

UML® Validation Extension

UML® Validation Extension	1
Disclaimer	3
Dependencies	3
Limitations of the trial version.	3
Supported rule sets	4
Example of a rule specification in the Activity file:	4
Activity diagram rules	5
Class diagram rules	7
Component diagram rules	9
Connector (non-UML diagram specific) rules	12
General (non-UML diagram specific) rules	13
Requirement rules	14
Sequence diagram rules	15
State Machine diagram rules	16
Use Case diagram rules	17
Installation	20
Verifying the installation	22
Licensing	25
Trial version	25
Licensed version	25
Request the license activation	25
Receive the license activation	26
Running the validation	28
Package level validation	28
Diagram level validation	29
Element level validation	29
User Interface	30
Customizing the Rule Set	31
Verifying the result set	32
Troubleshooting	34



UML® Validation User Guide

Support and contact information.....	35
--------------------------------------	----

Disclaimer

Version 4.5.x of the *UML Validation Extension*:

- Has been successfully tested for deployment with Enterprise Architect version 15.x, and 16.x (both 32- and 64-bit versions). There is no guarantee that versions prior to EA 15.x will work properly. No effort will be made to support earlier releases of Enterprise Architect.
- Is not compatible with previous versions of the add-in. The license key remains valid, but the old product needs to be uninstalled before upgrading.

The add-in, as well as these guidelines, may or may not be applicable to any later version of the tool as released by the vendor, Sparx Systems. If required, updates to this software will be made available to support future versions of Enterprise Architect.

Great care has been taken during development to use SQL statements that are supported across the common backend database platforms:

- SQLite (QEA and QEAX Sparx Systems file based repositories).
- Firebird
- MySQL 8.0
- SQL Server 2019
- Oracle (Express Edition 21c).

Note that Microsoft Access databases are NOT supported.

Should a statement fail to execute correctly, please refer to the [Troubleshooting](#) part of this User Guide for assistance.

If any other problems are encountered, either during installation or operation of this software, please [contact us](#) through any of the channels listed at the bottom of this document.

Dependencies

The add-in depends on the following components being installed on the system:

- Interop.EA.dll (part of the standard Sparx installation files).
- Microsoft .Net Framework 4.8.

Limitations of the trial version.

The following limitations apply to the trial version:

- The software activation is granted for five (5) consecutive days only.
- Only the first five (5) rule matches for any rule are reported back to the User.
- Only the first twenty (20) rule violations are reported back to the User per Enterprise Architect session.

Supported rule sets

A global property for the entire rule configuration file can be set to exclude all Packages with a particular Status value (see screenshot below). This allows filtering out sandbox or other non-production models from the result set.

Note that this filter option only applies to validations performed at the [Package level](#) (sub-Packages are automatically included unless explicitly filtered out).

The rules are split into separate files relating to specific model constructs being validated: [Activities](#), [Classes](#), [Components](#), [Connectors](#), [Requirements](#), [Interactions \(Sequence\)](#), [State Machines](#) and [Use Cases](#). In addition, a [General](#) rule set validates aspects of the model not specific to UML.

A special "ALL_RULES" file runs all of the rule files in sequence.

Example of a rule specification in the Activity file:

