



ENTERPRISE ARCHITECT

User Guide Series

Introducing the Metamodel Views

Author: Sparx Systems

Date: 16/10/2024

Version: 17.0

CREATED WITH  **ENTERPRISE ARCHITECT**

Table of Contents

Introducing the Metamodel Views	3
Built-in Metamodel Diagram View	5
Custom Metamodel Diagram View	9
Define Metamodel Constraints	15
Constraints on Meta-Constraint Connector	20
Metamodel Constraints and the Quick Linker	27

Introducing the Metamodel Views

Enterprise Architect includes an extremely effective and flexible system of Views of both system-defined and user-defined metamodels. The Views system provides highly focused diagrams that limit the number of elements and connections available to only the core required to achieve a specific task. For example, a Hierarchy View imposed on a Class diagram might limit the only element available to 'Class' and the only connector to 'Inheritance'.

Using the Views system to guide the modeling palette and relationships available, you will build tight and purposeful diagrams that use only the required elements within the current modeling context. Cutting out the noise and reducing the set of constructs available is a great way of making sure a design is addressing the intended purpose and avoiding extraneous elements that might negatively impact the readability and correctness of the model.

Metamodel Views

Category	Description
System	Enterprise Architect provides a wide range of built-in Metamodel Views that address numerous modeling scenarios and domains. Many of the Model Builder patterns are pre-set with a Metamodel View, and the 'Diagram Builder' dialog includes many derivative diagram views that extend and refine the capabilities of the base diagram types.
Custom	In addition to using the system-defined Metamodel based views in Enterprise Architect, it is also possible to create your own Metamodels and easily add them to the current model, where you and other modelers can then apply them to various diagrams as needed. For example, you might define a specific Metamodel set that addresses the needs of Requirements modeling in your organization, and then mandate that all Requirement diagrams use that Metamodel View.

View System Facilities

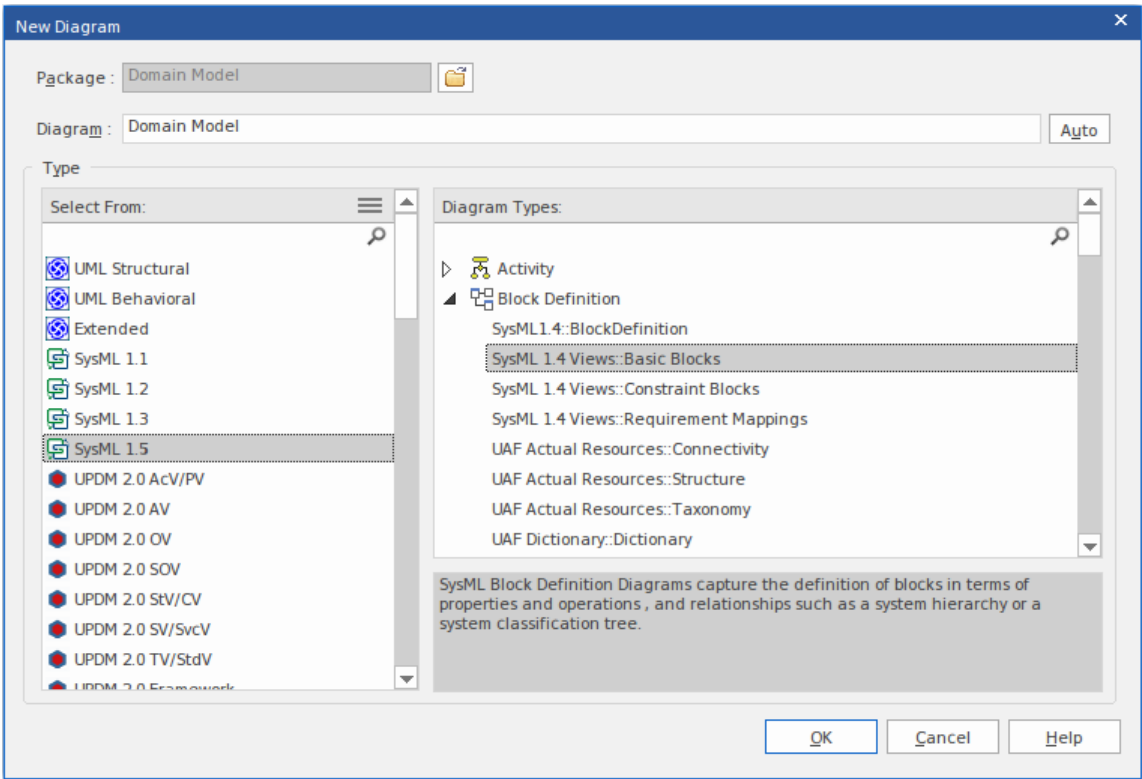
Facility	Description
Diagram Filter	In addition to limiting the available palette, the View system also allows the modeler to enable a diagram filter that will gray out any elements that are not part of the current view set. This allows the modeler to correct any parts of their model that don't meet the purpose of the selected View, or to filter out elements that are required to be there, but do not form part of the current modeling goal.
Diagram Properties	The 'Properties' dialog for a diagram includes a drop-down list of available Views for the currently selected diagram type. Selecting one of these Views will reduce the palette of constructs available and limit the entries in the Quick Linker. Modelers can easily activate a View or even remove one if necessary - the actual model content will not change.
Diagram Views	The 'Diagram Builder' dialog includes a number of different Views that offer different palette sets and focus goals for diagram types such as UML, SysML, BPMN and UAF, amongst others. If you have the goal of modeling a simple Activity diagram with no advanced features, the Simple Activity View under the UML Activity diagram section could be a better option than using the full Activity

