SysML® Validation Extension

Disclaimer

Version 5.0.x of the *SysML® Validation Extension*:

- Has been successfully tested for deployment with version16.x of Enterprise Architect (a separate installer is provided for the 32- and 64-bit versions of the tool). There is no guarantee that versions prior to EA 16.x will work properly, and no effort will be made to support earlier releases.
- 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 the 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, including:

- 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 database 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 eight (8) consecutive days.
- Only the first five (5) 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 each rule configuration file can be set to exclude all Packages with a particular Status value (see the screenshot below). This allows filtering out sandbox or other non-production folders 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 SysML model constructs being validated: Activity, Block Definition elements, Bock Definition features, Internal Block, Requirements, Interaction (Sequence), State Machine, and Use Case. In addition, a General rule set validates aspects of the model not specific to SysML.

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="5" RunSilent="False" VerboseLevel="3" 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>
```

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 ouput 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 SysML 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’).

Example (Features) rule set

```
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<!DOCTYPE Validations >
<Validations Version="6" RunSilent="False" VerboseLevel="3" IgnorePackageWithStatus="">
    <FEAT Description="Attribute, Operation, and other feature related rules">
        <Attributes>
            <UnresolvedDataType Enabled="True" ID="UnresolvedAttributeDataType" Description =
            "Report Attributes with unresolved Classifiers">
                <ElementType Name="'Signal'" MatchFilterCriteria="True"/>
                <ElementStereotype Name="" MatchFilterCriteria="True" />
            </UnresolvedDataType>
        </Attributes>
        <Operations>
            <UnresolvedOperation Enabled="True" ID="UnresolvedBlockOperation" Description =
            "Report unclassified Operations (i.e. with a null or unresolved return type)">
                <ElementType Name="'Class'" MatchFilterCriteria="True"/>
                <ElementStereotype Name="'block','InterfaceBlock'" MatchFilterCriteria="True" />
            </UnresolvedOperation>
            <UnresolvedOperationParameter Enabled="True" ID="UnresolvedBlockOperationParameter"
            Description = "Report Operations/Receptions with unresolved parameter Classifiers">
                <ElementType Name="'Class'" MatchFilterCriteria="True"/>
                <ElementStereotype Name="'block','InterfaceBlock'" MatchFilterCriteria="True" />
            </UnresolvedOperationParameter>
            <UnusedOperation Enabled="True" ID="UnReferencedBlockOperation" Description = "Report
            Operations which are not referenced in any Sequence message, Call Operation Action,
            Accept Event Action, State Machine  behavior (entry/do/exit) or transition
            trigger/effect">
                <ElementType Name="'Class'" MatchFilterCriteria="True"/>
                <ElementStereotype Name="'block','InterfaceBlock'" MatchFilterCriteria="True" />
            </UnusedOperation>
            <UnmatchedReceptionToSendSignal Enabled="True" ID=
            "UnmatchedBlockReceptionToSendSignalAction" Description = "Report (Signal) Receptions
            for which there is no corresponding Send Signal Action">
                <ElementType Name="'Class'" MatchFilterCriteria="True"/>
                <ElementStereotype Name="'block','InterfaceBlock'" MatchFilterCriteria="True" />
            </UnmatchedReceptionToSendSignal>
            <UnmatchedReceptionToAcceptSignal Enabled="True" ID=
            "UnmatchedBlockReceptionToAcceptSignalAction" Description = "Report (Signal)
            Receptions for which there is no corresponding Accept Signal Action">
                <ElementType Name="'Class'" MatchFilterCriteria="True"/>
                <ElementStereotype Name="'block','InterfaceBlock'" MatchFilterCriteria="True" />
            </UnmatchedReceptionToAcceptSignal>
        </Operations>
    </FEAT>
</Validations>
```

Activity validation rules

