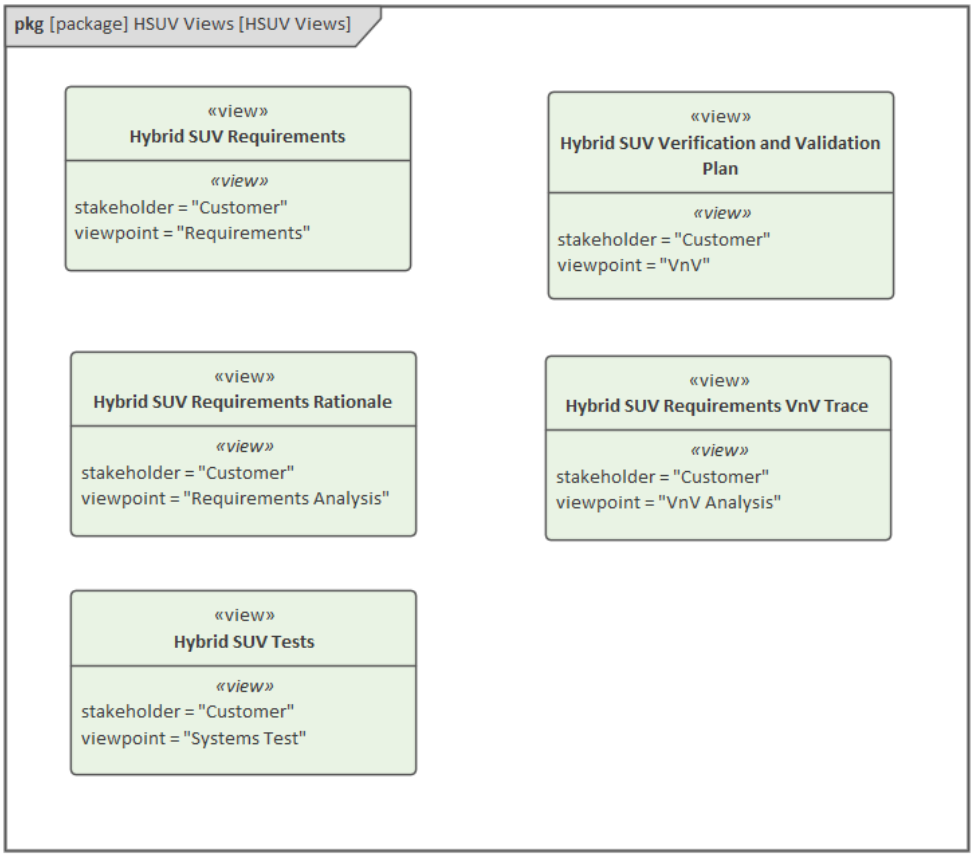


Your use of the SysML diagram facilities can show the model structure in different ways. The simplest way is to just show Packages as containers, with their full contents. For example:

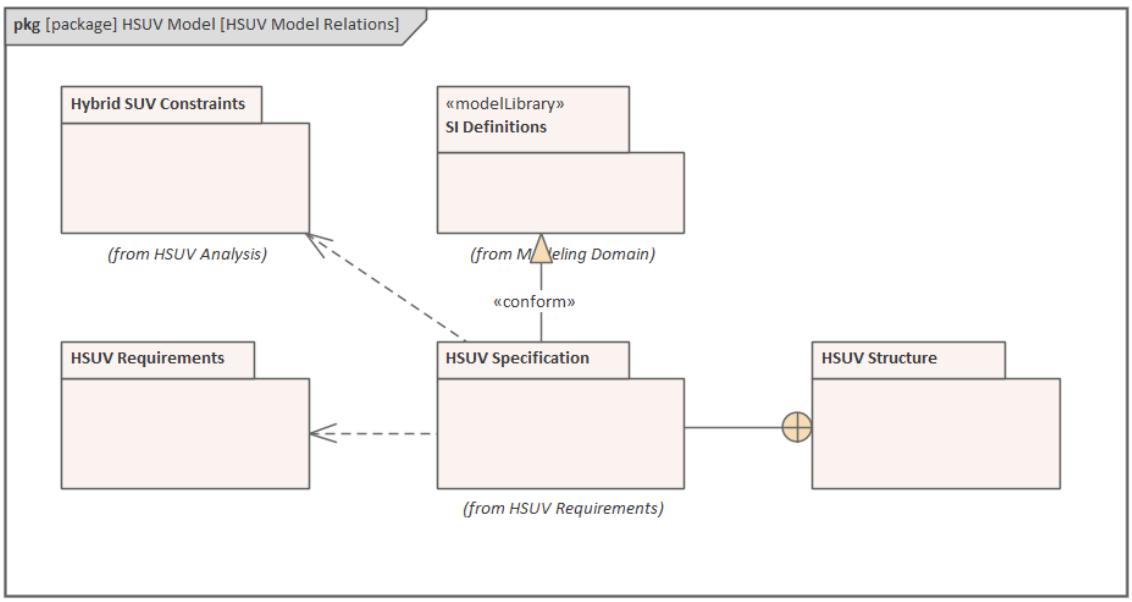


This appearance is achieved by setting the diagram and Package element to show compartments, specifically the 'Package Contents' compartment. It shows the complete contents, but gives no indication of structure or importance.

If you want to show certain structures that are relevant to the purpose of the diagram, but not the complete content, you can turn off the compartment and drag the required elements onto the Package on the diagram.



This format shows Package contents that are important to a purpose, but it does not indicate structure or relationships. You can show the structure clearly using a third format, where the content elements are separate and linked by labeled connectors.



Depending on what you want to represent in the diagram, you can use any combination of these formats for different elements in the same diagram.

Elements

The main elements that you can create in Package diagrams are:

- Model
- Model Library
- Package
- View
- View Point
- Stakeholder

The main relationships that you can use in Package diagrams are:

- Conform
- Dependency
- Import
- Containment
- Realization
- Refine
- Expose

SysML Use Case Models

A Use Case represents a single unit of meaningful work, providing a high-level view of behavior observable to someone or something outside the system. The SysML notation for a Use Case is an ellipse, as for the UML notation. Use Cases are intended to provide a more detailed expression of the requirements modeled on a higher level.

Use Case diagrams capture Use Cases and the relationships between Actors and the subject (system). You can use them to:

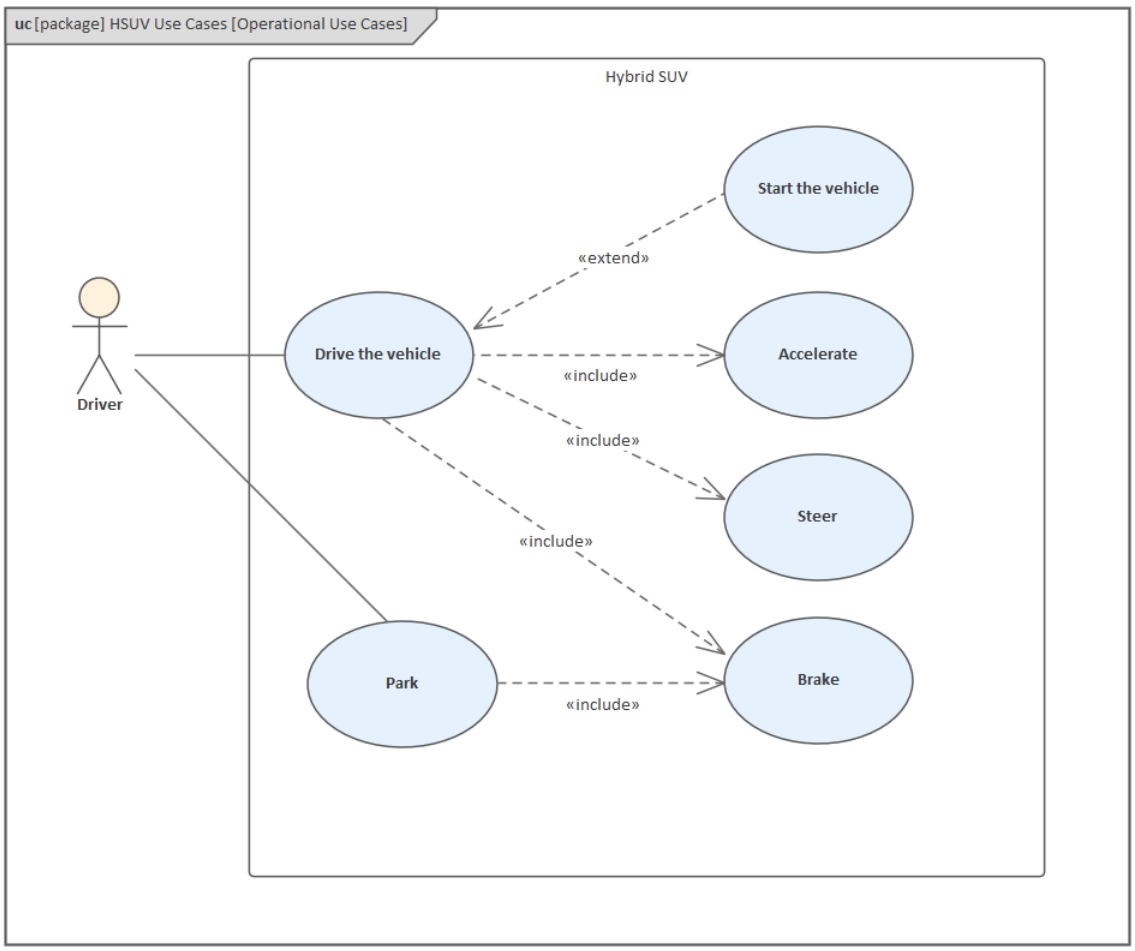
- Expand on (or 'realize') the functional requirements of the system
- Describe the manner in which outside things (Actors) interact at the system boundary
- Describe the response of the system

Use Cases also support more detailed definition via Use Case Scenarios.