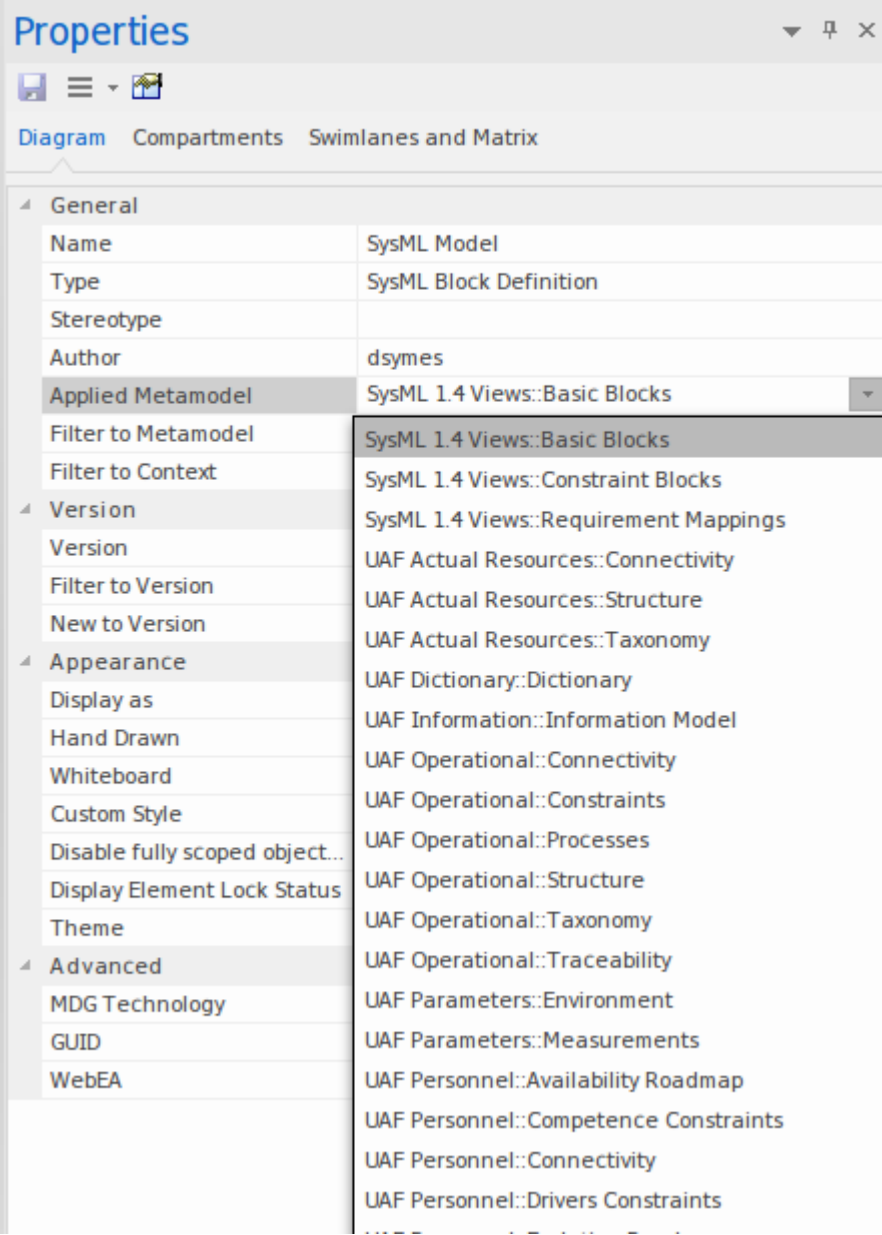
	diagram set.
--	--------------

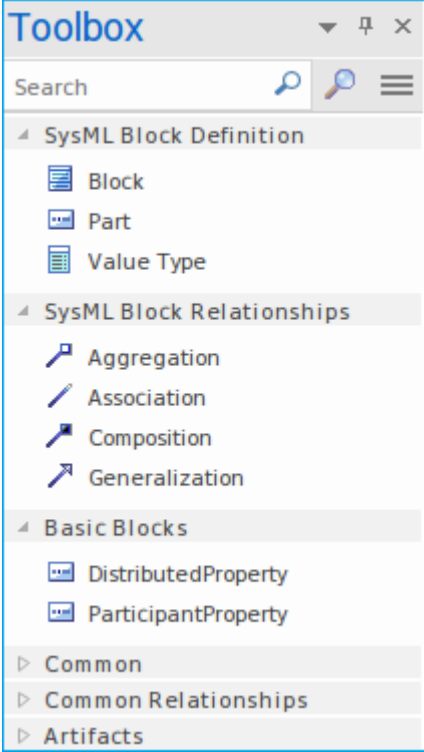
Built-in Metamodel Diagram View

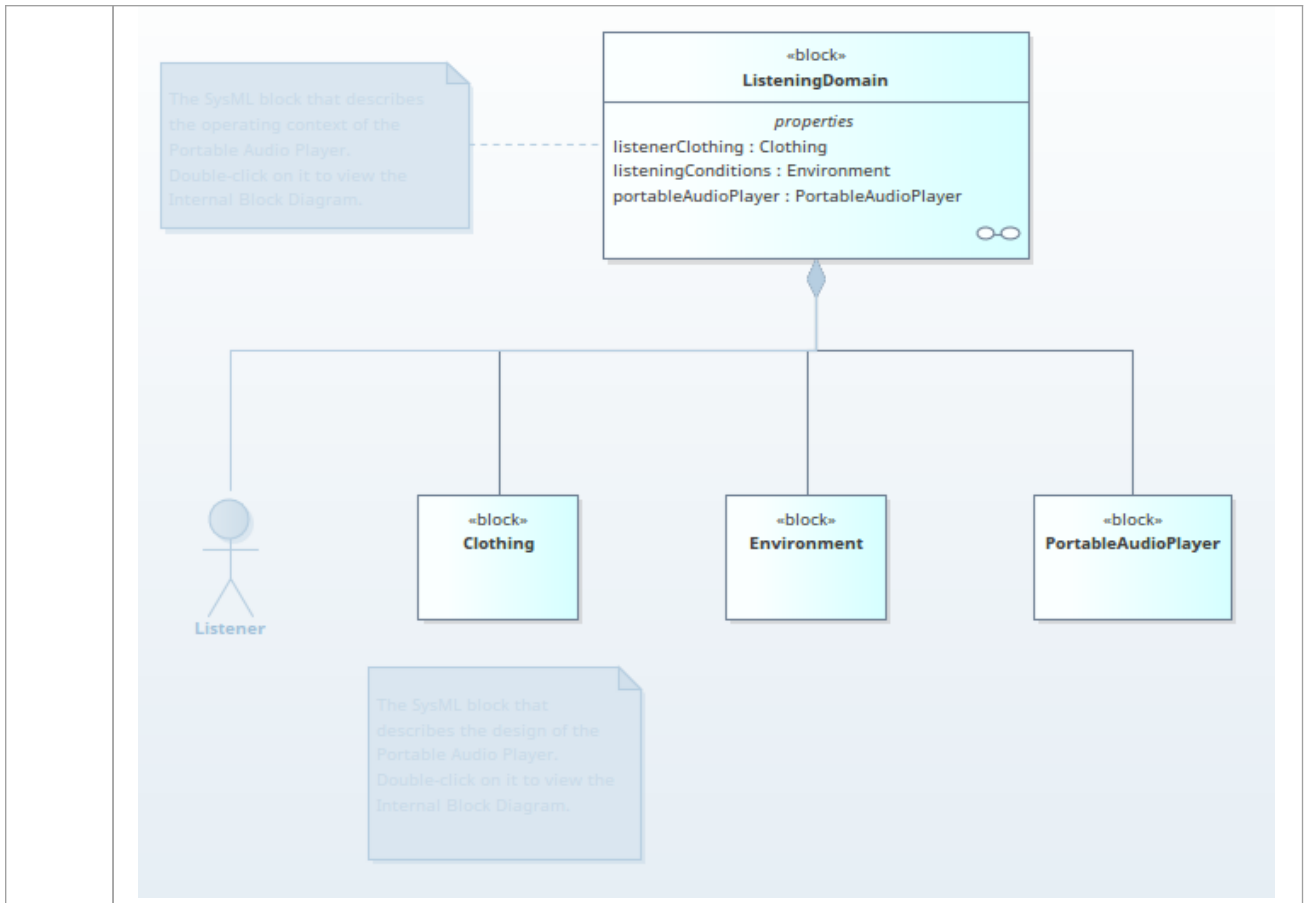
The 'New Diagram' dialog includes a number of different Views that offer different palette sets and focus goals for diagram types such as UML, SysML, BPMN and UAF, amongst others. As an example, if you have the goal of modeling a simple SysML Block Definition diagram with no advanced features, the 'Basic Blocks View' under the 'SysML 1.5 Block Definition Diagram' section might be a better option than using the full Block Definition diagram set. This example is used to provide values in the procedures in this topic.

Working with Diagram Views

Step	Action
1	<p>In the Browser window, click on the Package or element under which to place the diagram.</p> <p>Open the 'New Diagram' dialog, select 'SysML 1.4 Views:: Basic Blocks' and click on the OK button to create the diagram.</p> 
2	<p>In the Properties window for the created diagram, the 'Applied Metamodel' field will show the applied diagram View. You can also click on the drop-down arrow in this field and select another of the available diagram Views from the list.</p>

	
<p>3</p>	<p>In the Diagram Toolbox, the restricted set of elements and relationships associated with the diagram view will be visible.</p>

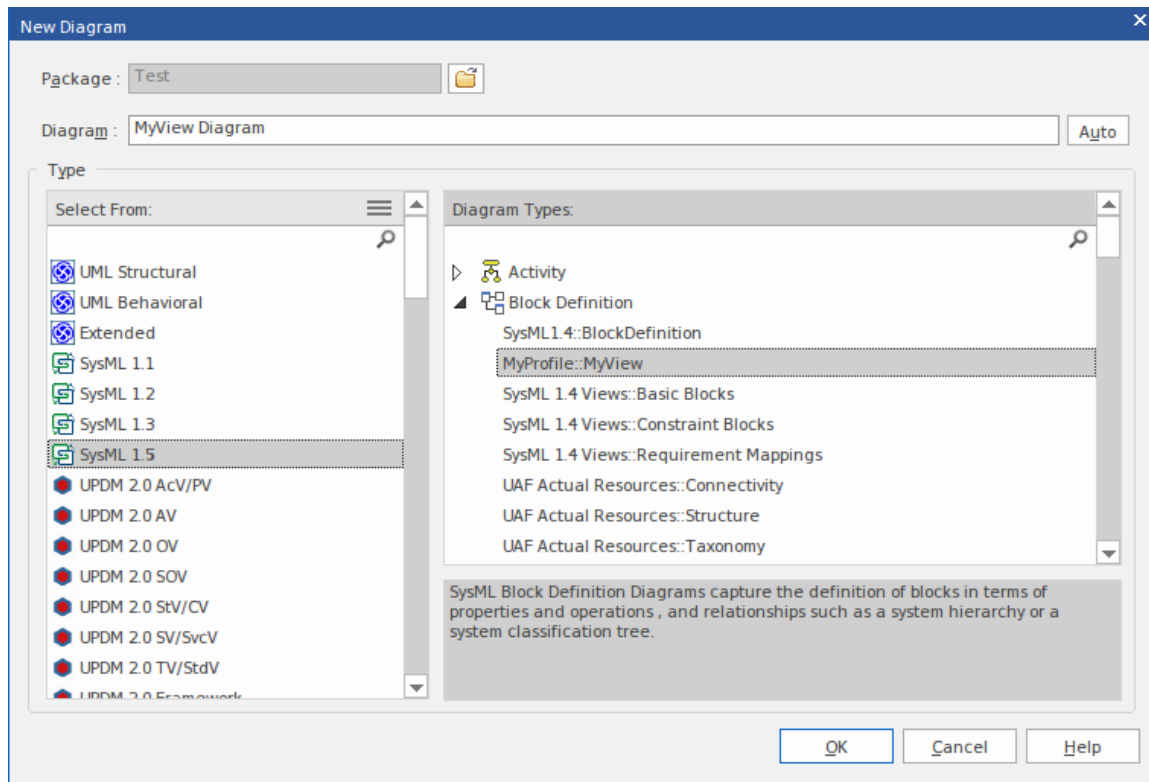
	 <p>Changing the diagram views in the 'Applied Metamodel' option list will change the elements and relationships in the Toolbox.</p>
<p>4</p>	<p>Selecting the 'Filter to Metamodel' option in the Properties window will gray out any elements that are not part of the current diagram View set. This allows you to correct any parts of your model that don't meet the purpose of the selected View, or to filter out elements that might be required to be there, but do not form part of the current modeling goal.</p>



Custom Metamodel Diagram View

Enterprise Architect has a wide range of built-in diagram views, but you can also create your own Metamodels that define custom diagram Views. For example, you might define a specific Metamodel that addresses the needs of Requirements modeling in your organization, and then mandate that all Requirements diagrams use that diagram View instead of the built-in Requirement diagram Views. You can quickly add your diagram Views to the current model, where you or other modelers can apply them to your diagrams.