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<!DOCTYPE Validations >
<Validations Version="4" RunSilent="False" VerboseLevel="2" IgnorePackageWithStatus="">
  <ACT Description="Activity Diagram Validations">
    <Elements>
      <ObjectsWithNoClassifier Enabled="True" ID="PartitionWithNoClassifier" Description = "Verify Activity Partitions have a Classifier defined">
        <ElementType Name="'ActivityPartition'" MatchFilterCriteria="True" />
        <ElementMetatype Name="" MatchFilterCriteria="True" />
      </ObjectsWithNoClassifier>
    </Elements>
  </ACT>
</Validations>
```

Note the option for the entire file to:

- Ignore Packages with a specific status value.
- Run the validation in silent mode (i.e. no prompts).
- Set the verbosity level of the output log.

While the structure of each rule may vary considerably, some of the consistent properties are:

Enabled	Set it to "False" to disable the rule.
ID	Unique identifier of the rule. When creating custom rules, assign them unique values.
Description	Displays in the UML Validator output tab in the System Output window as the rules are executed.
Selection criteria	The "MatchFilterCriteria" option indicates (when TRUE) if the search runs as specified (e.g. select elements where type = 'ActivityPartition') or (when FALSE) if the query is negated (e.g. select elements where type <> 'ActivityPartition').

Activity diagram rules

Rule ID	<i>PartitionWithNoClassifier</i>
Description	Verify that Activity Partitions have a Classifier defined.
Notes	Includes checking if the specified Classifier exists in the current repository.
Rationale	Ensure the Actions and Decisions included in a Partition are properly allocated to a structural element that is capable of performing that behavior.

Rule ID	<i>CallBehaviorActionWithNoClassifier</i>
Description	Verify Call Behavior Actions have a Classifier defined.
Notes	Checks if the Call Behavior Action is actually invoking an Activity, Interaction or State Machine. Includes verifying if the specified Classifier exists in the current repository.
Rationale	Ensure the Action invokes an actual behavior, in case it was created as a placeholder, or the invoked behavior was later deleted in the model.

Rule ID	<i>CallOperationActionWithNoClassifier</i>
Description	Verify Call Operation Actions have a Classifier defined.
Notes	Checks if the Call Operation Action is actually invoking an Operation. Includes verifying if the specified Classifier exists in the current repository.
Rationale	Ensure the Action invokes an actual Operation, in case it was created as a placeholder, or the invoked Operation was later deleted in the model.

Rule ID	<i>SendSignalActionWithNoClassifier</i>
Description	Verify Send Signal Actions have a Signal Classifier defined.
Notes	Checks if the Send Signal Action has a reference to a Signal element. Includes verifying if the specified Classifier exists in the current repository.
Rationale	Ensure the Action sends an actual Signal, in case it was created as a placeholder, or the Signal was later deleted in the model.

Rule ID	<i>AcceptEventWithNoTrigger</i>
Description	Verify Accept Event Actions have a Trigger defined.
Notes	Includes checking if the specified Trigger exists in the current repository.
Rationale	Ensure an actual Trigger for the Action is assigned, in case it was created as a placeholder, or the Trigger was later deleted in the model.

Rule ID	<i>DecisionNodeConnectors</i>
Description	Validate that incoming and outgoing connectors of Decision Nodes are compliant with the UML specification.
Notes	None.
Rationale	Logically, the Input and output connectors of a Decision must be of the same type: either Control Flow or Object Flow (with the exception in the former case of a single Decision Input Flow).

Rule ID	<i>ActionPinConnectors</i>
Description	Validate only Object Flow connects Action Pins together.
Notes	The rule checks for the case where the source and target Action Pins are marked as Control Pins, in which case a Control Flow is allowed.
Rationale	As per the UML specification.

Rule ID	<i>ActionConnectors</i>
Description	Validate only Control Flow connects Actions (as input or output).
Notes	None.
Rationale	As per the UML specification.

Class diagram rules

Rule ID	<i>ClassAssociationEndProperties</i>
Description	Verify for each (navigable) connector end that role name and multiplicity are set.
Notes	Verifies both Association and Aggregation connector types.