Use Case Diagram

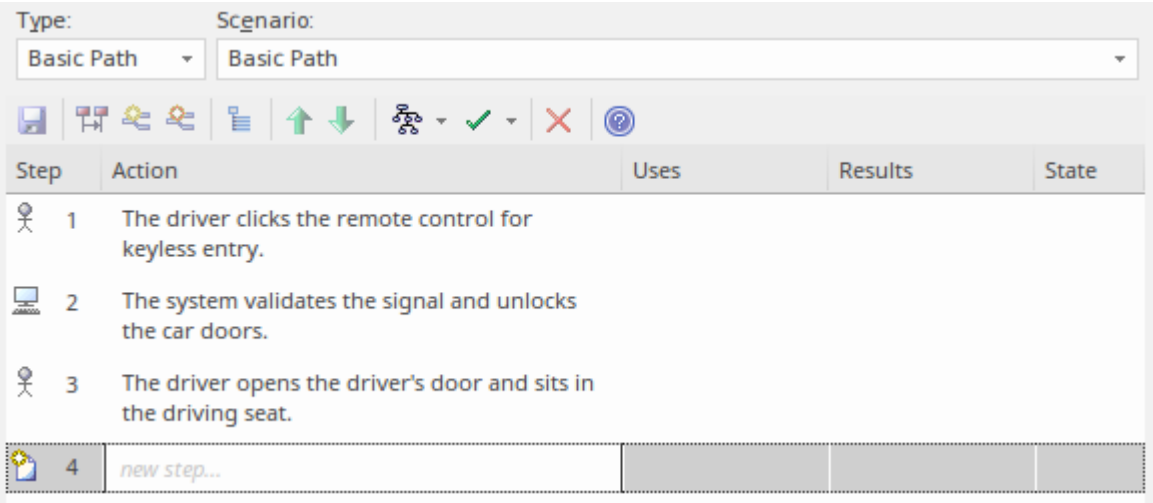
A SysML Use Case diagram is used to define and view Use Cases and the Actors that derive value from the system. The Use Case diagram describes the relationship between the Actors and the Use Cases. Enclosing the Use Case within a Boundary defines the border of the system; the Actors by definition lie outside the boundary. (All elements are internal to the SysML diagram frame.)

While the Use Case diagram can appear simplistic, it is a great communication device that describes the value or goals that external roles achieve from interacting with the system. Each Use Case can be detailed, with descriptions, constraints and any number of scenarios that contain sets of steps performed alternately by Actor and system to achieve the desired goal.



Use Case Scenario

A core aspect of modeling Use Cases is performed using the Use Case Scenario feature, which helps add a more detailed text-based representation of the underlying Use Case.



Using the Scenario builder, behavioral model diagrams can then be generated from these text-based details to provide a base for the more detailed design.

Elements

The main elements that can appear in Use Case diagrams are:

- Actor
- Use Case
- Boundary

The main connectors that can appear in Use Case diagrams are:

- Communication Path
- Generalize
- Include
- Extend

Usage

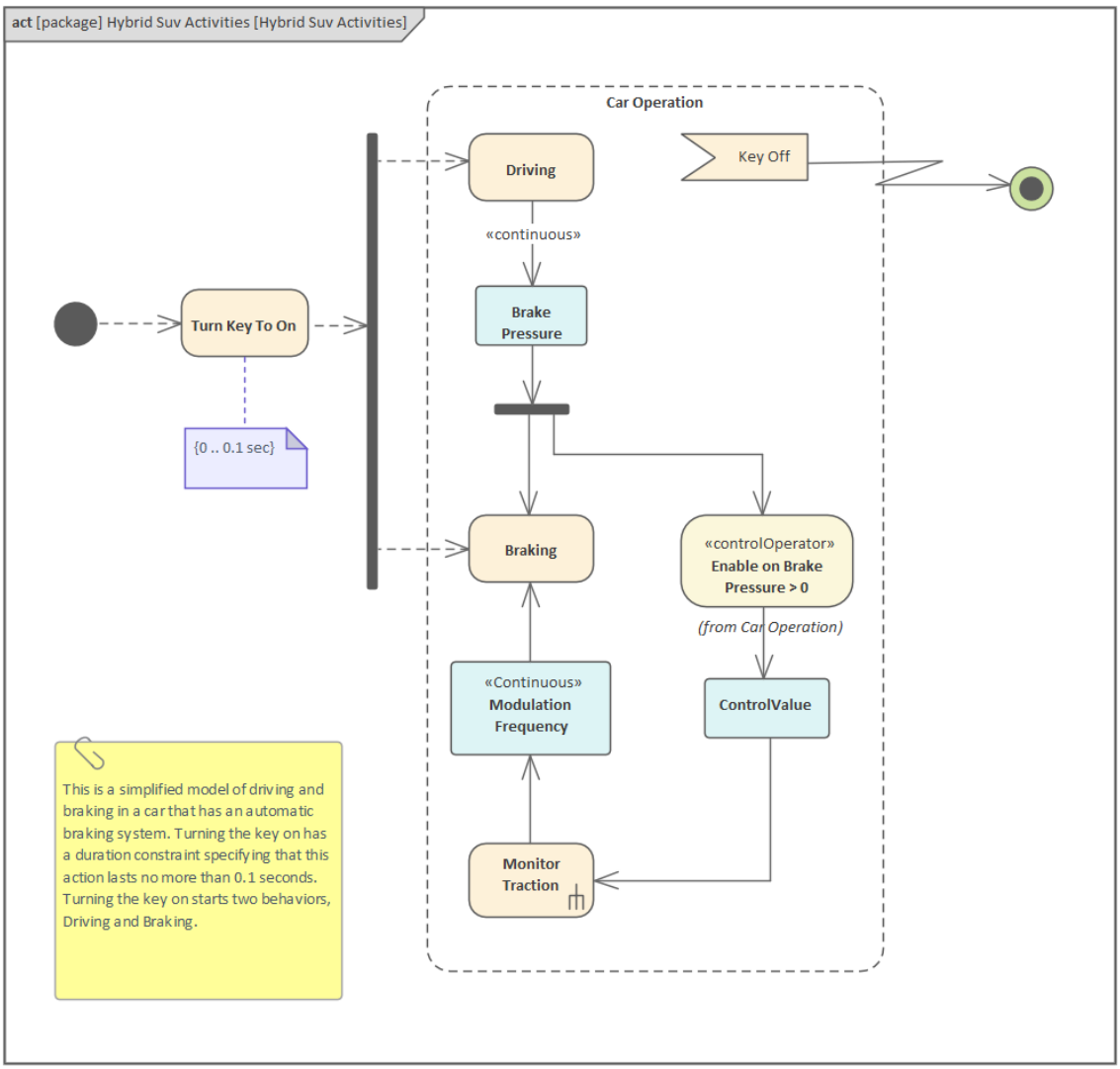
A Use Case diagram can be used to define the details of a Use Case and its Scenarios and Constraints, with tight connectivity and traceability, as the Use Case diagram and the textual details of the Use Cases, Scenarios and Constraints are all contained in the same model. From the Use Case scenarios you can generate Activity diagrams, Sequence diagrams and StateMachine diagrams as starting points for the analytical modeling.

SysML Activity Diagram

A SysML Activity diagram is an extension of the UML Activity diagram. The Activity diagram is a tool for representing the sequence of Actions that describe the behavior of a Block or other structural element; the sequence is defined using Control Flows. Actions can contain Input and Output Pins that act as buffers for items that flow from one Action to another as the task carried out by the Action either consumes or produces them. The items can be physical materials, energy, power, data, information, or anything else that can be produced, conveyed or consumed, depending on the system and the activity being described.

Activity Diagrams can be used to define situations where parallel processing occurs in the execution of some activities. Activity diagrams are useful for engineering modeling, where they detail the processes involved in system activities.

This is an example of an Activity diagram.



The SysML Activity diagram is based on the UML Activity diagram, but additional semantics have been added in two areas:

- Continuous Flow, allowing restrictions on the rate at which entities flow along edges in an Activity, and providing mechanisms to ensure that the most recent information is available to Actions
- Probability, introduced into Activities to include the likelihood that a value will be available to an edge or output on a parameter set

Elements

The main elements that can appear in Activity diagrams are:

- Activity
- Structured Activity
- Action (various kinds)
- Action Pin
- Partition
- Control Operator
- Parameter (various kinds)
- Object Node
- Central Buffer Node
- DataStore
- Decision
- Merge
- Synch
- Initial
- Final
- Flow Final
- Region
- Exception
- Fork/Join

Connectors

The main connectors that can appear in Activity diagrams are:

- Control Flow
- Object Flow
- Interrupt Flow
- Dependency

Notes

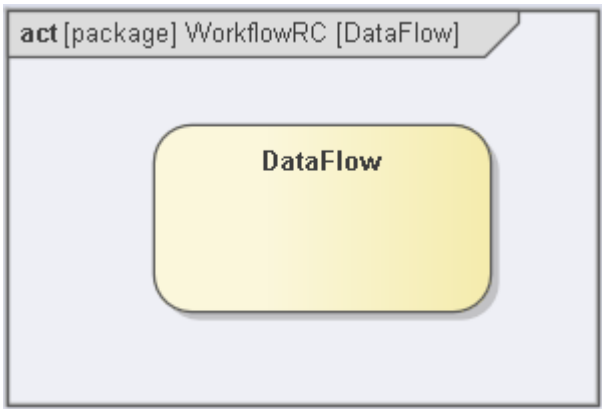
When creating an Activity diagram as a child of an Activity:

- Where the Activity contains Activity Parameters, on creating the child Activity diagram, these parameters are generated on to the Diagram Frame
- Right-clicking on the diagram and selecting 'Synchronize Structural Elements' brings onto the diagram any missing Activity Parameters that might have been added later in the parent Activity

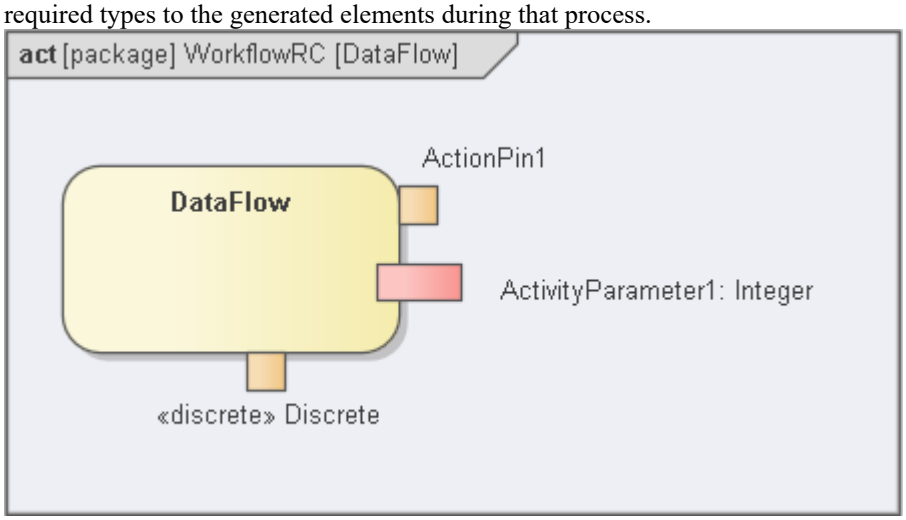
Synchronize Structural Elements - Activity Diagram

If you have already defined the Action Pins and/or Activity Parameters of a SysML Activity element, you can automatically display them on the element's child Activity diagram using a simple context-menu option on the child diagram.

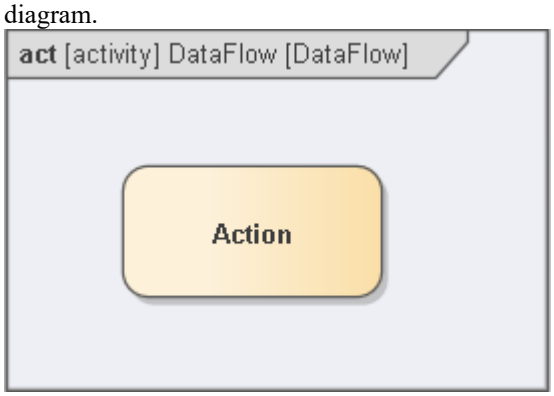
On an existing SysML Activity diagram, such as this one:



1. Drag from the SysML Activities page of the Diagram Toolbox the Action Pin and/or Parameter icons, and assign the required types to the generated elements during that process.



2. Right-click and select 'New Child Diagram | 'Activity Diagram'. Add the appropriate Action element(s) to the diagram.

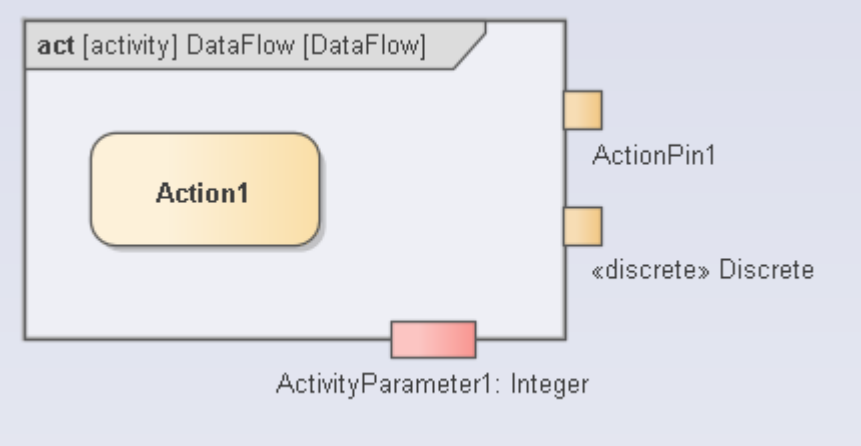


3. On the new diagram, external to the diagram frame, right-click and select the 'Synchronize Structural Elements' option.

These elements are added to the Activity diagram, as linked elements within the diagram frame:

- Each structural element (such as Activity Parameters) owned by the Activity element

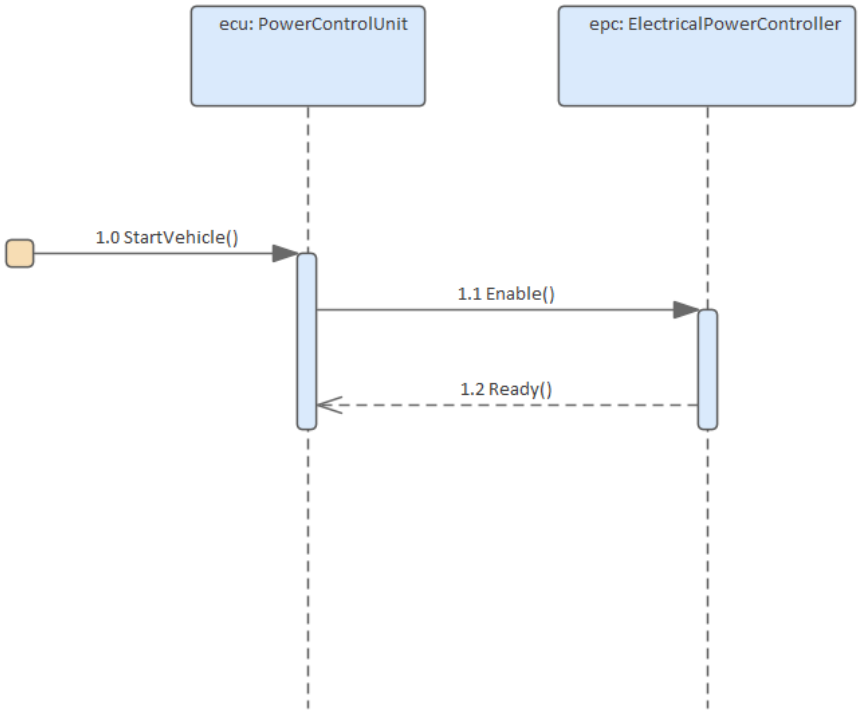
- Properties defined by existing Association connectors on the Activity element
- Edge mounted Activity Pins for the Activity



SysML Sequence Diagram

A SysML Sequence diagram, as for a UML Sequence diagram, is used to display the interaction between users, screens, objects and entities within the system. It provides a sequential map of messages passing between objects over time. Frequently, these diagrams are placed under Use Cases in the model to illustrate the Use Case scenario - how a user will interact with the system and what happens internally to get the work done.

A Sequence diagram is a form of Interaction diagram, which shows objects as Lifelines running down the page, with their interactions over time represented as Messages drawn as arrows from the source Lifeline to the target Lifeline. Sequence diagrams are good at showing which objects communicate with which other objects, and what messages trigger those communications.



Elements

The main elements that can appear in Sequence diagrams are:

- Sequence
- Fragment
- Endpoint
- Interaction
- Diagram Gate
- State/Continuation

Connectors

The main connectors that can appear in Sequence diagrams are:

- Message
- Self Message

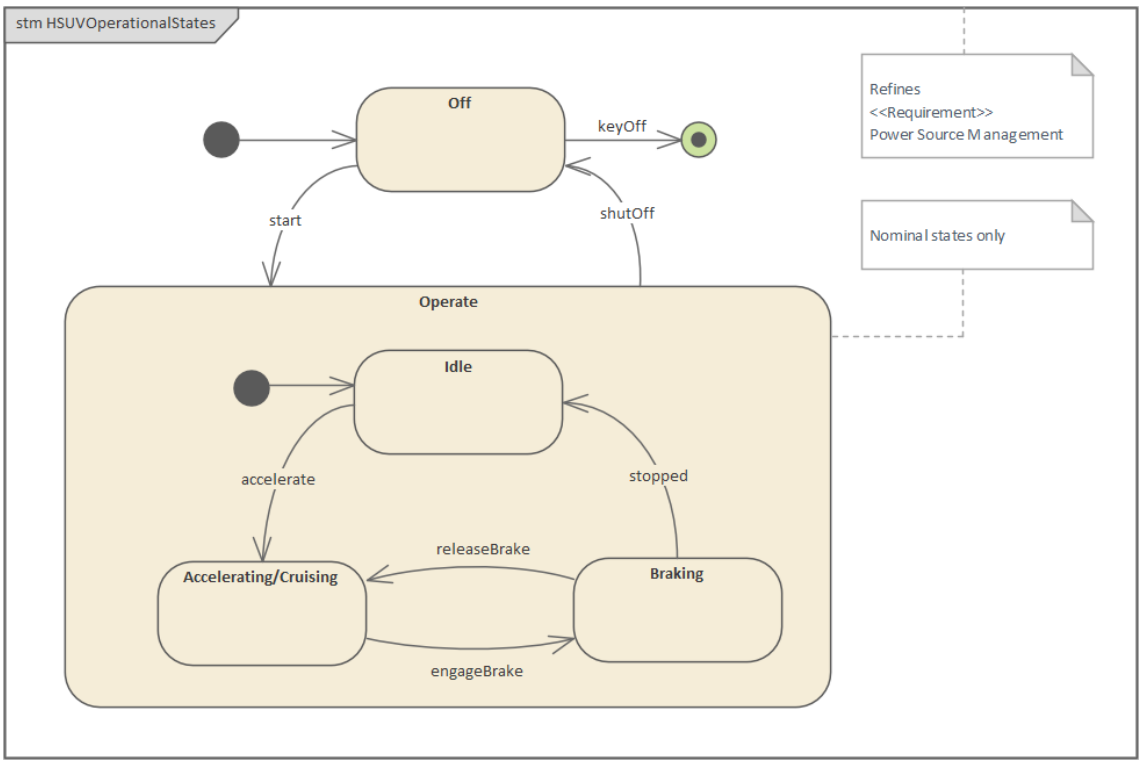
- Recursion
- Call From Recursion

Notes

The Lifelines on a Sequence diagram must be Objects (even though you can drop elements themselves as Lifelines onto a Sequence diagram, using Objects is more compliant as a UML construct), so when you drop a Block onto the Sequence diagram, in the 'Paste Element' dialog select the 'as Instance of Element (Object)' option. This creates a new object in the diagram's parent Package, based on the selected Block element. You then create the Messages between the Objects.