As an illustration, suppose you decide to make available a new SysML 1.4 Block Definition diagram View in your project, called 'MyView'. Users will access it through the 'New Diagram' dialog, expanding the Block Definition diagram type.

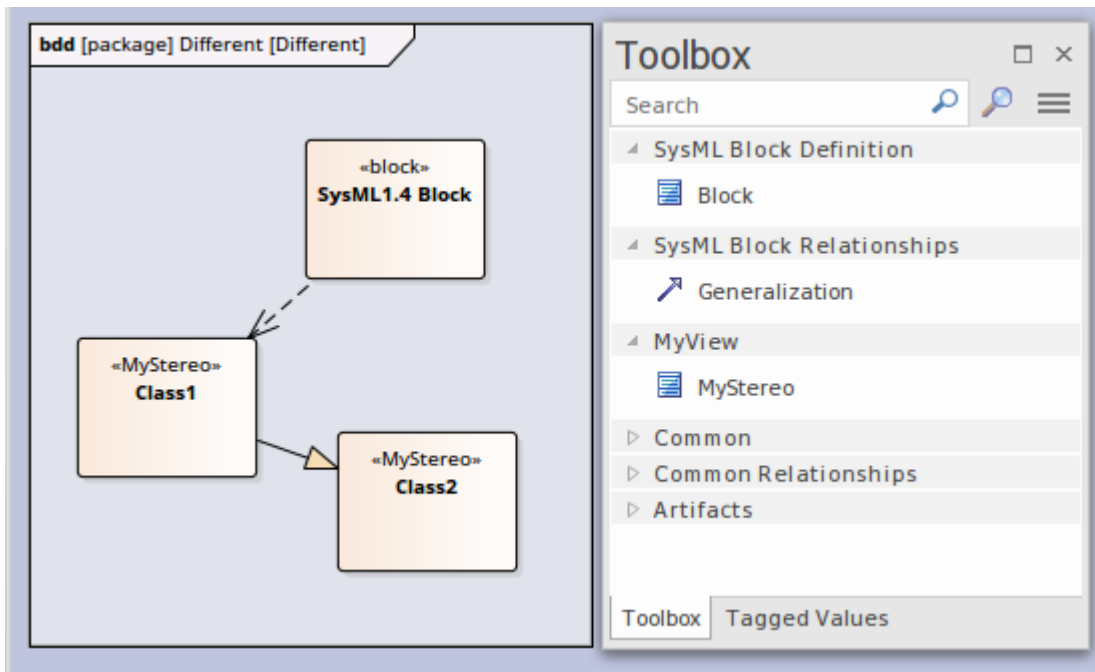


The fully extended name of the diagram View reflects the parent Profile name (MyProfile) and the View name (MyView) - hence 'MyProfile::MyView'. You could call the example View SysML 1.4 Views:: MyView to indicate that it is a member of the SysML 1.4 View suite.

If you are extending a UML base diagram type, with the Profile name 'UML', the equivalent View name could be something such as 'UML::Full Class'.

The users select the example diagram View to create a very simple SysML 1.4 Block diagram that can have:

- Two types of element:
 - a SysML 1.4 Block element (an extended Class from the SysML 1.4 technology)
 - a MyStereo element that you are defining within your new metamodel 'MyView' as a Class with the stereotype MyStereo
- One type of connector - a standard SysML Block Generalization (which is the same as a standard UML Generalization)



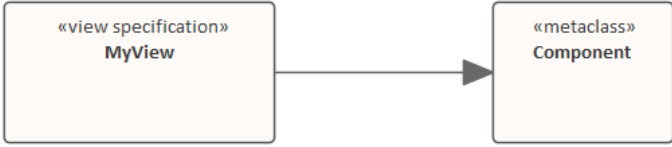

The diagram View makes the elements and connector available from the Toolbox, as shown, and from the Quick Linker. The table *Create Custom Diagram View in a Profile* explains how to create a Metamodel that defines a new diagram View, finishing with the MyView example.

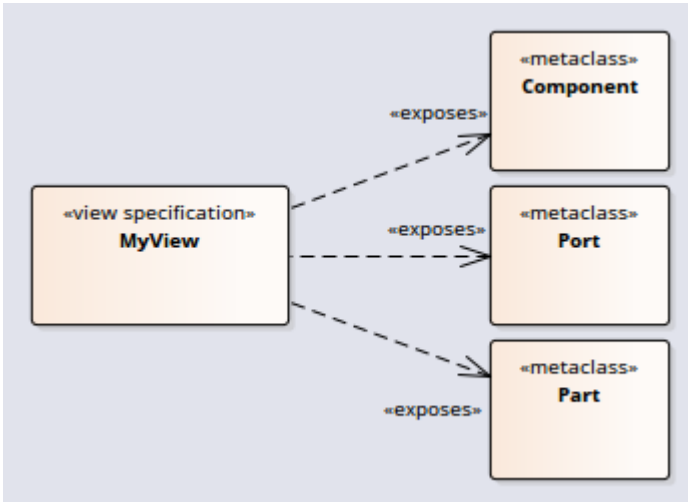
Access

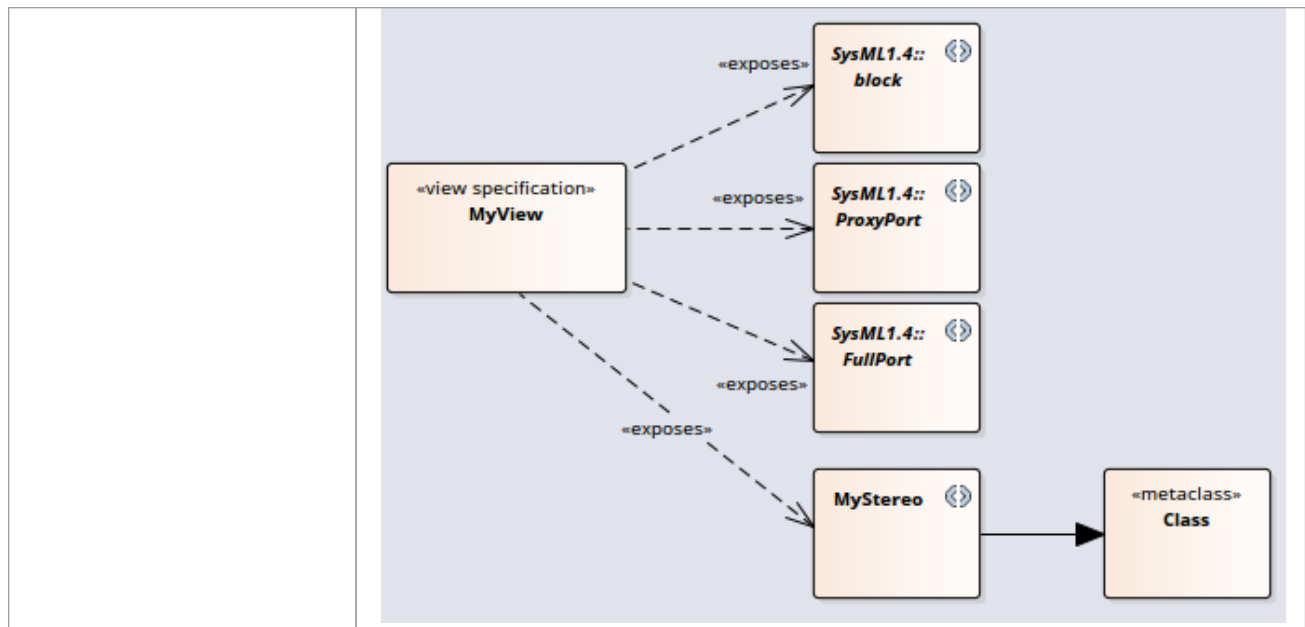
Ribbon	Design > Diagram > Toolbox: > Profile > Metamodel
Keyboard Shortcuts	Ctrl+Shift+3 : > Profile > Metamodel

Create Custom Diagram View in a Profile

Operation	Action
Create the Profile diagram	<p>In your profile Package, create a new Package diagram and, in the Diagram Toolbox, open the 'Profile' page (select the 'Design > Diagram > Toolbox' ribbon option, then click on and select 'Profile').</p> <p>Drag the 'Profile' icon onto the diagram and give it the name 'MyProfile', selecting to add a child Class diagram of the name 'MyView', which you open.</p> <p>Expand the 'Metamodel' page in the Toolbox and note the:</p> <ul style="list-style-type: none"> 'View Specification' element, which you can use to create a custom diagram View 'Exposes' connector, which you use to specify the contents of the Toolbox page associated with the custom diagram View
Add View Specification	<p>Within a Profile, you use the 'View Specification' stereotyped element to identify the new custom diagram View as an extension of an existing built-in or stereotyped</p>