Rationale	<p>Since each navigable connector end represents an Attribute of the source Class, that Attribute should be assigned a name, the same way all manually created Attributes are named.</p> <p>Multiplicity can be assumed equal to one when not specified, but assigning a multiplicity removes ambiguity in the model.</p>

Rule ID	<i>DiagramObjectsWithNoConnectors</i>
Description	Report UML Classes that are visually represented on one or more diagrams and yet do not have any relationships with other elements in the model.
Notes	<p>By default, all diagram types are searched, with the exception of Sequence diagrams. One or more specific diagram types can be specified in a custom rule.</p> <p>Connectors to Note elements are always excluded from the query.</p> <p>This rule is not specific to Class elements and can be reused in other contexts.</p>
Rationale	Find Class elements that may be in an unfinished state. Business domain level Classes are expected to have at least one or more relationships defined.

Rule ID	<i>ClassWithNoDescription</i>
Description	Report Class elements with no description/notes.
Notes	<p>The validation does not currently check for a Linked Document associated with the Class.</p> <p>A filter listing one or more specific stereotypes can be specified in a custom rule.</p> <p>This rule is not specific to Class elements and can be reused in other contexts.</p>
Rationale	Business domain level Classes are expected to have at least some documentation defined.

Rule ID	<i>UnrealizedInterfaces</i>
Description	Report Interface elements that are not realized by a UML Class or Component.
Notes	This rule can be customized to look for other realizing targets besides Class and Component, or to look for a connector type other than Realization.
Rationale	Class level Interfaces that are not realized are of no value to the model.

Rule ID	<i>UnresolvedAttributeDataType</i>
Description	Report Attributes with an unresolved data type Classifier.
Notes	<p>The rule verifies Attributes of any parent type (Class, Signal, Component, Activity, Node, etc.).</p> <p>When a Classifier has been assigned to the Attribute, the rule checks if that Classifier still exists in the current repository. It is not intended to report Attributes where no data type has been assigned.</p> <p>Primitive types (e.g., Integer, String, etc.) that are provided in the default dropdowns by Enterprise Architect are NOT considered Classifiers from a UML perspective.</p>
Rationale	Attributes should not be left assigned with a data type Classifier that no longer exists in the model.

Rule ID	<i>UnresolvedOperationArgument</i>
Description	Report Operations with an unresolved argument (including the return type, if any) Classifier.
Notes	<p>The rule verifies Operations of any parent type (Class, Signal, Component, Activity, Node, etc.).</p> <p>When a Classifier has been assigned to an Operation Parameter (including the return type), the rule checks if that Classifier still exists in the current repository. It is not intended to report Parameters where no data type has been assigned.</p> <p>Primitive types (e.g., Integer, String, etc.) that are provided in the default dropdowns by Enterprise Architect are NOT considered Classifiers from a UML perspective.</p>
Rationale	Operation Parameters should not be left assigned with a data type Classifier that no longer exists in the model.

Component diagram rules

Rule ID	<i>DiagramObjectsWithNoConnectorsEx</i>
Description	Report UML Components that are visually represented on one or more diagrams and yet do not have any direct or indirect relationships with other elements in the model.
Notes	<p>By default, all diagram types are searched, with the exception of Sequence diagrams. If desired, one or more specific diagram types can be specified in a custom rule.</p> <p>Connectors to Note elements are always excluded from the query.</p> <p>Indirect relationships searched for include:</p> <ul style="list-style-type: none"> • Component Port to Component Port. • Component Required or Provided Interface to Component Required or Provided Interface. • Component Port with Required or Provided Interface to Component Port with Required or Provided Interface.
Rationale	Find Component elements that may be in an unfinished state. Components are expected to have at least one or more connections with other Components defined.

Rule ID	<i>ComponentWithNoDescription</i>
Description	Report Component elements with no description/notes.
Notes	<p>The validation does not currently check for a Linked Document associated with the Component.</p> <p>A filter listing one or more specific stereotypes can be specified in a custom rule.</p> <p>This rule is not specific to Component elements and can be reused in other contexts.</p>