SysML StateMachine Diagram

A StateMachine diagram is an ideal vehicle for presenting information about the lifetime of a system element such as a Block, which might have complex behavior and might have life cycles that are difficult to understand. The diagram can be used to describe the important conditions (States) that an entity might pass through during its lifetime or life cycles. Typically, only entities that have important stages in their lifetime are modeled with StateMachine diagrams. The entity is said to transition from one State to another as defined by the StateMachine. Triggers and Events can be described that allow the State transition to occur, and Guards can be defined that restrict the change of State. Each State can define the behaviors that occur on entry to the State, while existing in the State, and on exit from the State.



Elements

The main element types that can appear in StateMachine diagrams are:

- State
- StateMachine
- Initial
- Final
- Choice
- Junction
- Entry
- Exit
- Terminate
- History
- Fork and Join

The main connector types that can appear in StateMachine diagrams are:

- Transition
- Dependency

Tools

A wide variety of tools are available for working with StateMachine diagrams, in addition to the StateMachine diagram itself. These include:

- State Table Editor - which allows the StateMachine diagram to be visualized in a table that - for some analysts - is easier to understand than a diagram; it contains the same information as the diagram and can be viewed in a number of different ways
- Dynamic Simulation - which allows processing through StateMachines to be visualized, showing how an entity transitions from one State to another
- Executable StateMachines - which, as well as utilizing the simulation engine and allowing StateMachines to be visualized, provide a complete language-specific implementation that can form the behavioral 'engine' for multiple software products on multiple platforms

Learn more

- SysML StateMachine Toolbox
- Executable StateMachines
- Dynamic Simulations
- Code Generation - StateMachines
- StateMachine Modeling For HDLs

SysML Toolboxes

Enterprise Architect's support for SysML provides Diagram Toolbox pages for the nine types of SysML diagram, which you can access through the 'Find Toolbox Item' dialog. If you enable SysML as the active technology, you can also open the SysML Toolbox pages by default.

These sets of Toolbox pages are available:

- 'Activity' contains the constructs required to construct SysML Activity models
- 'Block Definition' contains the constructs required to design SysML Blocks, ConstraintBlocks, data and value types
- 'Interaction' contains the constructs required to construct SysML interactions and Sequence diagrams
- 'Internal Block' contains the constructs required to design SysML Block compositions within Internal Block diagrams
- 'Model Elements' contains the constructs required to build SysML models, Package structures and views
- 'Parametrics' contains the constructs required to construct SysML Parametric diagrams using ConstraintBlocks
- 'Requirements' contains the constructs required to build SysML Requirements models
- 'StateMachine' contains the constructs required to build SysML StateMachines
- 'Use Case' contains the constructs required to build SysML Use Case models

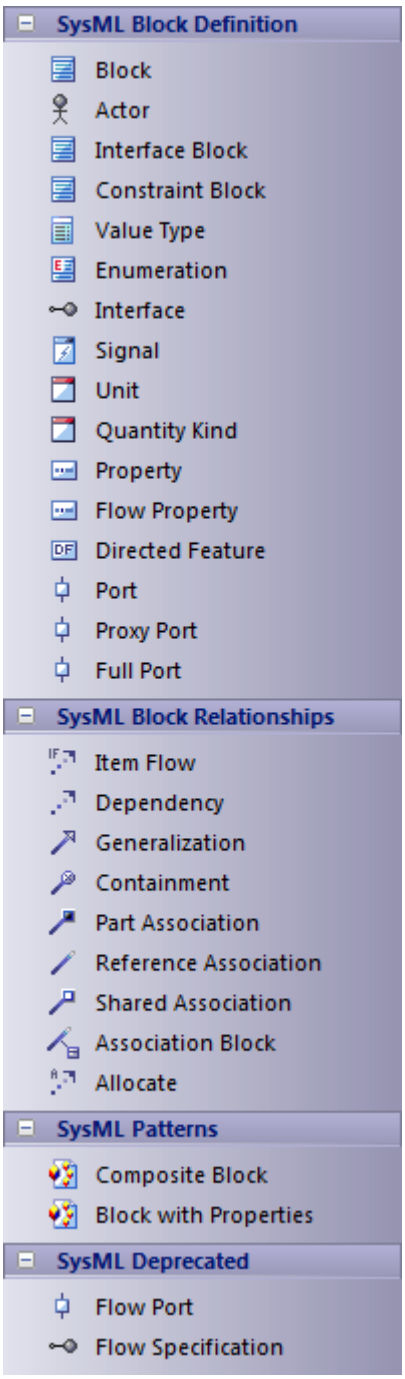
With the 'Model Elements' pages there is a set of SysML Common elements and relationships; these are also provided with the other 'SysML' Toolbox pages if the active technology is set on the Default Tools toolbar to 'SysML 1.1', 'SysML 1.2', 'SysML 1.3', 'SysML 1.4' or 'SysML 1.5'.

For details, see the Help topic for each set of SysML Toolbox pages.

SysML Block Definition Toolbox

When you are constructing SysML models, you can populate the Block Definition diagrams using the icons on the SysML Block Definition pages of the Diagram Toolbox.

You can also generate Property elements on the Block, based on the Association relationships created for the Block element. These Properties (or Parts) are initially created in the Browser window, but you can quickly render them on the Internal Block diagram for the Block.



Access

On the Diagram Toolbox, click on  to display the 'Find Toolbox Item' dialog and specify 'SysML n.n Block

Definition' (whichever version you are using).

Ribbon	Design > Diagram > Toolbox
Keyboard Shortcuts	Ctrl+Shift+3

SysML Block Definition Objects

Item	Action
Block	Defines a composite system entity in SysML.
Actor	Represents a user that interacts with one or more SysML systems.
Interface Block	A specialized kind of Block that has no behaviors or internal parts, which is used to type Proxy Ports.
Constraint Block	Defines a composite constraint as a system of parametric equations.
ValueType	Defines a SysML quantity, expressed as a measurable dimension with specific units.
Enumeration	Defines a data type as a set of symbols or values.
Interface	Defines an element that describes a specification of an interaction point with properties and methods.
Signal	Defines a SysML message, containing attributes, exchanged between system blocks in an interaction.
Instance Specification	Defines an object or instance of a block.
Unit	Represents a standard unit of measure in SysML.
QuantityKind	Identifies a measurable quantity in SysML.
Property	Describes the decomposition of a SysML Block in the context of its whole, using instances of reusable SysML Blocks.
Flow Property	Creates a single kind of Flow element that can flow to or from a Block.
Directed Feature	Generates an operation if dropped on a Block, or a DirectedFeature (Property) element if dropped on the diagram, both of which indicate that the associated Block either owns (provided) the feature, uses (required) the feature owned by another Block, or both (providedrequired). The direction value is defined in the 'SysML1.n' tab or 'Tags' tab of the Properties window for the element or the operation.
Port	Describes a structural interaction point of a SysML Block, which in turn connects

	interacting parts of a block.
Proxy Port	Exposes features of the owning Block or its internal parts.
Full Port	Specifies an element of the system separate from the owning Block or its internal parts.

SysML Block Definition Relationships

Item	Action
Item Flow	Specifies the items that flow across a connector in an interaction point. Used in the same way as UML Information Flows. See Using Information Flows
Dependency	Establishes a traceable relationship describing how one element is dependant upon another.
Generalization	Describes an element as a specialized descendant of another element, containing additional properties and behavior.
Containment	Graphically displays ownership of one element within a parent element.
Part Association	Describes the characteristics of a connection between a SysML Block and its internal parts, such as the multiplicity and type.
Reference Association	Describes the characteristics of a connection between separate SysML Blocks, such as the multiplicity and type.
Shared Association	Describes the characteristics of a common connection between SysML Blocks, such as the multiplicity and type.
Association Block	Describes an Association connector that is defined by a SysML Block.
Allocate	Connects model elements to formalize a refinement of behavior, structure, constraints or design expectations.
Connector	Structural connection between Properties and Ports.

SysML Patterns

Item	Action
Composite Block	A Pattern that creates a Composite Block, made up of Blocks related by Aggregation relationships.

Block with Properties	A Pattern that creates a Composite Block made up of nested parts.

SysML Deprecated

Item	Action
Flow Port	Describes what flows in and out of interacting SysML Blocks. This element type is deprecated. Instead create a Port that is typed by an Interface Block that owns Flow Properties.
Flow Specification	Defines a set of flow properties that correspond to individual pieces of a common interaction point. The element type is deprecated. Instead create an Interface Block that owns Flow Properties.