	<p>diagram.</p> <p>Drag the 'View Specification' icon onto the Profile diagram, and give the element a name; in our example, 'MyView'.</p> <p>The first thing to consider when defining a new View, is what diagram type or types it should be available for. The next two rows show how to define a View for a UML diagram and a Profile diagram.</p> <p>In both cases, click on the 'Extension' icon and drag from the View Specification to the diagram-type element, to create the Extension connector.</p>
<p>Extending a UML Diagram Type</p>	<p>To extend a base UML diagram type, drag the 'Class' icon from the Toolbox onto the diagram and, on the Properties window, give the element:</p> <ul style="list-style-type: none"> • The exact name of the diagram type (as listed in the <i>Built-in Diagram Types</i> Help topic) such as 'Logical' (for a Class diagram), and • The stereotype <<metaclass>> <p>This example shows 'MyView' as previously created, extending the UML Component diagram.</p>  <pre> classDiagram class MyView["«view specification» MyView"] class Component["«metaclass» Component"] MyView -- > Component </pre> <p>The result is that in the 'New Diagram' dialog, an extra View is added under the UML Component Diagram type.</p>
<p>Extending a Profiled Diagram Type</p>	<p>To extend a profiled diagram type, such as a BPMN or SysML diagram type, drag the 'Stereotype' icon onto the diagram and give the Stereotype element the exact fully qualified name of the diagram type.</p> <p>Because this is a reference to an external stereotype, it should also be marked as Abstract to prevent it being exported into the profile. To do that, display the Properties window, expand the 'Advanced' section and select the 'Abstract' checkbox.</p> <p>This example shows 'MyView' as previously created, extending the GRA-UML Component Diagram type.</p>  <pre> classDiagram class MyView["«view specification» MyView"] class Component["GRA-UML: «metaclass» GRA Component {abstract}"] MyView -- > Component </pre> <p>The result is that the 'New Diagram' dialog will show the View we are defining under the GRA-UML component diagram.</p> <p>Note: If you do not know the fully qualified name of the diagram type you are extending, query the API to get the 'Metatype' field. In a JavaScript console you can use:</p> <pre>?GetCurrentDiagram().MetaType</pre> <p>Alternatively, select the diagram in the Browser then look in the docked Properties window where it will be listed under MDG Technology.</p>
<p>Exposing Objects in the Diagram View Toolbox</p>	<p>An Exposes connector adds an object to the Toolbox page for the diagram View. For each element and connector to add to the diagram View's Toolbox page, you drag a 'definition element' onto the diagram and then click on the 'Exposes' icon in the Toolbox 'Profile' page and drag the cursor from the View Specification element to the 'definition element' to create the connector.</p>

	<p>The type of definition element depends on whether you are exposing a base UML element or a stereotyped element, as shown in the next two rows.</p>
<p>Exposing UML Element Types</p>	<p>If you are using base UML element or connectors in your custom diagram View, then for each element or connector:</p> <ol style="list-style-type: none"> 1. Drag the 'Metaclass' icon from the Toolbox 'Profile' page onto the diagram and give it the name of the base element or connector type it represents and 2. Add the Exposes connector between the View Specification element and the Metaclass element <p>For example:</p>  <pre> graph LR MyView["«view specification» MyView"] -.-> «exposes» Component["«metaclass» Component"] MyView -.-> «exposes» Port1["«metaclass» Port"] MyView -.-> «exposes» Port2["«metaclass» Part"] </pre>
<p>Exposing Profiled Element Types</p>	<p>If you are defining a new stereotyped object in the diagram view, or using stereotyped elements already defined in other profiles, then for each element or connector:</p> <ol style="list-style-type: none"> 1. Drag the 'Stereotype' icon from the Toolbox 'Profile' page onto the diagram, and give the element the name of the stereotyped element or connector it represents 2. If the Stereotype is defined in another profile, expand the 'Advanced' section of the Properties window and select the 'Abstract' checkbox 3. If the Stereotype is being defined here, add to the diagram the base element that the Stereotype extends, and create an Extension connector between the Stereotype and base element 4. Add the Exposes connector between the View Specification element and the Stereotype element <p>For example:</p>



Completing the Example

With reference to the earlier rows in the table, on the MyView Class diagram (the child of the MyProfile diagram):

1. Create the View Specification element MyView.
2. Create the Stereotype element SysML1.4::Block Definition and set it to Abstract.
3. Connect the View Specification to the SysML1.4::Block Definition with an Extension connector.
4. Create a Metaclass element called Generalization.
5. Create a Stereotype element called SysML1.4::Block and set it to Abstract.
6. Create a Stereotype element called MyStereo and a Metaclass element called UML Class and connect the Stereotype to the Metaclass with an Extension connector.
7. Connect the View Specification element to the Generalization element, the SysML1.4::Block element and the MyStereo element, each with an Exposes connector.

This illustration represents the diagram that you have created:

The diagram illustrates the following relationships:

- «stereotype» SysML1.4::Block Definition** (top left) is extended by **«view specification» MyView** (middle left) via a solid arrow labeled "Extension".
- «metaclass» Class** (top right) is extended by **«stereotype» MyStereo** (middle right) via a solid arrow labeled "Extension".
- «view specification» MyView** (middle left) exposes **«stereotype» MyStereo** (middle right) via a dashed arrow labeled "«exposes»".
- «view specification» MyView** (middle left) exposes **«metaclass» Generalization** (bottom left) via a dashed arrow labeled "«exposes»".
- «view specification» MyView** (middle left) exposes **«stereotype» SysML1.4::Block** (bottom right) via a dashed arrow labeled "«exposes»".

As you complete your diagram view, you might decide that elements of one type should be connected to elements of the same type or of other types by using specific kinds of connector. You would define this using Meta-Relationship connectors, as discussed in the *Define Metamodel Constraints* Help topic.

Save the View Specification diagram. You can now add it to an MDG Technology file as part of its parent Profile; you add the parent Profile to the 'MDG Technology Wizard - Profile files selection' page. See the *Add a Profile* Help topic.

Define Metamodel Constraints

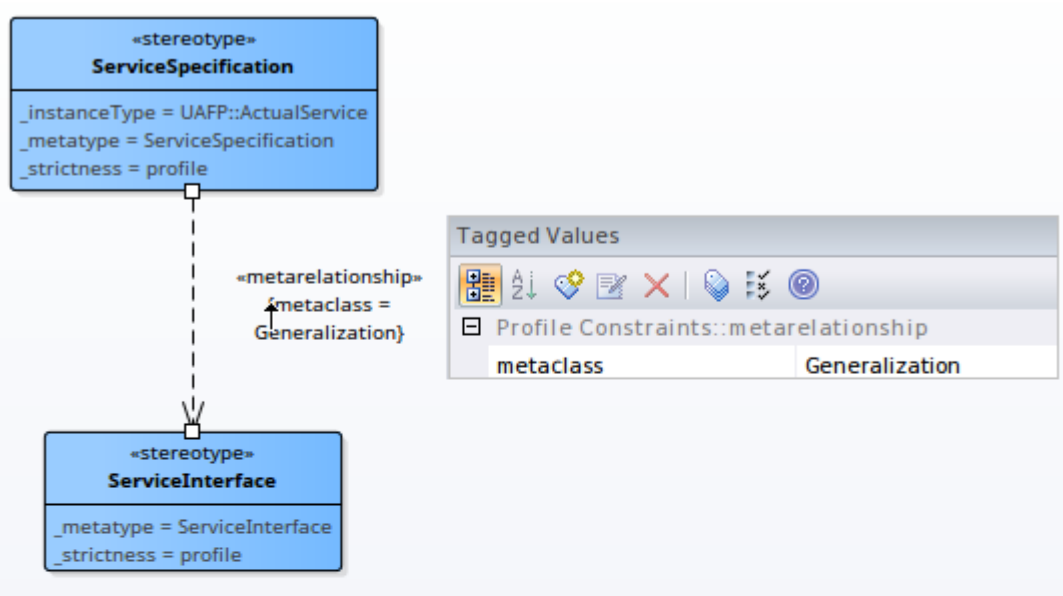
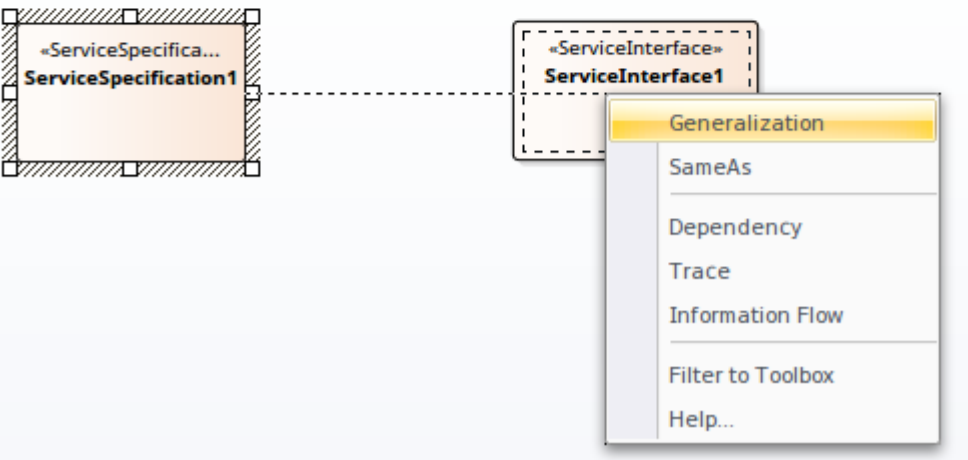
When extending UML to develop a domain-specific Profile, Enterprise Architect allows you to specify constraints to restrict the connectors that can be drawn from a Stereotype, either using the Quick Linker or from the Toolbox. These constraints are defined using the relationships under the 'Metamodel' page of the 'Profile' toolbox.