Rationale	Components are expected to have at least some documentation defined describing their purpose.

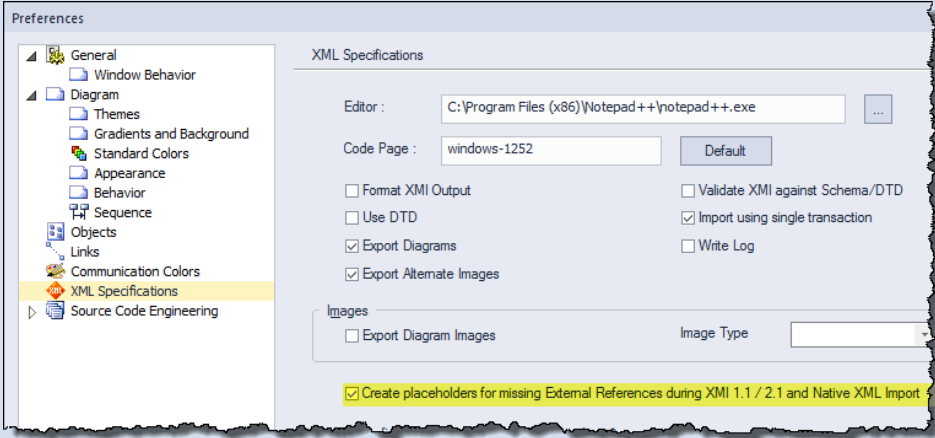
Rule ID	<i>UnrealizedInterfaces</i>
Description	Report Interface elements that are not realized by a UML Class or Component.
Notes	This rule can be customized to look for other realizing targets besides Class and Component, or to look for a connector type other than Realization.
Rationale	Component level Interfaces that are not realized are of no value to the model.

Rule ID	<i>MissingComponentToComponent</i>
Description	Report UML Components that are missing direct or indirect relationships with other Components.
Notes	<p>This rule applies to Components regardless of whether they are represented on diagrams or not.</p> <p>The relationship types searched for are: Association, Connector, Assembly and Delegate. This set can be extended in a custom rule.</p> <p>Indirect relationships searched for include:</p> <ul style="list-style-type: none"> • Component to Port and Port to Component. • Component Port to Component Port. • Component Port with Required or Provided Interface to Component Port with Required or Provided Interface.
Rationale	<p>Find Component elements that may be in an unfinished state with regards to connectivity to other Components.</p> <p>Components are expected to have at least one or more UML compliant relationships with other Components.</p>

Rule ID	<i>MissingComponentToInterface</i>
Description	Report UML Components that are missing direct or indirect connectivity via an Interface.
Notes	<p>This rule applies to Components regardless of whether they are represented on diagrams or not.</p> <p>The Interface connectivity searched for includes:</p> <ul style="list-style-type: none"> • A Realization relationship from the Component to an Interface. • A Required or Provided Interface child element of the Component classified by an Interface element. • A Port of the Component that is classified by an Interface. • A Required or Provided Interface (classified by an Interface) that is a child element of a Port that is itself a child of the Component.
Rationale	<p>Find Component elements that may be in an unfinished state with regards to the Interface/s being implemented.</p> <p>Components are expected to realize at least one Interface.</p>

Connector (non-UML diagram specific) rules

Rule ID	<i>ConnectorNotInDiagram</i>
Description	Report relationships not represented on any diagram
Notes	<p>Filters are available at the following levels:</p> <ul style="list-style-type: none"> • The connector type and stereotype. • The source element type and stereotype. • The target element type and stereotype. <p>None of these values are set in the default rule.</p>
Rationale	In a typical repository, there are many relationships for which a graphical display offers no benefit. It is expected that the User of this validation extension will customize the rule to make it useful for a given context.

Rule ID	<i>UnresolvedConnectorEnds</i>
Description	<p>Locates relationships in which either the source or the target element is missing from the repository. Missing connector end references typically occur during the import (manually or through version control) of incomplete models from other repositories.</p> <p>Caveat: the rule can only detect this condition if the <i>Create Placeholders for Missing External References</i> property is set!</p> 
Notes	None
Rationale	These missing elements may affect the integrity of the model.

General (non-UML diagram specific) rules

Rule ID	<i>EmptyDiagram</i>
Description	Reports diagrams with no elements present.
Notes	None.
Rationale	Diagrams with no contents should be removed from the model.

Rule ID	<i>OrphansWithNoConnectors</i>
Description	Similar to the Enterprise Architect “orphan report” (elements not in any diagram) but also verifies that the element is not referenced in other ways. WARNING: On a large dataset, execution can take a considerable amount of time!
Notes	<p>By default, Action Pins named ‘target’ or ‘result’, automatically created by EA for simulation purposes, are filtered out.</p> <p>Verifies that the element:</p> <ul style="list-style-type: none"> • Has no connectors/relationships. • Is not used as a Classifier. • Is not used as a Trigger specification. • Is not used as a reference Tagged Value.
Rationale	Highlight database elements that may be deprecated, or should not reside in the Package structure selected for validation.