SysML Internal Block Toolbox

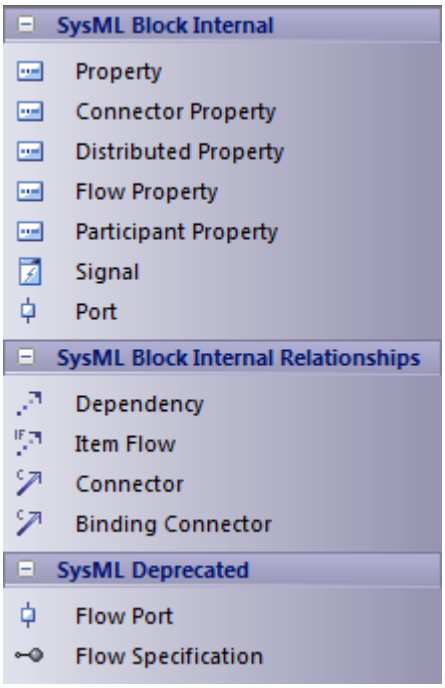
When you are constructing SysML models, you can populate the Internal Block diagrams using the icons on the 'SysML Block Internal' pages of the Diagram Toolbox.

The Block that owns the Internal Block diagram is automatically represented by a diagram frame enclosing the Internal Block diagram elements. You can:

- Hide the frame by right-clicking on the diagram and selecting the 'Hide Diagram Frame' option (and show it again by selecting the 'Show Diagram Frame' option); if the diagram frame is shown, the 'Diagram Frame' settings on the 'Start > Appearance > Preferences > Preferences > Diagram' page will be ignored
- Make the frame selectable to move or resize it, by right-clicking on it and selecting the 'Selectable' option
- Create Ports on the frame and create connectors between them and any internal structural elements
- Generate Property elements on the diagram inside the Block frame, based on the Associations that the Block element has on the Block diagram

If set to non-selectable, the frame will auto-resize to fit the bounds of the diagram, expanding from its default size but not shrinking smaller.

Note that diagrams showing Diagram Frames applied using release 14.0 or later of Enterprise Architect will draw the parent object on the diagram when opened using a release of Enterprise Architect earlier than release 14.0.



Access

On the Diagram Toolbox, click on  to display the 'Find Toolbox Item' dialog and specify 'SysML n.n Internal Block' (whichever version you are using).

Ribbon	Design > Diagram > Toolbox
Keyboard Shortcuts	Ctrl+Shift+3

SysML Block Internal

Item	Description
Property	Describes the decomposition of a SysML Block in the context of its whole using instances of reusable SysML Blocks.
Connector Property	Creates an instance of an Association Block.
Distributed Property	Creates a stereotype of Property, to apply a probability distribution to the values of the property.
Flow Property	Creates a single kind of Flow element that can flow to or from a block.
Participant Property	Creates the end of a connector owned by an Association Block.
Directed Feature	Creates a Feature that might be required, provided or both.
Adjunct Property	Creates a Property for which the value is constrained to the value of a connector typed by an Association Block, Call Action, Object Node, variable, parameter, Interaction Use or SubMachine State.
Bound Reference	Creates a Property with the <<BoundReference>> stereotype. Such properties will have binding connectors to highlight their use as constraining other properties.
End Path Multiplicity	Creates a Property with the EndPathMultiplicity stereotype. Such properties will be related by redefinition to properties that have BoundReference applied.
Classifier Behavior Property	Creates a Property with the ClassifierBehaviorProperty stereotype. Such properties will constrain their values to be the executions of classifier behaviors.
Signal	Defines a SysML message, containing attributes, exchanged between system Blocks in an interaction.
Port	Describes a structural interaction point of a SysML Block which, in turn, connects between interacting parts of a block.

SysML Block Internal Relationships

Item	Description
Dependency	Establishes a traceable relationship describing how one element is dependant upon another.
Item Flow	Specifies the items that flow across a connector in an interaction point. Used in the same way as UML Information Flows.
Connector	Establishes Communication links between parts.

Binding Connector	Establishes a connection between two Parts in a system decomposition. If the Parts are different, the system provides an option to synchronize them.

SysML Deprecated

Item	Description
Flow Port	Describes what flows in and out of interacting SysML Blocks. This element type is deprecated. Instead create a Port that is typed by an Interface Block that owns Flow Properties.
Flow Specification	Defines a set of flow properties that correspond to individual pieces of a common interaction point. The element type is deprecated. Instead create an Interface Block that owns Flow Properties.

SysML Activity Toolbox

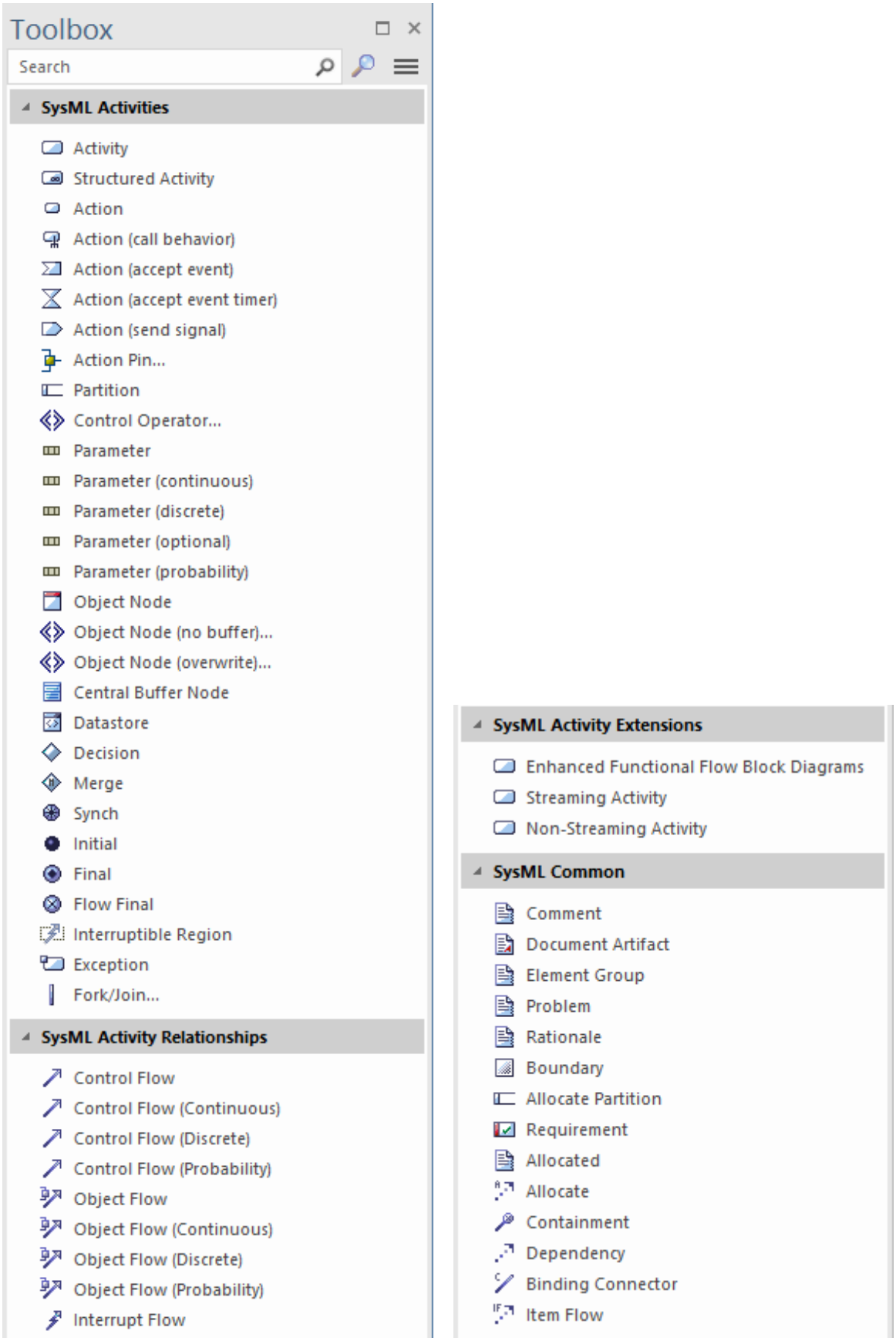
When you are constructing SysML models, you can populate the Activity diagrams using the icons on the 'SysML Activity' pages of the Diagram Toolbox.

The element that owns the Activity diagram is automatically represented by a diagram frame enclosing the Activity diagram elements. You can:

- Hide the frame by right-clicking on the diagram and selecting the 'Hide Diagram Frame' option (and show it again by selecting the 'Show Diagram Frame' option)
- Make the frame selectable to move or resize it, by right-clicking on it and selecting the 'Selectable' option
- Create structural elements (such as ActivityParameters if the owner is an Activity) on the frame and create connectors between them and other elements on the diagram

If set to non-selectable, the frame will auto-resize to fit the bounds of the diagram, expanding from its default size but not shrinking smaller.

Note that diagrams showing Diagram Frames applied using release 14.0 or later of Enterprise Architect will draw the parent object on the diagram when opened using a release of Enterprise Architect earlier than release 14.0.


Toolbox
Search
SysML Activities
Activity
Structured Activity
Action
Action (call behavior)
Action (accept event)
Action (accept event timer)
Action (send signal)
Action Pin...
Partition
Control Operator...
Parameter
Parameter (continuous)
Parameter (discrete)
Parameter (optional)
Parameter (probability)
Object Node
Object Node (no buffer)...
Object Node (overwrite)...
Central Buffer Node
Datastore
Decision
Merge
Synch
Initial
Final
Flow Final
Interruptible Region
Exception
Fork/Join...
SysML Activity Relationships
Control Flow
Control Flow (Continuous)
Control Flow (Discrete)
Control Flow (Probability)
Object Flow
Object Flow (Continuous)
Object Flow (Discrete)
Object Flow (Probability)
Interrupt Flow
SysML Activity Extensions
Enhanced Functional Flow Block Diagrams
Streaming Activity
Non-Streaming Activity
SysML Common
Comment
Document Artifact
Element Group
Problem
Rationale
Boundary
Allocate Partition
Requirement
Allocated
Allocate
Containment
Dependency
Binding Connector
Item Flow

Access

On the Diagram Toolbox, click on  to display the 'Find Toolbox Item' dialog and specify 'SysML n.n Activity' (whichever version you are using).