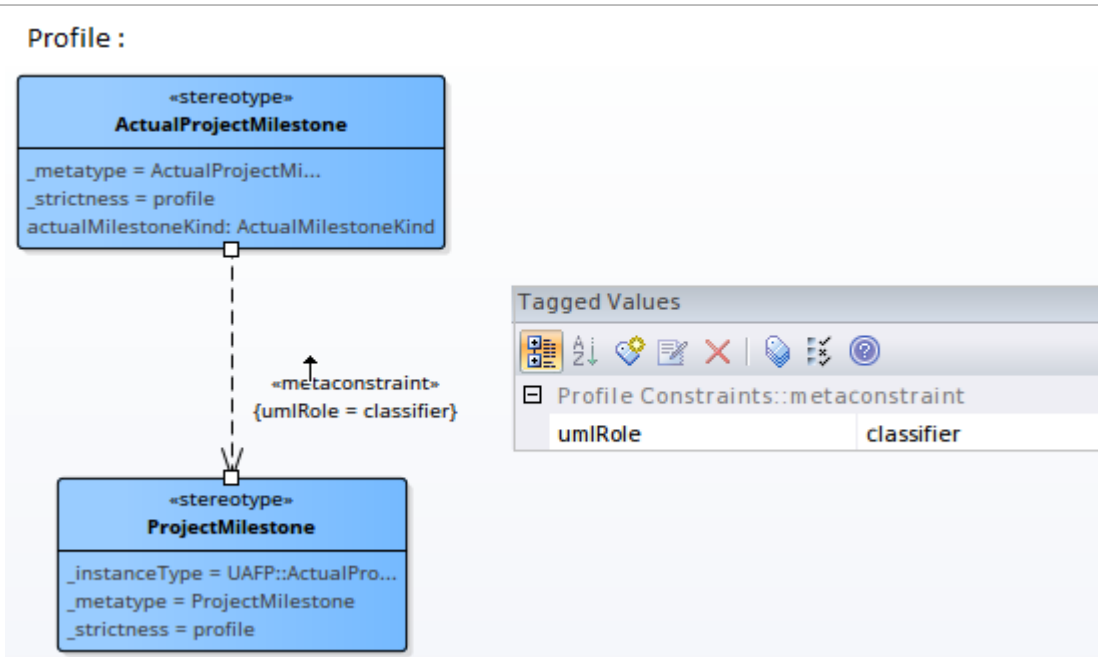
Access

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

Add Metamodel Constraints to a Profile

Item	Detail
Meta-Relationship	<p>A «metarerelationship» connector between two Stereotypes is used to specify a valid UML Connector between these two Stereotypes.</p> <p>The name of the UML Connector should be set in the tag 'metaclass' on the «metarerelationship» connector.</p>

	<p>Profile :</p>  <p>Quick Linker in Model :</p>  <p>In the Profile example, a «metarerelationship» connector is drawn from ServiceSpecification to ServiceInterface and the name of the UML Connector is specified in the 'Tags' tab of the Properties window for the connector.</p> <p>After importing this Profile into a model, Enterprise Architect will show the UML Connector when the Quick Linker is used to draw a relationship between a ServiceSpecification and ServiceInterface.</p>
<p>Meta-Constraint</p>	<p>A «metacconstraint» connector between two Stereotypes is used to specify a constraint between these two Stereotypes.</p> <p>The constraint should be set in the tag 'umlRole' on the Meta-Constraint connector.</p>



In the Profile example, a «metaconstraint» connector is drawn from ActualProjectMilestone to ProjectMilestone and the constraint is specified as classifier on the tag 'umlRole' in the connector's Tagged Values.

After importing this Profile into a model, Enterprise Architect will show only the ProjectMilestone stereotyped elements when assigning a classifier for ActualProjectMilestone element.

Constraint values for the tag 'umlRole' include:

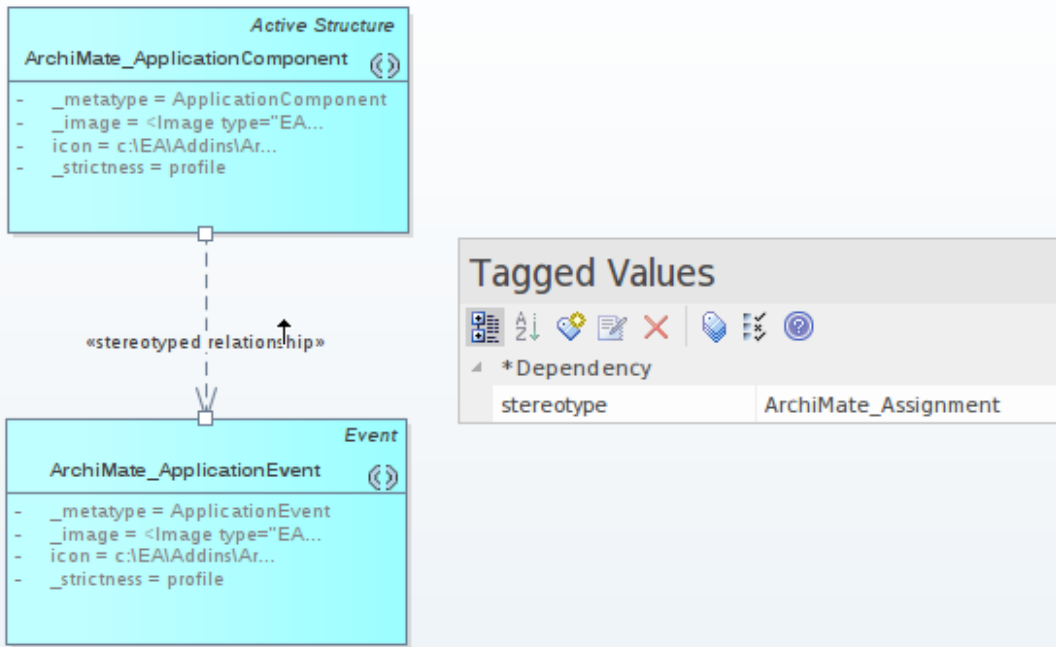
- classifier – restricts the classifier for the source Stereotype element to the target Stereotype element
- type – restricts the type for the source Stereotype element to the target Stereotype element
- behavior - restricts the behavior for the source Stereotype element to the target Stereotype element
- conveyed - restricts the conveyed element for the source Stereotype element to the target Stereotype element
- slot - restricts the slot for the source Stereotype element to the target Stereotype element
- client/source/end[0].role/informationSource – restricts the source of a connector to the target Stereotype element
- supplier/target/end[1].role/informationTarget - restricts the target of a connector to the target Stereotype element
- realizingConnector/realizingActivityEdge/realizingMessage - restricts the relationship that can realize an information flow
- typedElement/instanceSpecification – when dropping as classifier from the Browser window, this constraint restricts the type to the target Stereotype element
- owner/class/activity/owningInstance – restricts the container of this element to the target Stereotype element; this constraint is used to create embedded element rules for the Quick Linker and validate nesting during Model Validation
- ownedElement/ownedAttribute/ownedOperation/ownedParameter/ownedPort – restricts the element/attribute/operation/parameter/port that can be owned by the source Stereotype element; this constraint is typically used to validate nesting during Model Validation
- annotatedElement/constrainedElement – restricts the target of a Note Link connector to the target Stereotype element

<p>Stereotyped Relationships</p>	<p>You can use a «stereotyped relationship» connector between two Stereotypes or Metaclasses to specify a valid stereotyped connector between <i>instances</i> of those elements.</p> <p>When specifying the relationship, if the relationship being referenced is defined in the profile in which the rule is defined, the stereotype property can be set to only the name of that stereotype. However, if the</p>
----------------------------------	---

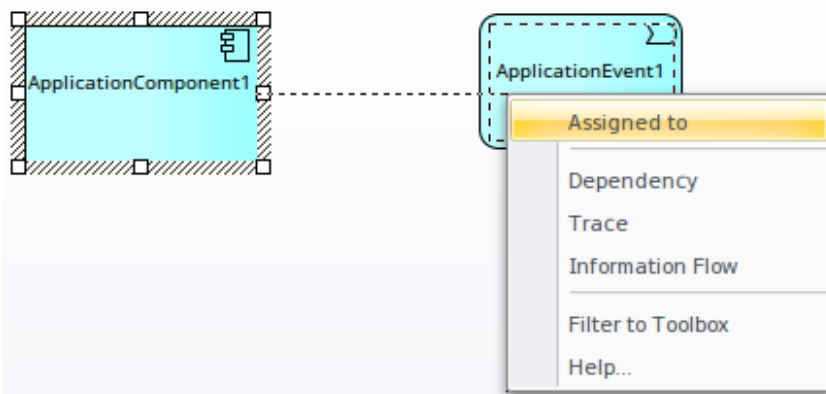
ionship

relationship is defined in another profile you must use a fully qualified stereotype name corresponding to where the stereotype is defined.