Requirement rules

Rule ID	<i>UnrealizedRequirements</i>
Description	Verify Requirements are realized using a connector type that is UML compliant.
Notes	By default, the rule specifically checks for the Realization relationship targeting the Requirement. The source of the relationship can be any element.
Rationale	Every Requirement should have a relationship with one or more elements to document how it is being met by the UML architecture.

Sequence diagram rules

Rule ID	<i>SequenceMessageNoOperation</i>
Description	Report Sequence messages with no underlying Operation.
Notes	<p>An option is provided to filter out Sequence messages:</p> <ul style="list-style-type: none"> • Targeting one or more element types. • Targeting one or more element stereotypes.
Rationale	Sequence Messages should refer to actual Operations of the Class, Interface, or Port classifying the Lifeline, and not just represent made up text strings.

Rule ID	<i>SequenceMessageMissingOperation</i>
Description	Report Sequence messages with an unresolved Operation reference (e.g., the Operation is missing from the model).
Notes	<p>An option is provided to filter out Sequence messages:</p> <ul style="list-style-type: none"> • Targeting one or more element types. • Targeting one or more element stereotypes.
Rationale	Verify the integrity of the Sequence diagram by reporting messages referring to Operations that are not present in the model.

Rule ID	<i>SequenceObjectsWithNoClassifier</i>
Description	Verify Lifelines have a Classifier defined.
Notes	None.
Rationale	Lifelines should have a Classifier defined, and not refer to anonymous Objects.

State Machine diagram rules

Rule ID	<i>TransitionTriggerSpecified</i>
Description	Verify that any defined Trigger for a Transition has a specification value.
Notes	<p>The rule verifies that the Trigger:</p> <ul style="list-style-type: none"> • Has a type defined (either Call, Signal, Change, or Timer). • For a Call, that there is an Operation specified for the Trigger, and the Operation reference is not unresolved. • For a Timer, that there is non-empty text specification defined. • For a Change, that there is non-empty text specification defined. • For a Signal, that there is a Signal specified for the Trigger, and the Signal reference is not unresolved.
Rationale	A State Transition Trigger should have a specification defined to document the event that will cause the Trigger to fire.

Rule ID	<i>AssignedBehaviors</i>
Description	Check for States with entry/do/exit behaviors that do not reference specified model behaviors.
Notes	<p>The rule verifies that for each declared behavior (entry, do, or exit) of a State, the <i>Behavior</i> and the <i>Code</i> fields of the (pseudo) Operation are not both null.</p> <p>Code, pseudo-code/text, or a reference to a model behavior (i.e., an Operation, Activity, or Interaction) all count as the specification of actual behavior.</p>
Rationale	A State behavior should have some implementation information defined, beyond just its name.

Use Case diagram rules

Rule ID	<i>UseCaseToUseCaseConnectors</i>
Description	Verify Use Case to Use Case connector types are UML compliant.
Notes	By default, checks if the connector type is either <i>UseCase</i> (which covers <i>include</i> and <i>extend</i> relationships) or <i>Generalization</i> . Optionally add <i>Dependency</i> to the Connector filter criteria in a custom rule.
Rationale	As per the UML specification.

Rule ID	<i>ActorToUseCaseConnectors</i>
Description	Verify Actor to Use Case connector types are UML compliant.
Notes	By default, checks if the connector type is either <i>UseCase</i> or <i>Association</i> .
Rationale	As per the UML specification.

Rule ID	<i>ActorWithNoDescription</i>
Description	Report Actor elements with no description/notes.
Notes	The validation does not currently check for a Linked Document associated with the Actor.
Rationale	It is recommended to provide a clear description of the role performed by an Actor, and not just to rely on its name which can be a source of ambiguity.

Rule ID	<i>UseCaseWithNoDescription</i>
Description	Report Use Case elements with no description/notes.
Notes	The validation does not currently check for a Linked Document associated with the Use Case.
Rationale	A minimum description of the context in which the Use Case is set to execute should be provided, even if the Use Case is supported in the model by a structured specification or Activity diagram.

Rule ID	<i>DuplicateActorName</i>
Description	Report duplicate Actor elements.