Ribbon	Design > Diagram > Toolbox
Keyboard Shortcuts	Ctrl+Shift+3

SysML Activities

Item	Action
Activity	Defines a SysML Block of executable behavior as a UML Activity.
Structured Activity	Defines a SysML Block of executable behavior as a UML Structured Activity.
Action	Declares a unit of execution in an Activity as a UML Action.
Action (call behavior)	Declares a unit of execution that calls another behavior.
Action (accept event)	Declares a unit of execution that accepts an event raised by the system.
Action (accept event timer)	Declares a unit of execution that accepts an event raised by a time epoch.
Action (send signal)	Declares a unit of execution that sends a signal as an event.
Action Pin	Defines the data values passed out of and into an Action. See also: Action Pin
Partition	Creates an Activity Partition to group execution elements according to the node responsible for their execution.
Control Operator	Controls the execution of an Activity.
Parameter	Provides access to input and output objects within the Activity.
Parameter (continuous)	Defines a parameter with a rate of flow where the increment of time between items approaches zero.
Parameter (discrete)	Defines a parameter with a rate of flow where the increment of time between items is non-zero.
Parameter (optional)	Defines a parameter whose contents are optional in the Activity's execution.
Parameter (probability)	Tags a parameter with the probability of the parameter being in use in the Activity.
Object Node	Declares a variable in the Activity, typed by a ValueType, DataType or Block.
Object Node (no buffer)	Declares an ObjectNode in an Activity that discards unconsumed tokens.

Object Node (overwrite)	Declares an ObjectNode in an Activity that overwrites tokens.
Central Buffer Node	Declares an ObjectNode that stores tokens for consumption throughout the Activity.
Datastore	Defines permanently stored data. See also: Datastore
Decision	Creates a branch of control in an Activity, based on a decision.
Merge	Merges two or more Activity control branches.
Synch	Establishes a rendezvous point for two or more Activity flows, in order to synchronize their execution in the Activity.
Initial	Declares the start of an Activity's execution.
Final	Declares the end of an Activity's execution, and the termination of the Activity.
Flow Final	Declares the end of an Activity's execution path without terminating the Activity.
Interruptible Region	Groups a subset of an Activity into a common execution context.
Exception	Declares a node of execution that happens outside the normal flow of execution of an Activity.
Fork/Join	Simultaneously branches / joins a set of Control or Object Flows.

SysML Activity Relationships

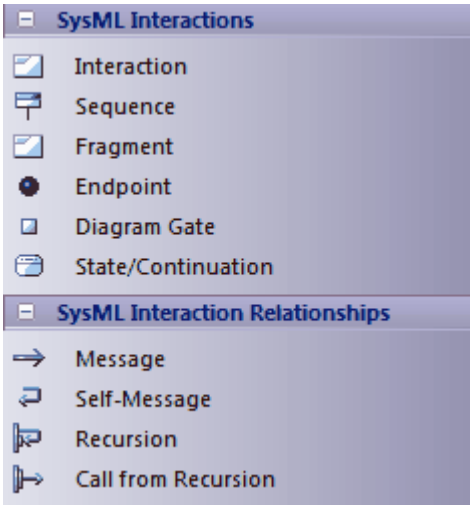
Item	Action
Control Flow	Establishes a flow of logic between two Activity nodes.
Control Flow (Continuous)	Declares a continuous control flow.
Control Flow (Discrete)	Declares a discrete control flow.
Control Flow (Probability)	Tags a control flow with a probability of the likelihood of the flow's traversal.
Object Flow	Establishes a flow of objects (data) between two Activity nodes.
Object Flow (Continuous)	Declares a continuous object flow.
Object Flow (Discrete)	Declares a discrete object flow.
Object Flow (Probability)	Tags an object flow with the probability of the flow's traversal.
Interrupt Flow	Declares a control flow that interrupts flow within a Region.

SysML Activity Extensions

Item	Action
Enhanced Functional Flow Block Diagrams	Action: Declares an Activity used to contain an Enhanced Functional Flow Block diagram (EFFBD).
Streaming Activity	Declares an Activity where the flow of tokens passes through its parameters continuously throughout the Activity's execution.
Non-Streaming Activity	Declares an Activity where the flow of tokens passes through its parameters at the start of the Activity's execution.

SysML Interaction Toolbox

When you are constructing SysML models, you can populate Interaction and Sequence diagrams using the icons on the SysML Interaction pages of the Diagram Toolbox.



Access

On the Diagram Toolbox, click on  to display the 'Find Toolbox Item' dialog and specify 'SysML n.n Interaction' (whichever version you are using).

Ribbon	Design > Diagram > Toolbox
Keyboard Shortcuts	Ctrl+Shift+3

SysML Interaction Objects

Item	Action
Interaction	Defines a SysML Block of executable behavior as a UML Interaction.
Sequence	References an instance of a SysML Block as a Lifeline in the Interaction.
Fragment	Declares a portion of an interaction as a group with specific behavior semantics.
Endpoint	Creates an entry or exit point for the Interaction.
Diagram Gate	Creates an endpoint for the interaction, which bridges between nested interactions.
State/Continuation	Constrains the Interaction with assertions of the state that the lifeline is expected to be in.

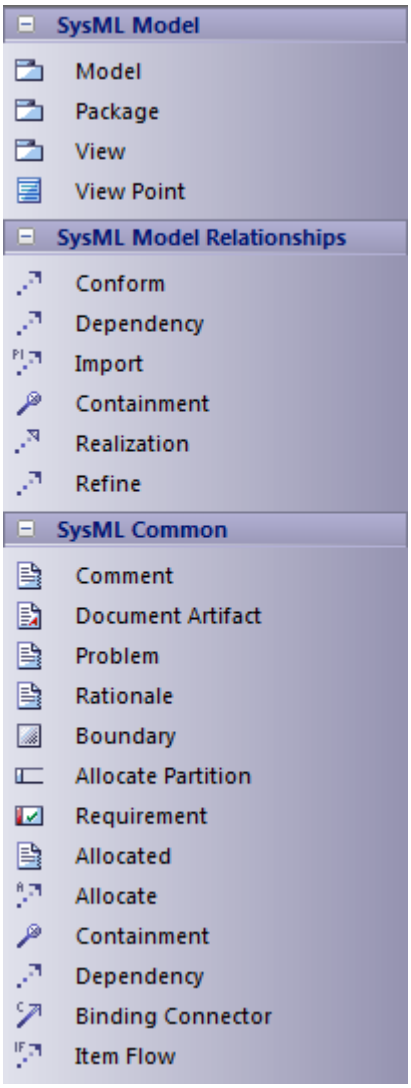
SysML Interaction Relationships

Item	Action
Message	Describes a message exchange between two Lifelines in an Interaction.
Self-Message	Describes a message exchange between a Lifeline and itself in an Interaction.
Recursion	Describes a recursive message exchange between a Lifeline and itself in an Interaction.
Call from Recursion	Describes a message exchange between two Lifelines within a recursive exchange.

SysML Model Toolbox

When you are constructing SysML models, you can populate the diagrams with Model, Package and View elements using the icons on the 'SysML Model' pages of the Diagram Toolbox.

The SysML Model toolbox includes a page of SysML Common element and relationship icons. You can add this page to all Diagram Toolboxes so that it is always available regardless of what type of diagram you are using; to do this, set the SysML 1.5 Technology to 'Active' ('Specialize > Technologies > Manage Technology: SysML 1.5 : Set Active' ribbon path).



Access

On the Diagram Toolbox, click on  to display the 'Find Toolbox Item' dialog and specify 'SysML n.n Model Elements' (whichever version you are using).

Ribbon	Design > Diagram > Toolbox
Keyboard Shortcuts	Ctrl+Shift+3

SysML Model Element Objects

Item	Description
Model	Creates a Package containing a SysML Model.
Model Library	Creates a Package containing a SysML Model Library.
Package	Groups model constructs in a single unit of containment.
View	Creates a stereotyped Class that defines a SysML View of a system, from the perspective of a SysML View Point.
View Point	Creates a stereotyped Class that defines a SysML View Point, which specifies the rules and conventions for the construction and use of Views.
Stakeholder	Creates a stereotyped Class that defines a SysML Stakeholder.

SysML Model Relationships

Item	Description
Conform	Establishes a conformance dependency of a View to the defining View Point.
Dependency	Establishes a traceable relationship describing how one element is dependant upon another.
Import	Represents a reuse of elements from one model Package in another.
Containment	Graphically displays ownership of one element within a parent one.
Realization	Identifies a design fulfillment of a specification between elements.
Refine	Represents a refinement of one element by another.
Expose	Relates a View to another model element.