Profile :



Quick Linker in Model :



In the Profile example, a «stereotyped relationship» connector is drawn from ApplicationComponent to ApplicationEvent and the stereotype of the relationship is set to 'Assignment' in the connector's Tagged Values.

After importing this Profile into a model, Enterprise Architect will show the 'Assigned' option when the Quick Linker is used to draw a relationship between an ApplicationComponent and ApplicationEvent.

Special Metaclasses

You can specify the source of a connector to be a superclass of all specialized forms, and the target to a special metaclass that specifies a relationship to the actual metaclass when it is used. You use one of these terms as the element name for a Class element with the stereotype «metaclass».



Item	Detail
source.m etatype	The target element must match the exact stereotype defined at the source.
source.m etatype general	The target element can match the exact stereotype used at the source, and any concrete (isAbstract=false) generalized stereotypes.
source.m etatype specific	The target element can match the exact stereotype used at the source, and any concrete (isAbstract=false) specialized stereotypes.
source.m etatype both	The target element can match the exact stereotype used at the source, and any concrete (isAbstract=false) generalized or specialized stereotypes.
<profile_ name >::*	Replace '<profile_name>' with the name of a profile; this will expand to a list of all the concrete stereotypes in the given profile.
<none>	Use this metaclass name when you want to prevent the source element from inheriting the specified connector from its supertypes.

Constraints on Meta-Constraint Connector

When creating a domain-specific Profile, Enterprise Architect allows you to specify constraints between related Stereotypes. As an example, you can restrict the element that can be set as a classifier on a Stereotyped element.

A Meta-Constraint connector, on the 'Metamodel' page of the 'Profile' toolbox, between two Stereotypes is used to specify the constraint between the two Stereotypes. The constraint should be set in the tag 'umlRole' on the Meta-Constraint connector.

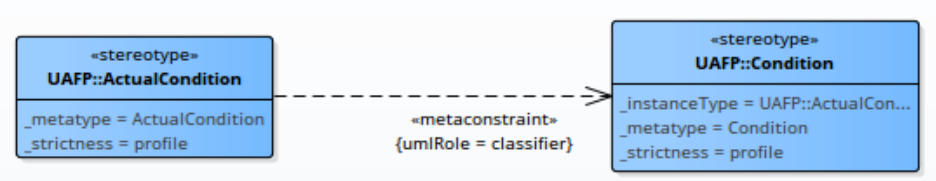
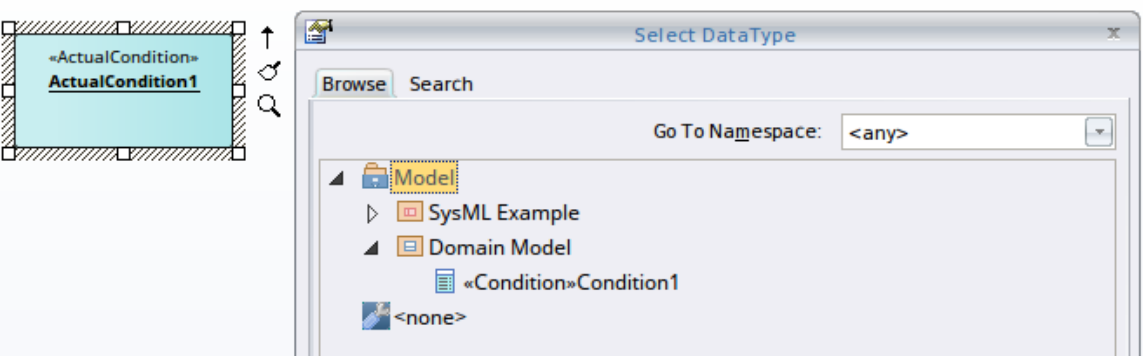
Access

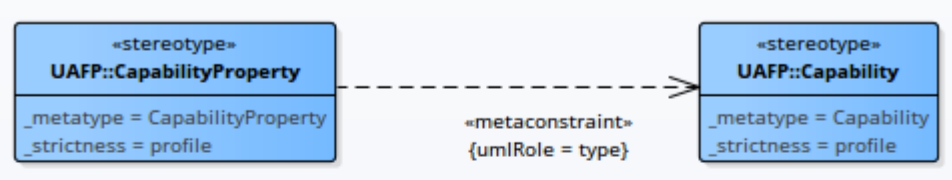
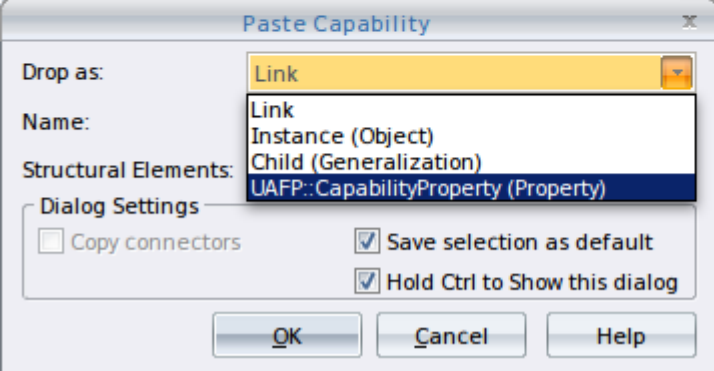
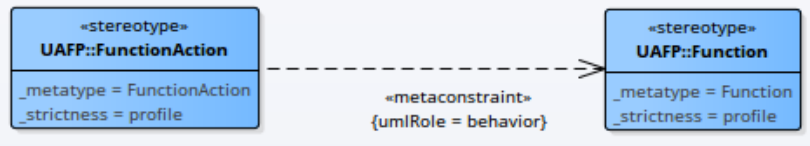
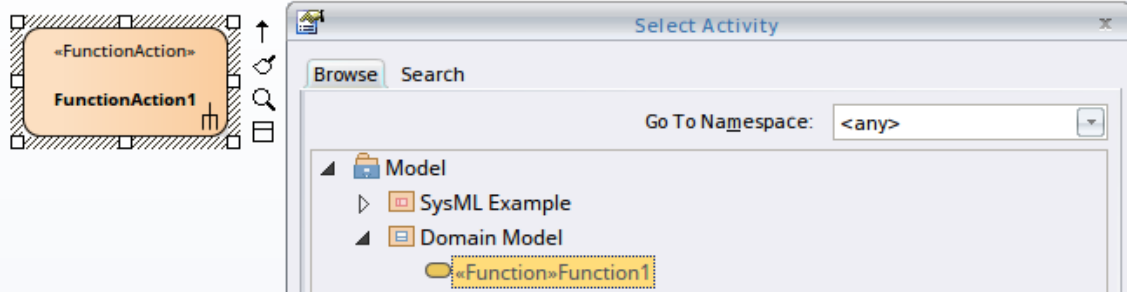
Ribbon	Design > Diagram > Toolbox :  > Profile > Metamodel
Keyboard Shortcuts	Ctrl+Shift+3 :  > Profile > Metamodel

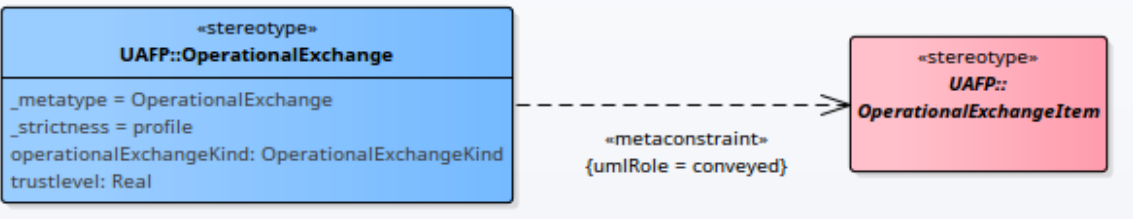
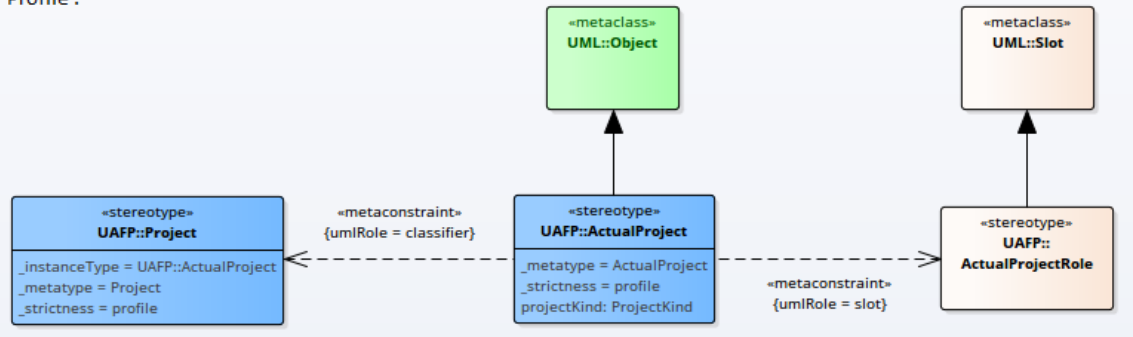
Constraint values for tag 'umlRole'