Notes	None.
Rationale	<p>Actors should:</p> <ul style="list-style-type: none"> • Have a unique name. • Be defined one time in the repository (typically in a library). • Reused wherever needed.

Rule ID	<i>MissingUseCaseToRequirement</i>
Description	Report Use Cases with no UML compliant relationship to one or more Requirements.
Notes	<p>By default, checks for the UML Realization relationship from the Use Case to the Requirement.</p> <p>Type and Stereotype filters are available at the Use Case and Requirement level for custom rule creation.</p>
Rationale	Use Cases should relate back to the functional Requirements for which they provide context.

Rule ID	<i>UseCaseWithNoConstraints</i>
Description	Report Use Cases with no pre- and post-conditions defined.
Notes	<p>Type and Stereotype filters are available at the Use Case and Constraint level for custom rule creation.</p> <p>By default, the rule checks for <u>both</u> pre- and post-conditions to be present.</p>
Rationale	<p>A well-defined Use Case states:</p> <ul style="list-style-type: none"> • The pre-conditions required for the Use Case to execute successfully. • The state of the system after the execution of the Use Case, in case of success as well as in the case of failure. <p>Constraints are particularly useful for defining black box test cases.</p>

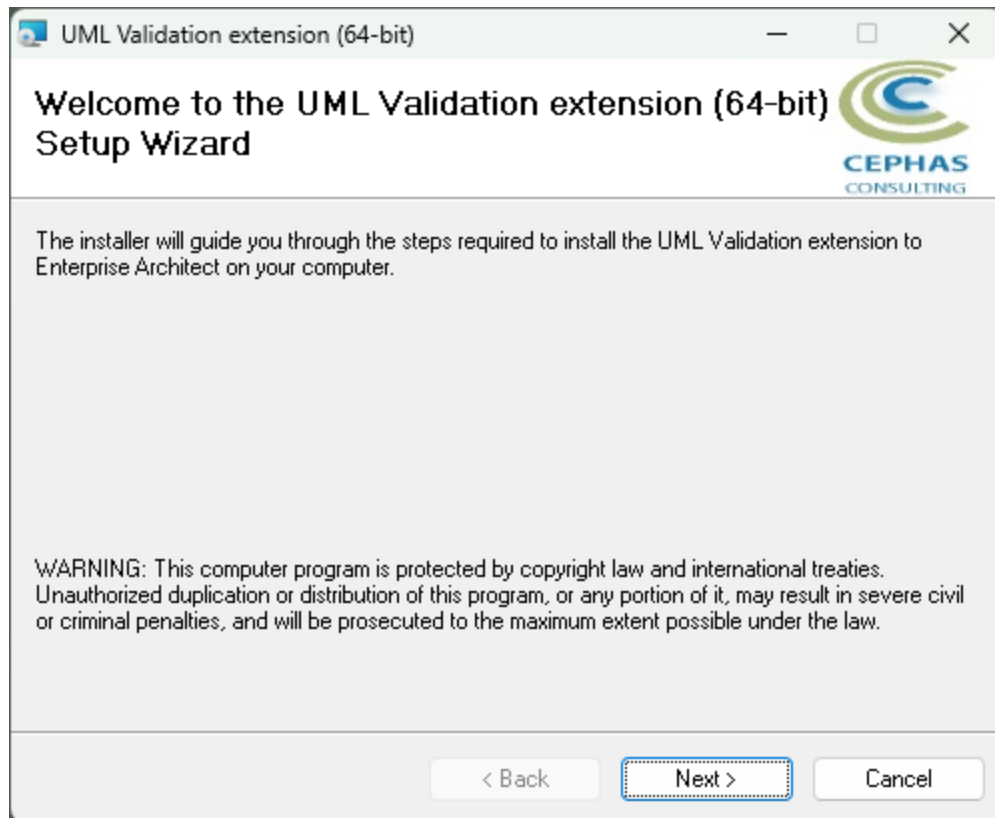
Rule ID	<i>DiagramObjectsWithNoConnectors</i>
Description	Report Use Cases that are visually represented on one or more diagrams and do not have any relationships with other elements in the model.
Notes	<p>By default, all diagram types are searched, with the exception of Sequence diagrams. One or more specific diagram types can be specified in a custom rule.</p> <p>Connectors to Note elements are always excluded from the query.</p> <p>This rule is not specific to Use Case elements and can be reused in other contexts.</p>
Rationale	Find Use Cases that may be in an unfinished state. Use Cases are expected to have at a minimum a relationship with an Actor, or with another Use Case.

Installation

The installation process is the same for both the trial and the full version.

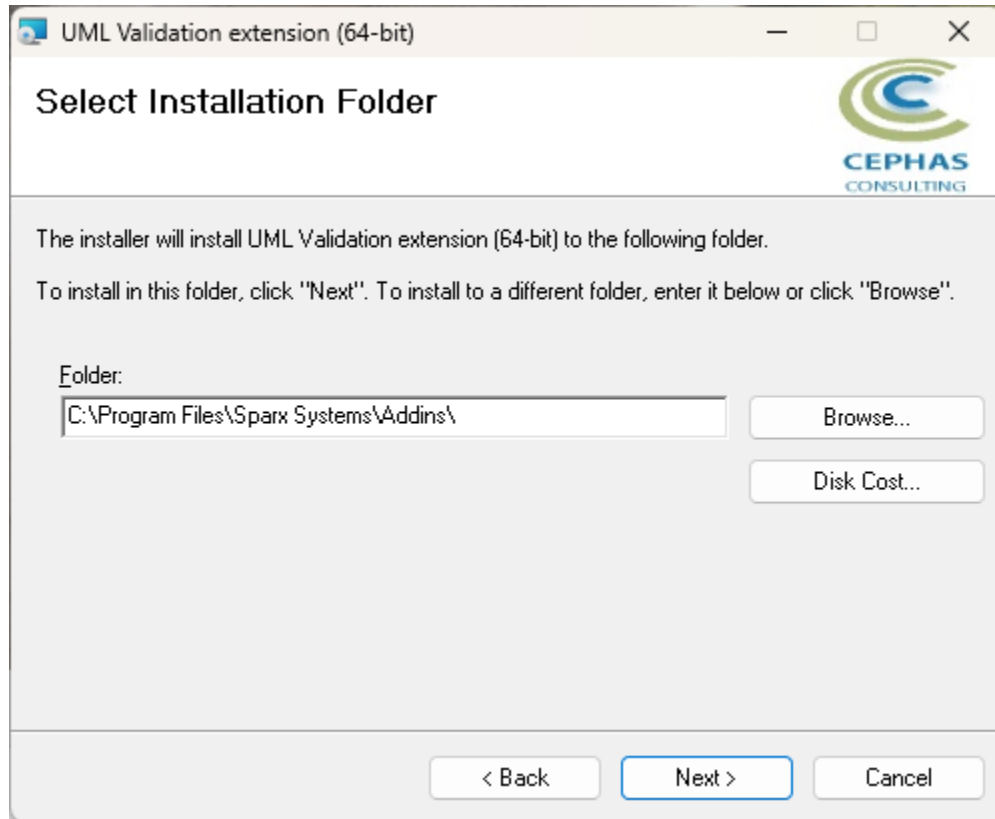
First, **exit any running instances of Enterprise Architect**, then launch the “setup.exe” program and follow the on-screen instructions.

For example, for the 64-bit version:



The installation will attempt to update the Windows registry, so the User needs to ensure that s/he has sufficient privileges to run the setup program.

The recommended install path is to place the DLL and any supporting files in an *Addins* folder in the Sparx Systems installation directory, e.g., for the 64-bit version:

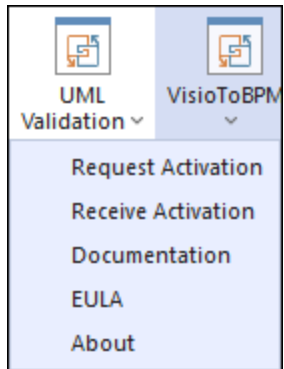


Note that while older versions of the software can be upgraded, it is recommended to uninstall and reinstall the software (Custom rule sets will be preserved).

Should the installation fail for any reason other than insufficient User privileges, please take appropriate screenshots and email the data to the [support](#) address listed at the bottom of this document.

Verifying the installation

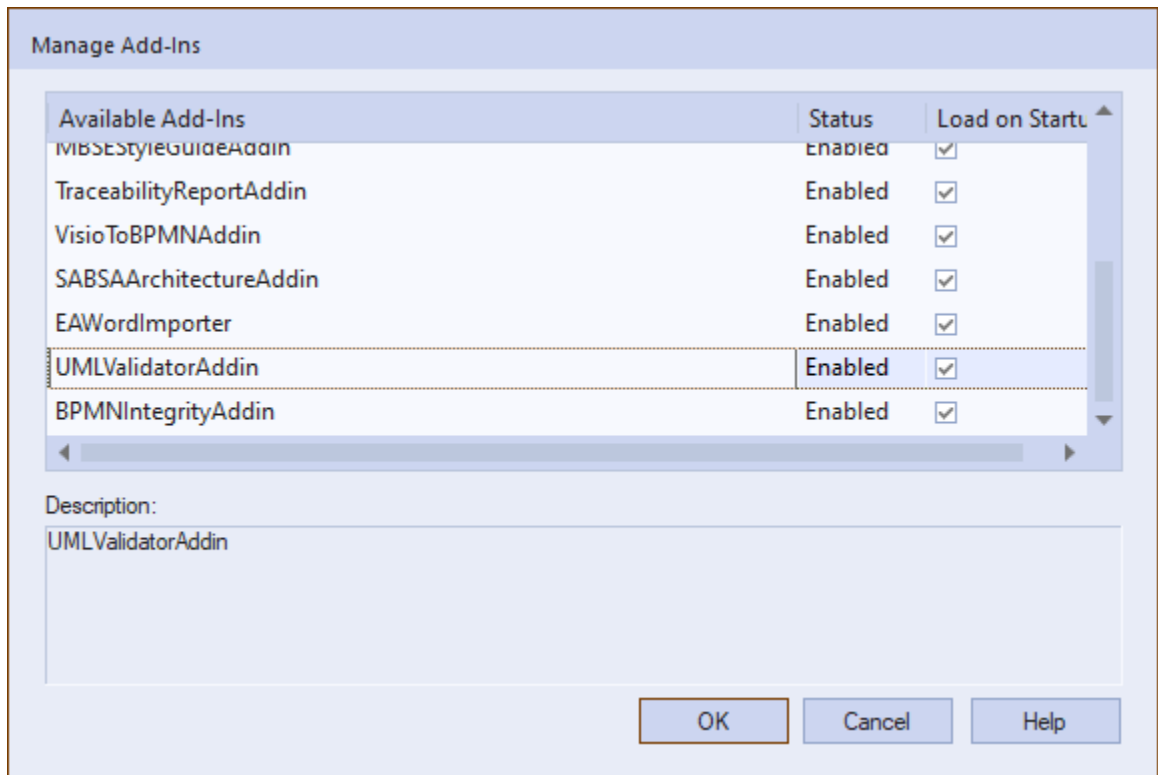
Bring up Enterprise Architect, without necessarily opening a repository, and verify that the *UML Validation* extension has been loaded using the *Specialize* → *Add-Ins* ribbon panel:



Should the extension not be present, use the same panel and select:



Then check if the *UML Validation* extension is present and enabled:



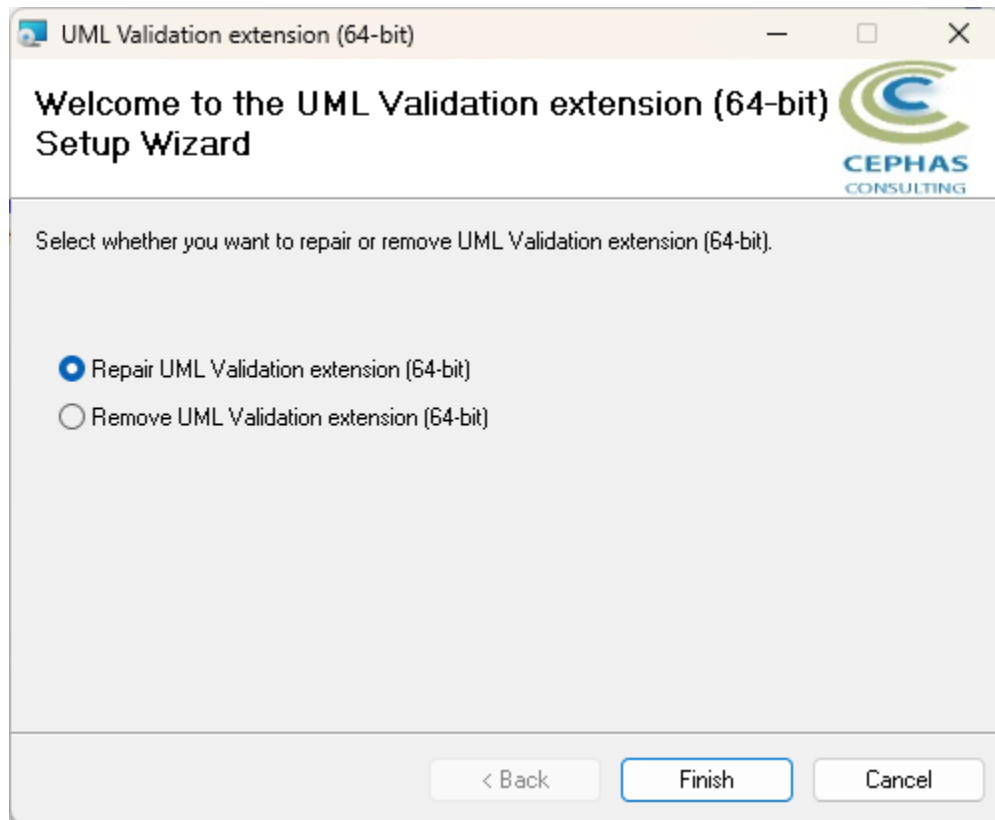
If an error status is shown, this typically means that either:

- The installation process failed and that the DLL cannot be located in the Windows registry, or in the file system.
- The installation did succeed but the DLL file was later moved or deleted.

If the *UML Validation* entry itself is not found, then the extension installation did not complete successfully.

To fix an incorrect installation:





- Exit out of all instances of Enterprise Architect.
- Launch the setup process again. The installer will automatically provide a repair option:



If, after the repair procedure, the *UML Validation* extension is still not loaded correctly in Enterprise Architect, remove the program through the Windows control panel and start the installation process over.

UML® Validation User Guide

At the completion of a successful installation the following files are installed in the selected directory e.g., for the 32-bit version:

(C:) > Program Files (x86) > Sparx Systems > Addins > UML Validation				