SysML Common

Item	Description
Comment	Creates a textual annotation that can be attached to a set of elements of any other

	type. The attachment is created separately, using a Notelink connector.
Document Artifact	Attaches a Linked Document to the diagram by associating this element with the document.
Element Group	Creates a textual annotation that indicates how many model elements it is attached to.
Problem	A stereotyped Comment that documents the failure of model elements to satisfy a requirement.
Rationale	A stereotyped Comment that documents the justification for decisions.
Boundary	Defines a conceptual boundary, to visually group logically related elements.
Allocate Partition	A stereotyped Activity Partition that contains elements deemed to be allocated to the classifier of the partition.
Requirement	Specifies the capabilities of the system, or the conditions that it should satisfy.
Allocated	A stereotyped Comment that defines the source element being allocated to the target element in an Allocate relationship.
Allocate	A stereotyped Abstraction that relates model elements to formalize a refinement of behavior, structure, constraints or design expectations. The Allocate relationship points from the element being allocated to the element that is the target of the allocation. The system provides an 'Allocations' search that lists all Allocate abstractions in tabular format, showing the 'To' and 'From' elements. Select the 'Explore > Search > Model' ribbon option, then select 'SysML 1.5' as the Search Category; 'Allocations' defaults as the Search Type. If a SysML Block element has any Allocate relationships to or from elements that are not visible on the same diagram, those elements can be listed in 'AllocatedTo' and 'AllocatedFrom' compartments of the Block element on the diagram. Press Ctrl+Shift+Y to display the 'Compartment Visibility' dialog and select the 'Allocatedto' and/or 'Allocatedfrom' checkboxes. Alternatively, you can list the hidden elements on a linked Note. Create a Note element on the diagram and link it to the visible Block element with a Notelink connector. Right-click on the connector, select the 'Link this Note to an Element feature' option and, on the 'Link note to element feature' dialog, click on the drop-down arrow on the 'Feature Type' field and select 'AllocatedTo' or 'AllocatedFrom'. Click on the OK button; the names of the linked elements are now displayed in the Notes element. (To show both 'To' and 'From' elements, create a separate Note for each type.) For either compartments or Notes, you should save the diagram just before setting up the facility, and possibly reload the diagram to activate the facility. If there are no Allocate relationships, or the related elements are on the diagram, the options are not available.
Containment	Graphically displays ownership of an element within a parent element.
Dependency	Establishes a traceable relationship describing how one element is dependant upon another.

Binding Connector	A stereotyped Connector that establishes a connection between two Parts in a system decomposition. If the Parts are different, the system provides an option to synchronize them.
Item Flow	A stereotyped Information Flow that specifies the items that flow across a connector in an interaction point.

SysML Parametrics Toolbox

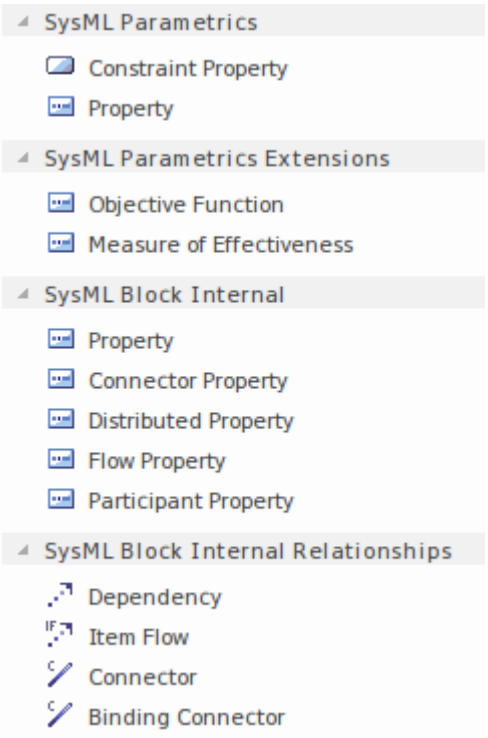
When you are constructing SysML models, you can populate the SysML Parametric diagrams with constraint blocks, using the icons on the 'SysML Parametrics' pages of the Diagram Toolbox.

The Block that owns the Parametric diagram is automatically represented by a diagram frame enclosing the Parametric diagram elements. You can:

- Hide the frame by right-clicking on the diagram and selecting the 'Hide Diagram Frame' option (and show it again by selecting the 'Show Diagram Frame' option)
- Make the frame selectable to move or resize it, by right-clicking on it and selecting the 'Selectable' option
- Create Ports and Parts on the frame and create connectors between them

If set to non-selectable, the frame will auto-resize to fit the bounds of the diagram, expanding from its default size but not shrinking smaller.

Note that diagrams showing Diagram Frames applied using release 14.0 or later of Enterprise Architect will draw the parent object on the diagram when opened using a release of Enterprise Architect earlier than release 14.0.



Access

On the Diagram Toolbox, click on  to display the 'Find Toolbox Item' dialog and specify 'SysML n.n Parametrics' (whichever version you are using).

Ribbon	Design > Diagram > Toolbox
Keyboard Shortcuts	Ctrl+Shift+3

SysML Parametrics Objects

Item	Description
ConstraintProperty	Instantiates a Constraint Block for use in a Parametric diagram.
Property	Defines a SysML property typed by a DataType, ValueType or Block.

SysML Parametrics Extensions

Item	Description
Objective Function	Defines a SysML Constraint Block for use as an objective function to evaluate Measures of Effectiveness (MOEs).
Measure of Effectiveness	Defines a SysML property for use as a Measure of Effectiveness (MOE).

SysML Block Internal

Item	Description
Property	Describes the decomposition of a SysML Block in the context of its whole using instances of reusable SysML Blocks.
Connector Property	Creates an instance of an Association Block.
Distributed Property	Creates a stereotype of Property, to apply a probability distribution to the values of the property.
Flow Property	Creates a single kind of Flow element that can flow to or from a block.
Participant Property	Creates the end of a connector owned by an Association Block.

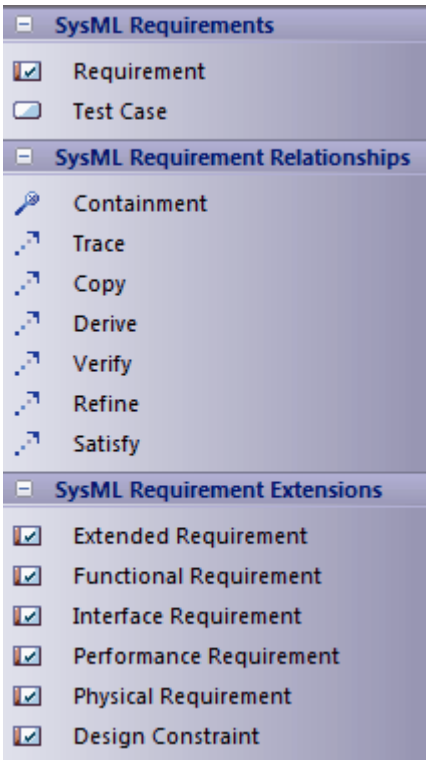
SysML Block Internal Relationships

Relationship	Description
Dependency	Establishes a traceable relationship describing how one element is dependant upon another.
Item Flow	Specifies the items that flow across a connector in an interaction point. Used in the same way as UML Information Flows.

Connector	Establishes Communication links between parts.
Binding Connector	Establishes a connection between two Parts in a system decomposition. If the Parts are different, the system provides an option to synchronize them.

SysML Requirements Toolbox

When you are constructing SysML models, you can populate the Requirements diagrams using the icons on the SysML Requirements pages of the Diagram Toolbox.



Access

On the Diagram Toolbox, click on  to display the 'Find Toolbox Item' dialog and specify 'SysML n.n Requirements' (whichever version you are using).

Ribbon	Design > Diagram > Toolbox
Keyboard Shortcuts	Ctrl+Shift+3

SysML Requirement Objects

Page	Item
Requirement	Specifies the capabilities of the system, or the conditions that it should satisfy.
Test Case	Describes the verification of a Requirement through methods of inspection, analysis, demonstration or testing.

SysML Requirement Relationships

Item	Description
Containment	Graphically displays ownership of one element within a parent element.
Trace	Declares a trace relationship between a SysML Requirement and another SysML element.
Copy	Declares a copy of one SysML Requirement by another.
Derive	Derives a SysML Requirement from another.
Verify	Declares a verification of a SysML Requirement by another SysML element.
Refine	Declares a refinement of a SysML Requirement by another SysML element.
Satisfy	Declares that the SysML Requirement is satisfied by another SysML element.

SysML Requirement Extensions

Item	Description
Extended Requirement	Extends a SysML Requirement with additional Tag properties.
Functional Requirement	Declares a SysML Requirement that describes the operation, or behavior, that the system must perform.
Interface Requirement	Declares a SysML Requirement that describes how the system connects, or interfaces with, other systems.
Performance Requirement	Declares a SysML Requirement that describes how the system performs against defined capabilities or conditions.
Physical Requirement	Declares a SysML Requirement that describes the physical characteristics, or physical constraints, of the system.
Design Requirement	Declares a SysML Requirement that specifies a constraint on the implementation of the system.

Notes

- SysML Requirements contain the Tagged Values 'Text' and 'ID', the values of which are not immediately visible in the 'Tags' tab of the Properties window; you can see the values more easily if you have the Summary window open

(press Ctrl+6 or click on the 'Start > All Windows > Design > Explore > Summary' ribbon option) when you click on these elements

SysML StateMachine Toolbox

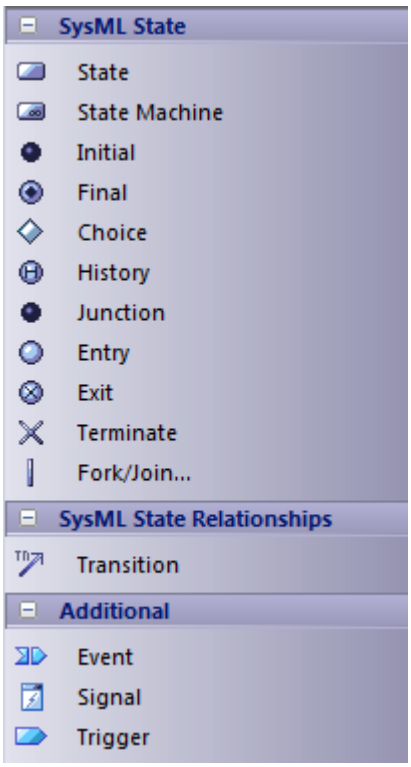
When you are constructing SysML models, you can populate the StateMachine diagrams using the icons on the 'SysML StateMachine' pages of the Diagram Toolbox.