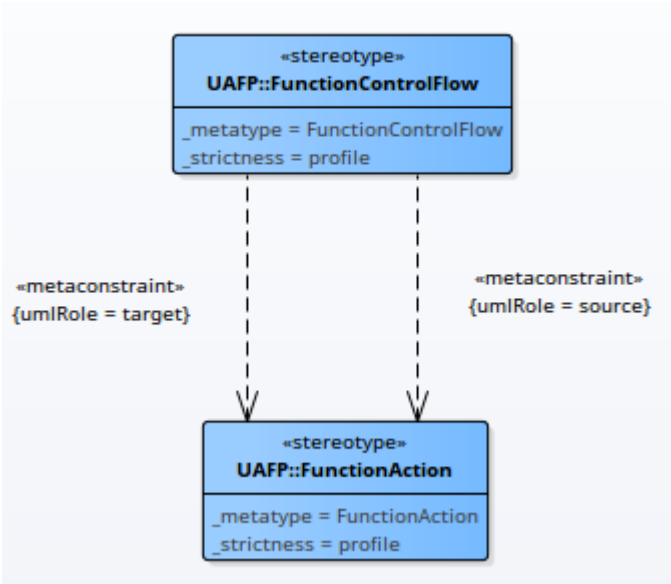
(NOTE: The table below presents all of the acceptable constraint values for the tag 'umlRole'. The values are case-sensitive and should be entered just as they are shown in the table.)

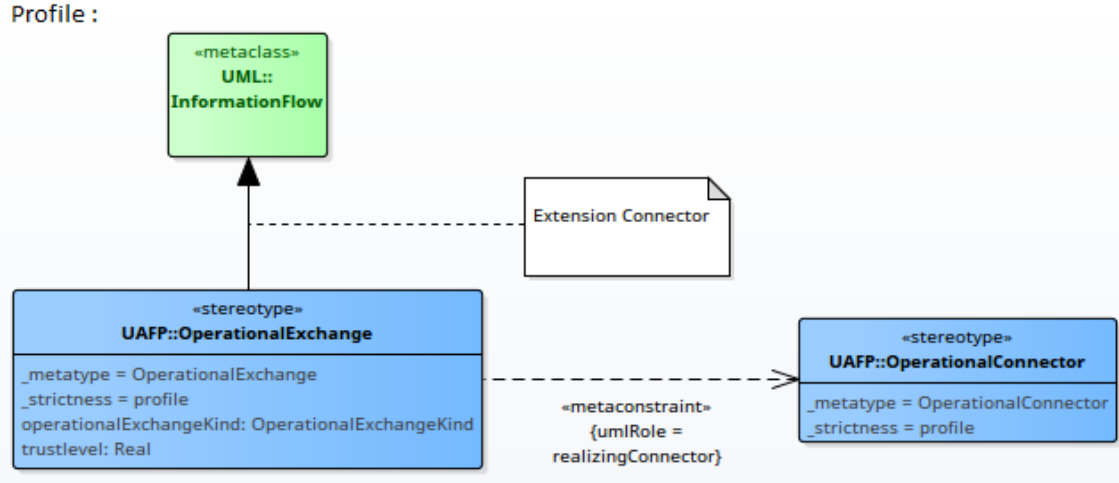
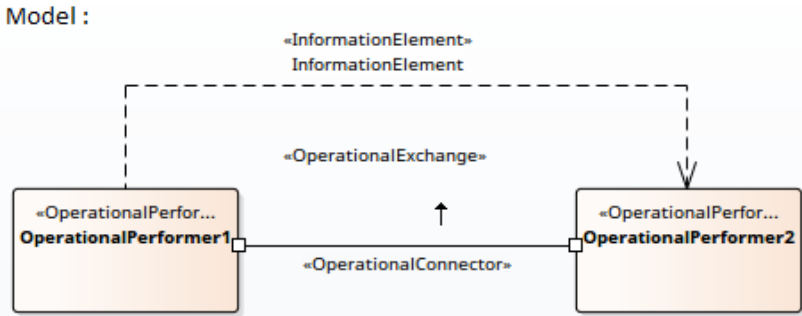
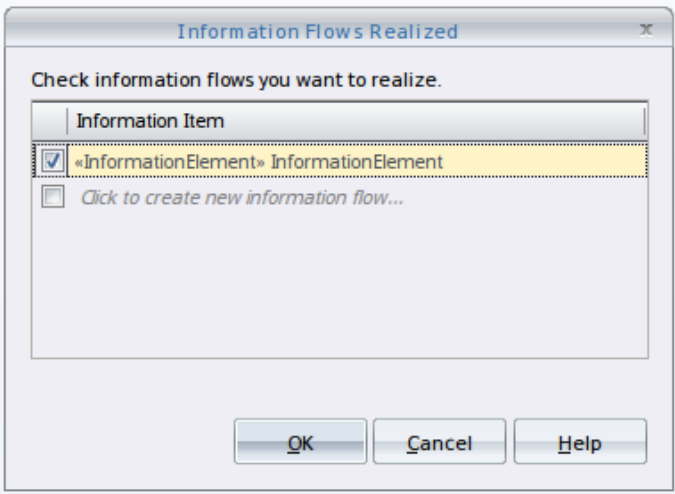
Constraint values for the tag 'umlRole' on the Meta-Constraint connector are:

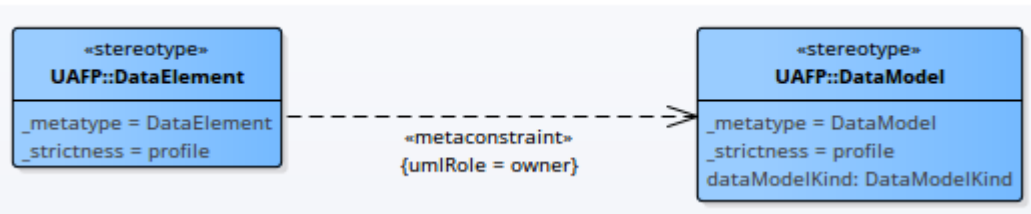
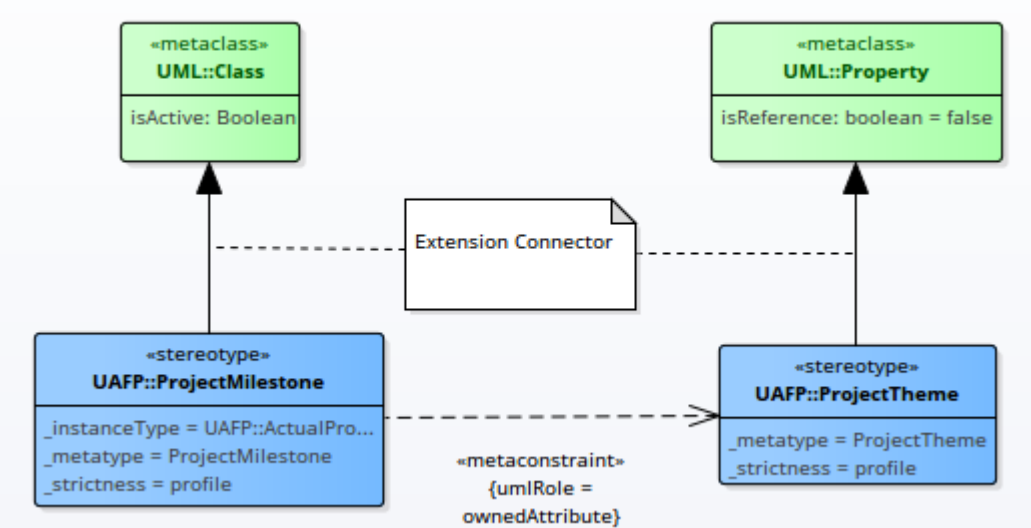
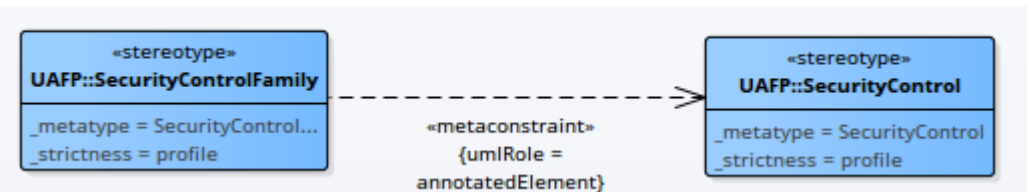
Constraint	Description
classifier	<p>Set this constraint to restrict the classifier for the source Stereotype element as the target Stereotype element.</p> <p>Profile :</p>  <p>Model :</p>  <p>In the Profile example, a Meta-Constraint connector is drawn from the stereotype ActualCondition to Condition and the constraint is specified as 'classifier' on the tag 'umlRole' in the connector's list of Tagged Values. This means that only a 'Condition' stereotyped element can be set as the classifier for an</p>