Rule ID	*PartitionWithNoClassifier*
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	*CallBehaviorActionWithNoClassifier*
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	*CallOperationActionWithNoClassifier*
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	*SendSignalActionWithNoClassifier*
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	*ActionPinWithNoClassifier*
Description	Verify Action Pins have a Classifier defined.
Notes	Includes checking if the specified Classifier exists in the current repository. By default, Action Pins named 'target', automatically created by EA for simulation purposes, are filtered out. The validation check to see if the Action Pin on the other side of the Object Flow relationship has a Classifier, in which case no error is reported.
Rationale	Ensure the structure of the object/s transmitted through the Action Pin is supported by a formal definition. Action Pins referring to Activity Parameters are excluded since the Classifier should be defined at the Parameter level (see next rule).

Rule ID	*ActivityParameterWithNoClassifier*
Description	Verify Activity Parameters have a Classifier defined.
Notes	Includes checking if the specified Classifier exists in the current repository.
Rationale	Ensure the structure of the object/s transmitted through the Activity Parameter is supported by a formal definition.

Rule ID	*ActionsWithNoParentPartition*
Description	Verify every Action is a child of a Partition or an Interruptible Activity Region
Notes	Interruptible Regions can span multiple Partitions. EA uses a parent-child relationship between a Region and its content, similar to the parent-child relationship between a Partition and its content. Optionally, for models that are not using Partitions, update the rule to accept Activity as an additional valid Parent.
Rationale	Ideally the Actions of an Activity are assigned to a Partition that can itself be allocated to a structural element (e.g., a Block) – see the above rule *PartitionWithNoClassifier.* This enables all the Actions performed by a structural element (e.g. a Block) across any Activity diagram to be easily traced.

Rule ID	*CallBehaviorActionWithNoMatchingAllocationToClassifier*
Description	Verify Call Behavior Actions can be traced to the Block that owns the behavior being called.
Notes	Traceability is confirmed when the Action: • Is owned by a Partition classified by the Block owning the behavior being called. • Is graphically enclosed in a Partition classified by the Block owning the behavior being called (e.g., the Action is owned by an Interruptible Activity Region, but is graphically located within the boundaries of the Partition). • Is owned by an Activity that is a child of the Block owning the behavior being called. • Has a SysML allocate relationship to the Block owning the behavior being called.
Rationale	As per the UML/SysML specification.

Rule ID	*CallOperationActionWithNoMatchingAllocationToClassifier*
Description	Verify Call Operation Actions can be traced to the Block that owns the Operation being called.
Notes	Traceability is confirmed when the Action: • Is owned by a Partition classified by the Block owning the Operation being called. • Is graphically enclosed in a Partition classified by the Block owning the Operation being called (e.g., the Action is owned by an Interruptible Activity Region, but is graphically located within the boundaries of the Partition). • Is owned by an Activity that is a child of the Block owning the Operation being called. • Has a SysML allocate relationship to the Block owning the Operation being called.
Rationale	As per the UML/SysML specification.

Rule ID	*AcceptEventWithNoTrigger*
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	*SignalAcceptEventWithNoMatchingReception*
Description	Verify Accept Event Actions with a Signal Trigger have a matching Reception defined in at least one Block.
Notes	None.
Rationale	A Reception is the required mechanism whereby a Signal event can be dispatched to a Classifier able to handle that Event.

Rule ID	*SignalAcceptEventWithNoMatchingReceptionInAllocatedBlock*
Description	Verify Accept Event Actions with a Signal Trigger have a matching Reception defined in a Block that can be traced from the Action.