The Block that owns the StateMachine diagram is automatically represented by a diagram frame enclosing the StateMachine diagram elements. You can:

- Hide the frame by right-clicking on the diagram and selecting the 'Hide Diagram Frame' option (and show it again by selecting the 'Show Diagram Frame' option)
- Make the frame selectable to move or resize it, by right-clicking on it and selecting the 'Selectable' option
- Create Ports and Parts on the frame and create connectors between them

If set to non-selectable, the frame will auto-resize to fit the bounds of the diagram, expanding from its default size but not shrinking smaller.

Note that diagrams showing Diagram Frames applied using release 14.0 or later of Enterprise Architect will draw the parent object on the diagram when opened using a release of Enterprise Architect earlier than release 14.0.



Access

On the Diagram Toolbox, click on  to display the 'Find Toolbox Item' dialog and specify 'SysML n.n StateMachine' (whichever version you are using).

Ribbon	Design > Diagram > Toolbox
Keyboard Shortcuts	Ctrl+Shift+3

SysML StateMachine Objects

Item	Description
State	Declares a significant condition in the life of a SysML Block within its StateMachine.
StateMachine	Describes the life-cycle behavior of a SysML Block in terms of its states and transitions.
Initial	Declares the starting state of the StateMachine.
Final	Declares the ending state of the StateMachine, and its completion.
Choice	Declares a Junction with a mandatory 'else' transition.
History	Represents the last active State of the StateMachine prior to its interruption.
Junction	Declares a decision point at which a Transition branches out into multiple guarded, alternative paths.
Entry	Declares an Entry point between StateMachines, SubstateMachines and Regions.
Exit	Declares an Exit point between StateMachines, SubstateMachines and Regions.
Terminate	Declares a termination State in which the StateMachine no longer operates.
Fork/Join	Simultaneously branches and joins a set of Transitions.

SysML State Relationships

Item	Description
Transition	Establishes a life-cycle path between one State and another, based on its operational conditions.

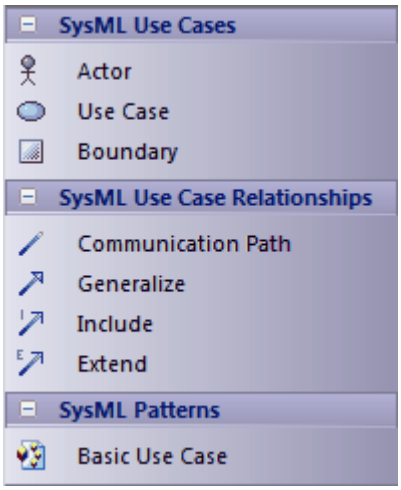
Additional

Item	Description
Event	Depicts the action of sending a signal.
Signal	A specification of Send request instances communicated between objects.
	Indicates an event that initiates an action (and might arise from completion of a

Trigger	previous action).

SysML Use Case Toolbox

When you are constructing SysML models, you can populate the Use Case diagrams using the icons on the 'SysML Use Cases' pages of the Diagram Toolbox.



Access

On the Diagram Toolbox, click on  to display the 'Find Toolbox Item' dialog and specify 'SysML n.n Use Cases' (whichever version you are using).

Ribbon	Design > Diagram > Toolbox
Keyboard Shortcuts	Ctrl+Shift+3

SysML Use Case Objects

Item	Description
Actor	Represents a user that interacts with one or more SysML systems.
Use Case	Describes the expected functionality of a system as a UML Use Case.
Boundary	Graphically binds elements in a diagram within a border.

SysML Use Case Relationships

Item	Description

Communication Path	Declares which Actors perform in the Use Case.
Generalize	Describes an element as a specialized descendant of another element, containing additional properties and behavior.
Include	Describes one Use Case as a subset of another.
Extend	Describes one Use Case as an extension of another.

SysML Patterns

Item	Description
Basic Use Case	A Pattern that creates a typical simple Use Case diagram of Actor, Use Case and System Boundary elements.

Migrate SysML Model to Later SysML Version

Enterprise Architect provides a useful feature to migrate a model from one version of SysML to the next. A model (or part of a model) created in an older version of the SysML Technology can be migrated to the next version using the Automation Interface. This function updates the Tagged Values and, if required, stereotypes to the later version for all elements, attributes, connectors and diagrams under the selected Package or element.

There is no facility to migrate a model from SysML 1.4 to SysML 1.5, because the two releases are functionally the same.

Migrate SysML 1.3 to SysML 1.4

Enterprise Architect's support for SysML 1.4 has a built-in script for migrating 1.3 models to 1.4.

You must enable both the SysML 1.3 Technology and the SysML 1.4 Technology (select the 'Specialize > Technologies > Manage Technology' ribbon option, and select the 'Enable' checkbox for each of the two Technologies).

1. Select the SysML 1.3 Package in the Browser window.
2. Open the Scripting window and open the SysML 1.4 script group.
3. Execute the Migrate script.

Migrate SysML 1.2 to SysML 1.3

Enterprise Architect's support for SysML 1.3 has a built-in script for migrating 1.2 models to 1.3.

You must enable both the SysML 1.2 Technology and the SysML 1.3 Technology (select the 'Specialize > Technologies > Manage Technology' ribbon option, and select the 'Enable' checkbox for each of the two Technologies).

1. Select the SysML 1.2 Package in the Browser window.
2. Open the Scripting window and open the SysML 1.3 script group.
3. Execute the Migrate script.

Migrate SysML 1.1 to SysML 1.3

Firstly, work through the steps in *Script for Migrating SysML 1.1 to SysML 1.2*.

Then work through the steps in *Migrate from SysML 1.2 to SysML 1.3*.

Script for Migrating SysML 1.1 to SysML 1.2

Run this VB script, which calls the Migrate() function to migrate the SysML 1.1 Package or element to SysML 1.2:

```
Sub MigrateElement (sGUID, lngPackageID)
        Dim proj as EA.Project
        set proj = Repository.GetProjectInterface
        proj.Migrate sGUID, "SysML1.1", "SysML1.2"

        'refresh the model
```

```
        If lngPackageID<>0 Then
                Repository.RefreshModelView (lngPackageID)
        End If
End Sub
Sub MigrateSelectedItem
        Dim selType
        Dim selElement as EA.Element
        Dim selPackage as EA.Package

        selType = GetTreeSelectedItemType
        If selType = 4 Then 'means Element
                set selElement = GetTreeSelectedObject
                MigrateElement selElement.ElementGUID, selElement.PackageID
                MsgBox "Element Migration Completed",0,"SysML Migration"
        ElseIf selType = 5 Then 'means Package
                set selPackage = GetTreeSelectedObject
                MigrateElement selPackage.PackageGUID, selPackage.PackageID
                MsgBox "Package Migration Completed",0,"SysML Migration"
        Else
                MsgBox "Select a Package or Element in the Browser window to initiate migration",0,"SysML
Migration"
        End If
End Sub
Sub Main
        MigrateSelectedItem
End Sub
Main
```

Notes

- When migrating from SysML 1.1 to SysML 1.2, the:
  - Stereotype dimension is changed to quantitykind
  - Stereotype dataType is removed from SysML 1.2
  - Tagged Value dimension is migrated to quantitykind; this applies to stereotypes <<unit>> and <<valueType>>
  - Tagged Value isConjugated in stereotype <<flowport>> is migrated to custom properties

Simple Parametric Simulation

You use the scripting function of Enterprise Architect to simulate a SysML model from a Parametric diagram, using the 'Simulation Configuration' dialog.

This facility is in addition to the full SysML Parametric Simulation also provided in Enterprise Architect. The simpler simulation has less functionality when compared to the OpenModelica-based or Simulink-based simulation, because the behavior of each Constraint Block is represented by a script that calculates outputs from a known set of inputs instead of solving for the unknowns dynamically.

Access

Context Menu	On a Parametric diagram | Right-click | SysML | Simulate Diagram

Simulate a SysML model

Step	Action
1	The 'Parameters' panel lists all of the parameters that can be assigned input. Select each of the required parameters and click on the right Arrow button to assign them as input. Parameters designated as input parameters are listed in the 'Inputs' panel on the right. There must be at least one input parameter assigned for the simulation to execute.
2	Assign a set of values for each of the designated input parameters. For each input parameter, in the 'Input Values' panel select one of the two possible value kinds: • Discrete - To enter a constant or a comma-separated range of discrete values • Range - To enter a range of values beginning at the 'From' value and ending at the 'To' value; the input values are incremented by the 'Step' value
3	Specify the classes of output value: • 'Parameters' - To output the parameters' data, select the checkbox • 'Variables' - To output the data generated within each internal variable, select the checkbox; internal variables are automatically generated by the Simulator
4	Specify how the simulation results are to be reported. The 'Output Format' panel enables you to choose how the simulation outputs the simulation data: • Plot To Graph: To plot the results on a 2-dimensional graph, select the checkbox; if you select this option, you must specify an input parameter for the plot's X Axis • Title - To enter a title for the graph, type in the title text • Output to File - To output the results to a CSV text file, select the checkbox and type or browse (click on ) for the file name
5	Click on the OK button to execute the simulation.

Notes

- Systems Modeling Language (SysML) Parametric Model Simulation is available in the Unified and Ultimate Editions of Enterprise Architect