	<p>ActualCondition stereotyped element.</p> <p>After importing this Profile into a model, Enterprise Architect will show only Condition stereotyped elements in the 'Select DataType' dialog when setting the DataType for an ActualCondition stereotyped element.</p>
<p>type</p>	<p>Set this constraint to specify the type for the target Stereotype element when it is dropped from the Browser window into a diagram while pressing and holding the Ctrl key.</p> <p>Profile :</p>  <p>Model :</p>  <p>In the Profile example, a Meta-Constraint connector is drawn from the stereotype CapabilityProperty to Capability and the constraint is specified as 'type' on the tag 'umlRole' in the 'Tags' tab of the connector's Properties window.</p> <p>After importing this Profile into a model, when a Capability stereotyped element is dropped from the Browser window into a diagram while pressing and holding the Ctrl key, the 'Paste <item>' dialog will display CapabilityProperty as one of the options in the 'Drop as' list.</p>
<p>behavior</p>	<p>Set this constraint to restrict the behavior for the source Stereotype element to the same as the target Stereotype element.</p> <p>Profile :</p>  <p>Model :</p>  <p>In the Profile example, a Meta-Constraint connector is drawn from stereotype FunctionAction to Function and the constraint is specified as 'behavior' on the tag 'umlRole' in the 'Tags' tab of the Properties window</p>

	<p>for the connector. This means that only a 'Function' stereotyped element can be set as classifier for a FunctionAction stereotyped element.</p> <p>After importing this Profile into a model, Enterprise Architect will show only Function stereotyped elements in the 'Select Activity' dialog when setting the behavior for a FunctionAction stereotyped element.</p>
<p>conveyed</p>	<p>Set this constraint to restrict the Information Items that can be conveyed on a Stereotype that extends the Information Flow connector.</p> <p>Profile :</p>  <p>In the Profile example, a Meta-Constraint connector is drawn from stereotype OperationalExchange to OperationalExchangeItem and the constraint is specified as 'conveyed' on the tag 'umlRole' in the 'Tags' tab of the Properties window for the connector. This means that when an OperationalExchange connector is drawn, the Information Items that can be conveyed on the connector are restricted to OperationalExchangeItem stereotyped elements.</p>
<p>slot</p>	<p>Set this constraint to restrict the slot for the Stereotype element as the target Stereotype element.</p> <p>Profile :</p>  <p>In the Profile example, a Meta-Constraint connector is drawn from the stereotype ActualProject to ActualProjectRole and the constraint is specified as 'slot' on the tag 'umlRole' in the connector's Tagged Values. Note that the stereotype 'ActualProject' extends UML Object and can classify stereotype 'Project'. When an instance specification for the Project element is created (by dropping it from the Browser window into a diagram while pressing and holding the Ctrl key) in the model:</p> <ul style="list-style-type: none"> • The created instance specification will be stereotyped ActualProject • Any Property in the 'Project' stereotyped element will be created as an 'ActualProjectRole' stereotyped Property in the instance specification
<p>client/ source/ end[0].role/ informationSource</p>	<p>Set this Model Validation constraint to restrict the start element of a Stereotyped connector.</p>

	<p>Profile :</p>  <p>In the Profile example, a Meta-Constraint connector is drawn from stereotype 'FunctionControlFlow' to 'FunctionAction' and the constraint is specified as 'source' on the tag 'umlRole' in the connector's Tagged Values. This means that when a FunctionControlFlow connector is drawn, the source element should be a FunctionAction stereotyped element. Otherwise, Enterprise Architect will flag an error when performing a Model Validation.</p>
<p>supplier/ target/ end[1],role/ informationTarget</p>	<p>Set this model validation constraint to restrict the target element of a Stereotyped connector.</p>
<p>realizing Connector/ realizing Activity Edge / realizing Message</p>	<p>Set this constraint to restrict the relationship that can realize an Information Flow connector.</p>

	<p>Profile :</p>  <p>Model :</p>   <p>In the Profile example, a Meta-Constraint connector is drawn from stereotype OperationalExchange (which extends a UML InformationFlow metaclass) to OperationalConnector and the constraint is specified as 'realizingConnector' on the tag 'umlRole' in the connector's Tagged Values. This means that when an OperationalConnector connector is drawn, the Information Flow connector that can be realized on this connector can be an OperationalExchange stereotyped connector.</p>
<p>type dEle ment / insta nceS pecif icati on</p>	<p>When dropping as classifier from the Browser window, this constraint restricts the available type to the target Stereotype element.</p>

<p>owner/ class/ / activity/ owningInstance</p>	<p>Set this constraint to restrict the container/owner of the element to the target Stereotype element. This constraint is used to create embedded element rules for the Quick Linker and to validate nesting during Model Validation.</p> <p>Profile :</p>  <p>In the Profile example, a Meta-Constraint connector is drawn from the stereotype DataElement to DataModel and the constraint is specified as 'owner' on the tag 'umlRole' in the connector's Tagged Values. This means that DataElement stereotyped elements can be children of DataModel stereotyped element. In other words, only DataModel can contain/own DataElements in the Model.</p>
<p>ownedElement/ ownedAttribute/ ownedOperation/ ownedParameter/ ownedPort</p>	<p>Set this constraint to restrict the element/attribute/operation/parameter/port that can be owned by the source Stereotype element. This constraint is typically used to validate nesting during Model Validation.</p> <p>Profile :</p>  <p>In the Profile example, a Meta-Constraint connector is drawn from stereotype ProjectMilestone to ProjectTheme and the constraint is specified as 'ownedAttribute' on the tag 'umlRole' in the connector's Tagged Values. This means that ProjectMilestone stereotyped elements can contain 'ProjectTheme' stereotyped attributes in the model.</p>
<p>annotatedElement/ constraintElement</p>	<p>Set this model validation constraint to restrict the target of a NoteLink connector.</p> <p>Profile :</p>  <p>In the Profile example, a Meta-Constraint connector is drawn from stereotype SecurityControlFamily to SecurityControl and the constraint is specified as 'annotatedElement' on the tag 'umlRole' in the connector's Tagged Values.</p>

	<p>When the Profile is imported into a model, the target of a NoteLink connector from a SecurityControlFamily stereotyped element should be a SecurityControl stereotyped element. Otherwise, Enterprise Architect will flag an error when performing a Model Validation.</p>
--	---

Metamodel Constraints and the Quick Linker

When you drag the Quick Linker arrow to create a relationship to another element, a menu of available connector types and - if no target element is selected on the diagram - a menu of available element types display. The tables in this topic show where the names of the connector and element types are drawn from when you have - or have not - provided values for the Metamodel constraint properties.

Rule Filtering

The metamodel constraints primarily define what connections are valid. The Quick Linker is built from these valid relationships and is then filtered in a number of ways in order to present the relevant relationships to the user.

Item	Detail
Toolbox Filtering	<p>By default for all new diagrams the elements and relationships offered by the Quick Linker are restricted to match the types available in the toolbox.</p> <p>This can be changed by the user on a diagram by selecting the Complete view for the diagram or unchecking the 'Filter to Toolbox' option within the Quick Linker menu.</p>
Common Relationships	<p>Relationships defined with the <code>_IsCommon</code> property will not be offered as suggestions when a new element also needs to be created.</p> <p>These UML relationships include this behavior when they are used with a metarelationship:</p> <ul style="list-style-type: none"> • Abstraction • Dependency • InformationFlow • Realization • Usage

Connector Labels

This table identifies the points from which the Quick Linker can retrieve names to display in the menu for the available connector types.

Item	Detail
Meaning Forwards and Meaning Backwards	<p>Stereotypes with values defined in the <code>_MeaningForwards</code> and <code>_MeaningBackwards</code> properties will use those values to describe the connector in the Quick Linker menu.</p> <p>Note: If <code>_MeaningBackwards</code> is not defined for a stereotype, the Quick Linker will offer an option to create the relationship in the backwards or reverse direction.</p>
Metatype Name	<p>Stereotypes with values defined in the <code>_Metatype</code> properties will use those values to describe the connector in the Quick Linker menu when no 'name' properties are defined.</p>

e	
Stereotype Name	If no <code>_MeaningForwards</code> , <code>_MeaningBackwards</code> or <code>_Metatype</code> values are defined, the stereotype name will be used as the menu label for a relationship.
Metaclass Name	When using a Metarelationship connector to include UML relationships between your stereotypes, you do not have control of the labels used for the relationship. The Quick Linker will use the same labels as are used when those relationships are available between UML elements.

Element Labels

When you have dragged the Quick Linker to empty space, a menu displays the types of target element available. This table identifies where the Quick Linker retrieves names from to display in the menu of available elements.

Item	Detail
Metatype Name	Stereotypes with values defined in the <code>_metatype</code> properties will use those values to describe the element in the Quick Linker menu.
Stereotype Name	If no <code>_MeaningForwards</code> , <code>_MeaningBackwards</code> or <code>metatype</code> values are defined, the name of the stereotype will be used as the menu label for an element.
Metaclass Name	When using a Metarelationship connector or Stereotypedrelationship connector to link your stereotypes to UML elements, you do not have control of the labels used for the element. The Quick Linker will use the same labels as are used when those elements are connected under UML.