Name	Type	Size	Date modified	
 Cephas_Software_EULA.pdf	Adobe Acrobat D...	139 KB	10/4/2022 3:15 PM	
 UMLValidationExtension.pdf	Adobe Acrobat D...	609 KB	6/20/2023 2:14 PM	
 UMLValidatorAddin.dll	Application exten...	177 KB	6/20/2023 2:30 PM	
 UMLValidatorAddin.tlb	TLB File	7 KB	6/20/2023 2:30 PM	

Licensing

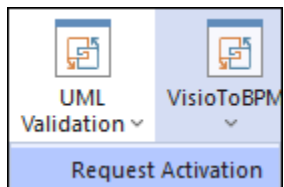
Trial version

The software installation automatically loads the trial version license key. No further action is required on the part of the end User.

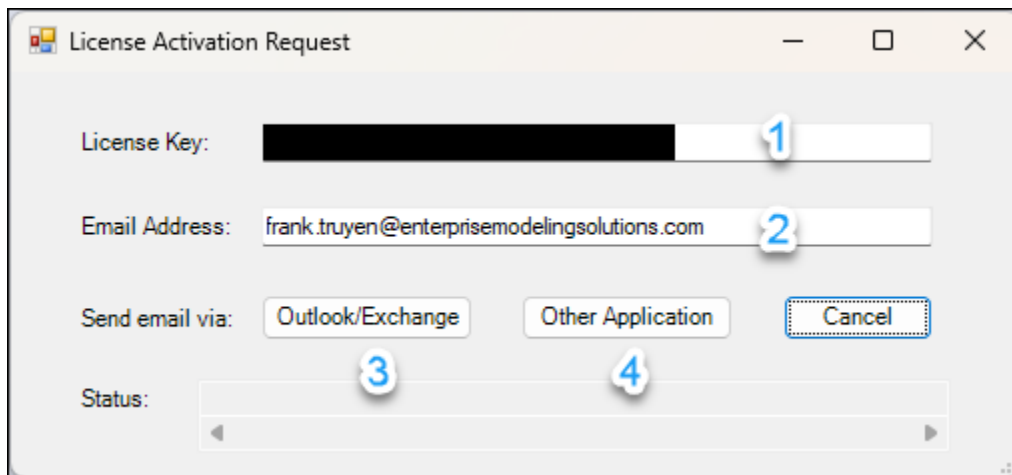
Licensed version

Once the full version of the product has been [purchased](#), a license key will be provided by Cephas Consulting via email. Once that key has been received, proceed with the following steps:

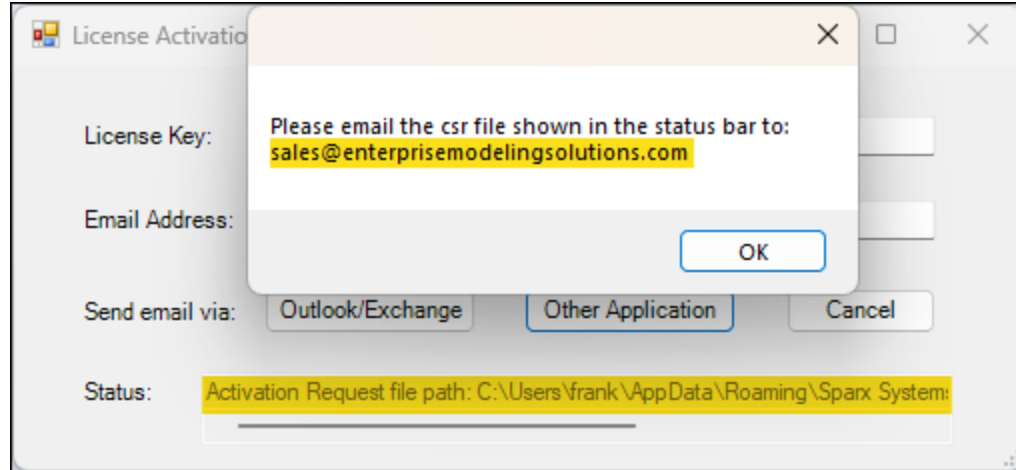
Request the license activation



Note: the license, once activated, is associated with your Windows login User ID (without the network domain information).



1. Paste in the license key from the email you received.
2. Provide the email address where you wish to receive the license activation notification.
3. Click this option if you use Microsoft Outlook with Exchange as your email application. This automatically emails your activation request (CSR file) to Cephas Consulting.
4. Otherwise, click this option to send the email yourself. The file path to the generated activation request (CSR file) is displayed in the Status bar of the dialog:



Attach the CSR file to your email and send the request to:

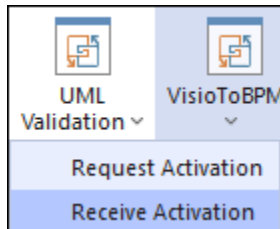
sales@enterprisemodelingsolutions.com

Receive the license activation

Upon receiving your activation request, Cephas Consulting replies with a license key file named "Fixed.keys".

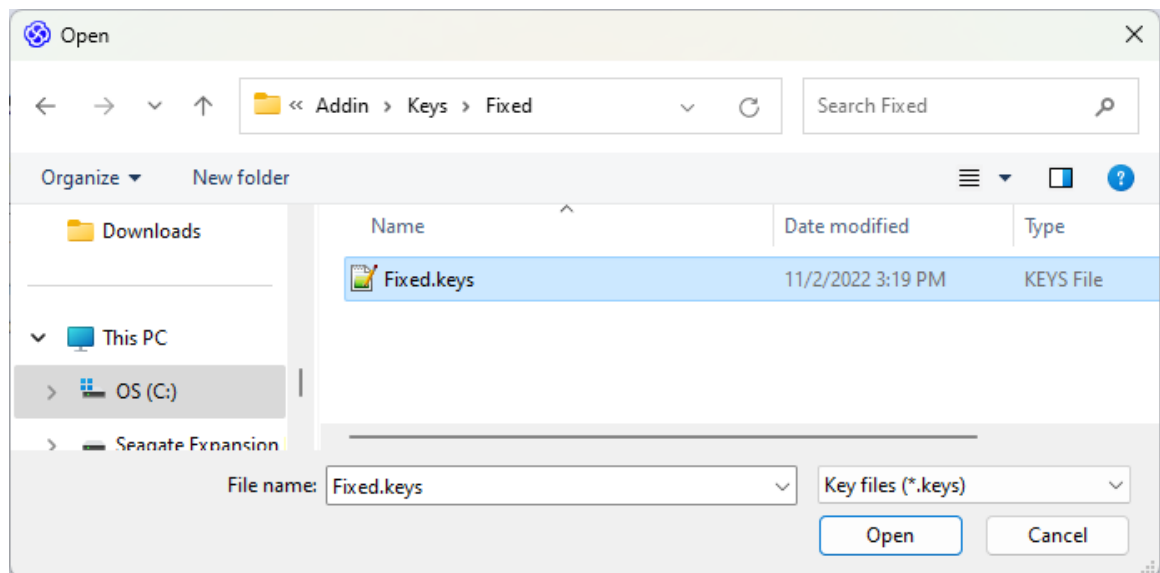
Caveat: this file can be saved anywhere (e.g. for backup purposes) on your system, but its name should not be changed.

To install the license key file, open Enterprise Architect and in the *Specialize* → *Add-Ins* ribbon panel, select:

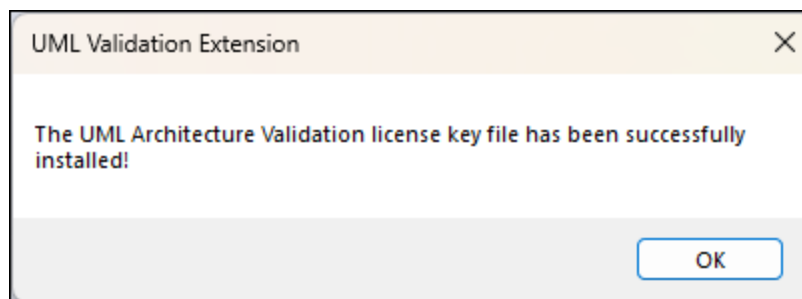


UML® Validation User Guide

Next, select the file you copied out of the received email. For example:



A confirmation of the license activation is then displayed:



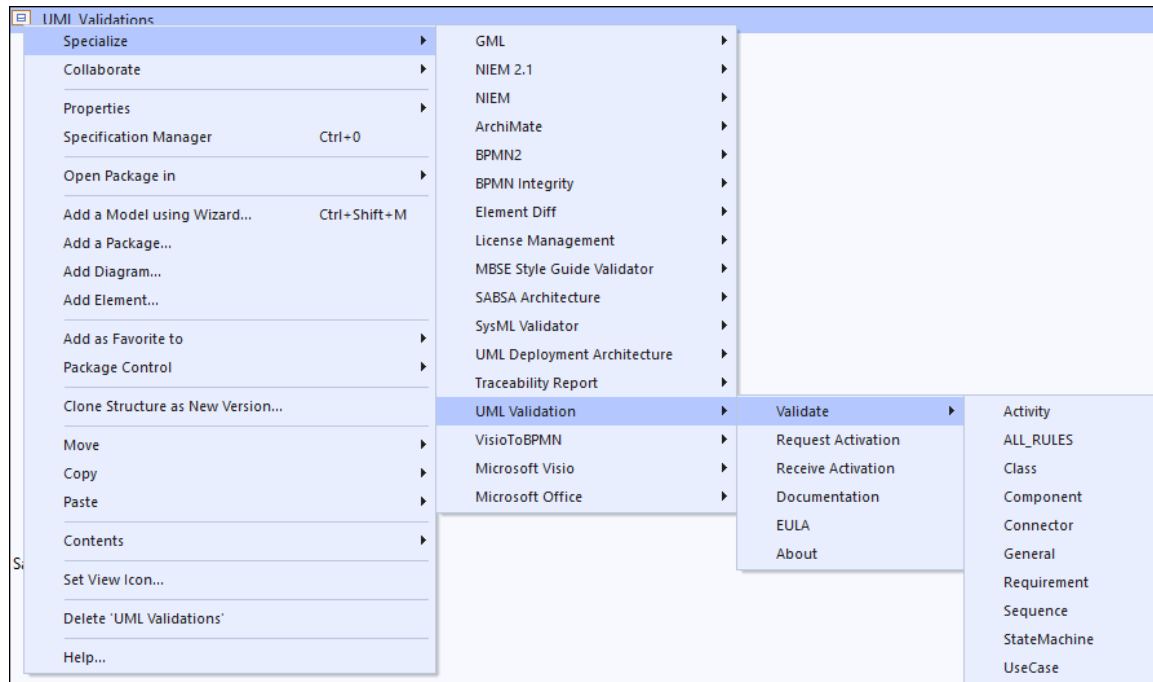
Next **exit the Enterprise Architect tool** and **restart it** in order to fully enable the extension.

Running the validation

Package level validation

Note: all child Packages in the hierarchy (if any) are included by default (Packages can be filtered out by setting the [IgnorePackageWithStatus](#) property in the rule set).

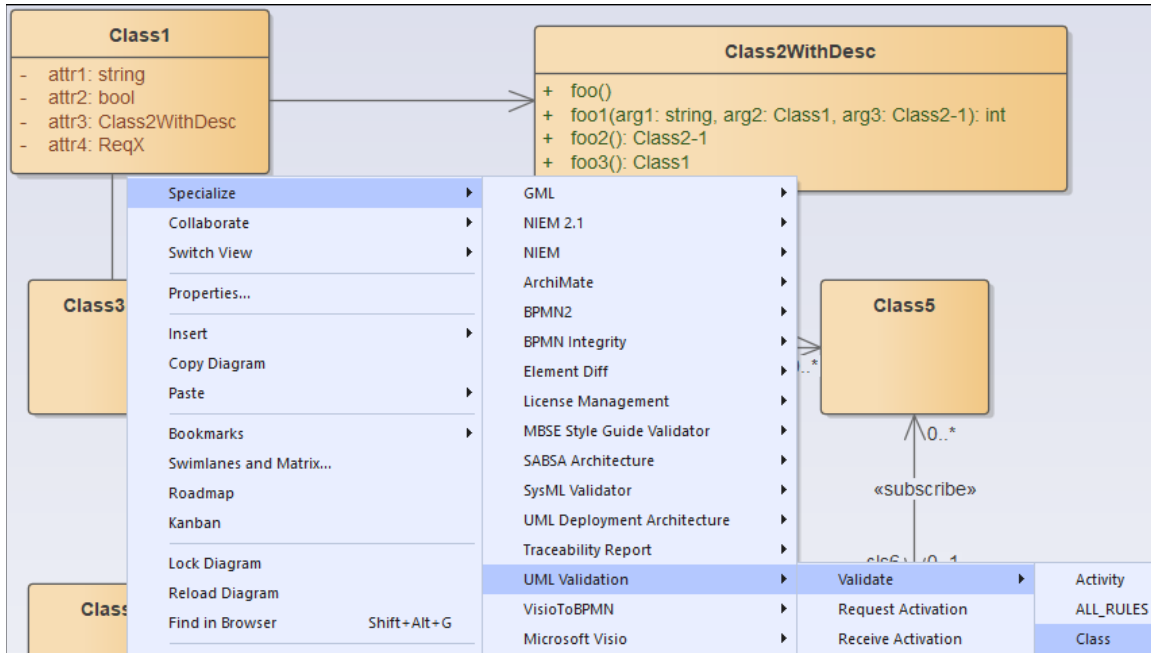
Right click a Package, or the top of a Package hierarchy, and select any of the available [rule sets](#):



Caveat: the larger the package hierarchy selected, the more time the validation will take!

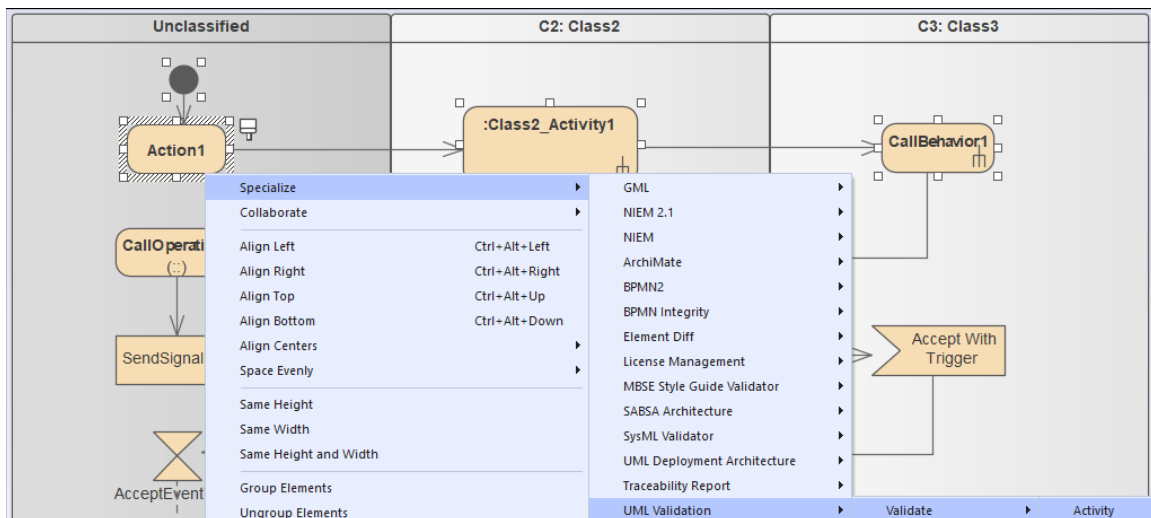
Diagram level validation

Right click a diagram background, and select the rule set appropriate for this type of diagram. For example, in the context of a Class diagram:



Element level validation

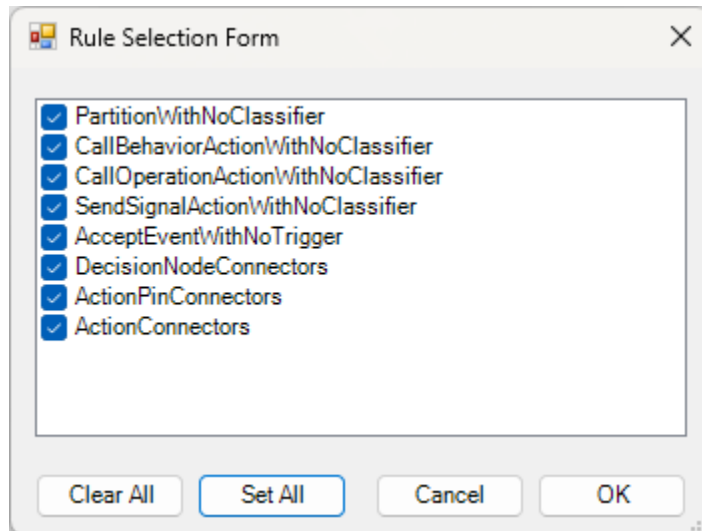
Select one or more diagram elements, then **right click** and select the rule set appropriate for this type of selection. For example, in the context of an Activity diagram:



User Interface

Unless the [RunSilent](#) option is enabled for the rule set, a form displays when launching the validation for a specific rule set (except for “ALL_RULES”).

For example, for Activity:



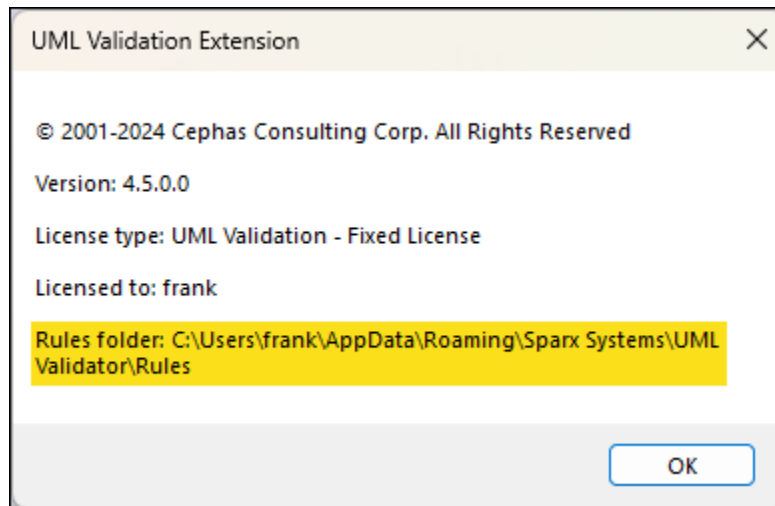
This allows rules to be enabled/disabled for a specific validation. To disable rules by default, create a custom rule set (see next section).

Customizing the Rule Set

Caveat: it is best not to modify the default rule sets!

Instead:

- Make a copy of the xml file **in the same** Rules folder. The location of that folder can be easily determined by selecting the [“About” option of the extension menu](#):



- Optionally move the original xml file to an archive folder.
- Open the file in any XML editor and make the required changes in that copy.

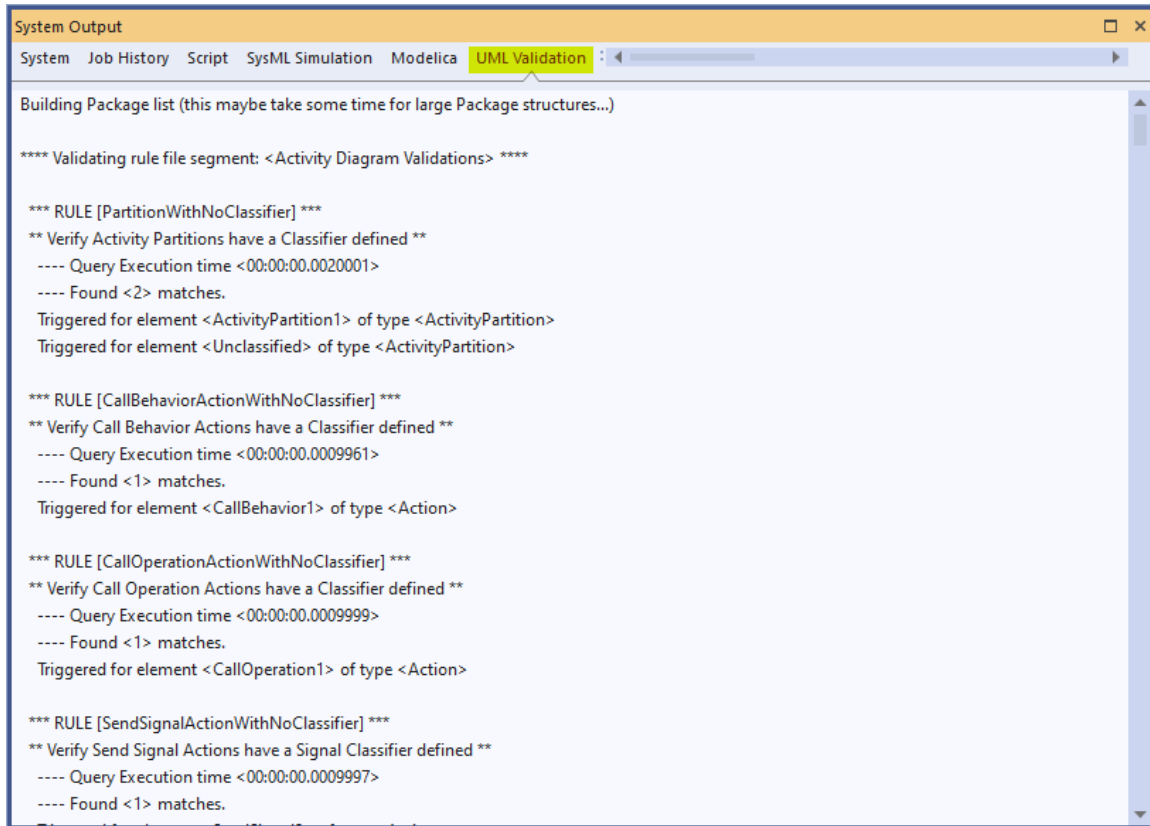
When adding a new rule to a set, ensure that it is given a unique ID (i.e., name) value.

Custom rule files are automatically detected in the Rules folder and [made available for selection](#).

Rule changes can be made in between validations, while EA is running!

Verifying the result set

During execution the “UML Validation” tab in the System Output window will automatically open and display the results of the validation:



```

System Output
System Job History Script SysML Simulation Modelica UML Validation

Building Package list (this maybe take some time for large Package structures...)

**** Validating rule file segment: <Activity Diagram Validations> ****

*** RULE [PartitionWithNoClassifier] ***
** Verify Activity Partitions have a Classifier defined **
---- Query Execution time <00:00:00.0020001>
---- Found <2> matches.
Triggered for element <ActivityPartition1> of type <ActivityPartition>
Triggered for element <Unclassified> of type <ActivityPartition>

*** RULE [CallBehaviorActionWithNoClassifier] ***
** Verify Call Behavior Actions have a Classifier defined **
---- Query Execution time <00:00:00.0009961>
---- Found <1> matches.
Triggered for element <CallBehavior1> of type <Action>

*** RULE [CallOperationActionWithNoClassifier] ***
** Verify Call Operation Actions have a Classifier defined **
---- Query Execution time <00:00:00.0009999>
---- Found <1> matches.
Triggered for element <CallOperation1> of type <Action>

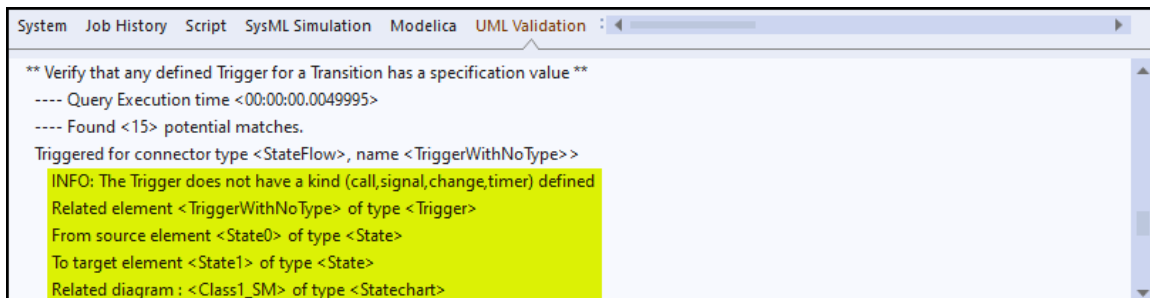
*** RULE [SendSignalActionWithNoClassifier] ***
** Verify Send Signal Actions have a Signal Classifier defined **
---- Query Execution time <00:00:00.0009997>
---- Found <1> matches.
  
```

Every rule match is listed in this window using the format:

Triggered for <connector, element, Attribute, Operation or diagram name ...>
{additional information}

For rules relating to connectors, the associated objects (source and/or target) and diagram/s are reported on additional output lines.

In other cases, additional output lines may be added to include related diagrams, object Classifiers, etc. For example:



```

System Job History Script SysML Simulation Modelica UML Validation

** Verify that any defined Trigger for a Transition has a specification value **
---- Query Execution time <00:00:00.0049995>
---- Found <15> potential matches.
Triggered for connector type <StateFlow>, name <TriggerWithNoType>>
INFO: The Trigger does not have a kind (call,signal,change,timer) defined
Related element <TriggerWithNoType> of type <Trigger>
From source element <State0> of type <State>
To target element <State1> of type <State>
Related diagram : <Class1_SM> of type <Statechart>
  
```


Active output lines are the ones that are NOT:

- blank
- starting with the characters '*', '+' or '-'.
- starting with the text 'INFO:'

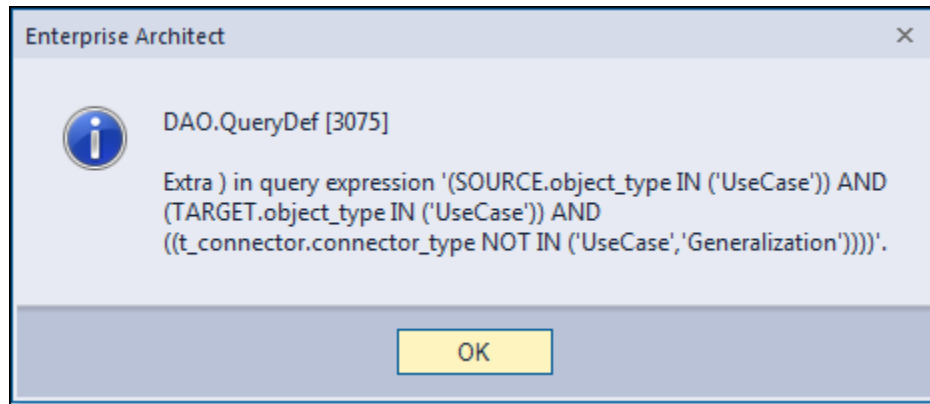
Single click an active output line to automatically locate its related element, Attribute, Operation or diagram in the Browser.

Double click an active output line to open the element, Attribute or Operation properties, or to open the diagram associated with the rule.

For connector related rules, a single or double click will attempt to open up at least one diagram where the connector is graphically represented (if any), and select the connector in that diagram (you may need to scroll the window to see the selection).

Troubleshooting

Should a SQL statement fail to execute properly, Enterprise Architect will display an error message dialog similar to this:



As of version 16.x of Enterprise Architect this type of error is not relayed back to the application which is unaware that a problem occurred.

Please follow this procedure:

- Take a screenshot of the error message.
- Locate the DBError.txt file in %APPDATA%\Sparx Systems\EA and include it in your message.
- Before dismissing the dialog box, look at the System Output window to determine the rule being executed at the time of the error. For example:

*** RULE [CallBehaviorActionWithNoClassifier] ***

- If you are validating the repository using a [customized rule set](#), please include that xml file in the data provided back to Cephas.
- Also provide:
 - Your database type (Microsoft Access, SQL Server, Oracle, etc.) and version number.
 - The version of the UML Validation extension by selecting the [“About” option of the extension menu](#).
 - The version of Enterprise Architect being used.
 - Your operating system and any other execution environment information that may be relevant.

Support and contact information

Use the contact information below for any installation or runtime issues with the extension.

Feature requests or suggestions for improvement are also welcome!

Contact: Frank Truyen

Email : contact@enterprisemodelingsolutions.com

Phone : 208-462-4863