Notes	The following modeling constructs are accepted as a valid trace to a Block that has a Reception defined for that Signal: 1) A SysML Allocate relationship exists from the Action to the Block. 2) The Action resides in a Partition that is classified by that Block. 3) The Action is graphically enclosed in a Partition classified by the Block (e.g., the Action is owned by an Interruptible Activity Region, but is graphically located within the boundaries of the Partition). 4) The Action is by an Activity that itself is a child of that Block.
Rationale	Similar to the previous rule, but also checks the consistency between behavior expressed in an Activity diagram and the structure specified in a Block diagram.

Rule ID	*SendSignalWithNoTrigger*
Description	Verify Send Signal Actions have a corresponding Trigger defined in an Activity or State Machine.
Notes	None.
Rationale	A Send Signal Action without at least one matching Trigger would not be acted upon.

Rule ID	*SignalAcceptEventToSendSignal*
Description	Verify Accept Event Actions with a Trigger of Type Signal have a corresponding Send Signal defined.
Notes	None.
Rationale	Without a Send Signal Action, the Trigger would never fire.

Rule ID	*IncompatibleActivityParametersAndActionPins*
Description	Verify Call Behavior Actions Pins are defined in accordance with the properties of the corresponding Activity Parameters.
Notes	Validates that: • Every Action Pin of the Call Behavior Action has a corresponding Activity Parameter. • The multiplicity values are the same. • The stereotype values are the same. • The data type/classifier values are the same. • The direction, exception and stream properties are the same.
Rationale	Currently Enterprise Architect does not fully enforce this correlation.

Rule ID	*IncompatibleConnectedActionPinClassifiers*
Description	Validate connected Action Pins have compatible Classifiers.
Notes	Relates to Action Pins connected together via Object Flow. By default, the rule verifies if the two Classifiers (when not the same) are nonetheless compatible by means of a Generalization hierarchy. An option in the rule provides the ability to ignore cases where only one side (either the source or target Action Pin) is not classified. This is enabled by default.
Rationale	Ensure the input and output specifications of an Object Flow are of the same type, or are compatible with one another.

Rule ID	*DecisionNodeConnectors*
Description	Validate that incoming and outgoing connectors of Decision Nodes are compliant with the UML/SysML specification.
Notes	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).
Rationale	As per the UML/SysML specification.

Rule ID	*MergeNodeConnectors*
Description	Validate that incoming and outgoing connectors of Merge Nodes are compliant with the UML/SysML specification.
Notes	Reports on cases where the Merge Node has: 1) Control Flow inputs but Object Flow outputs. 2) Object Flow inputs but Control Flow outputs. 3) Both Object and Control Flow inputs. 4) More than one output connector.
Rationale	As per the UML/SysML specification.

Rule ID	*ForkJoinNodeConnectors*
Description	Validate that incoming and outgoing connectors of Fork and Join Nodes are compliant with the UML/SysML specification.
Notes	Enterprise Architect uses the same metatype for both Fork and Join Nodes, which is technically incorrect. To make the distinction, the rule looks at the number of incoming and outgoing connectors. Reports on cases where the Fork Node has: 1) Control Flow input but Object Flow outputs. 2) Object Flow input but Control Flow outputs. Reports on cases where the Join Node has: 1) At least one Object Flow input but a Control Flow output. 2) Control flow only inputs but an Object Flow output.
Rationale	As per the UML/SysML specification.

Rule ID	*IncompatibleConnectedActionPinClassifiers*
Description	Validate connected Action Pins have compatible Classifiers.
Notes	Compatible implies that the two Classifiers: • Are the same. • Are not the same, but yet compatible via their membership in the same generalization hierarchy (this option is enabled by default). Another option of the rule (also enabled by default) accepts the case where only the source or the target of the Object Flow connector is classified. This modeling practice saves time and reduces diagram clutter.
Rationale	Ensure that the source and target Action Pins of an Object Flow are logically coherent with regards to the type of items sent and received.

Rule ID	*ActionPinConnectors*
Description	Validate only an Object Flow or an Interrupt 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/SysML specification.

Rule ID	*ActionConnectors*
Description	Validate only a Control Flow or an Interrupt Flow connects Actions together.
Notes	None.
Rationale	As per the UML/SysML specification.

Rule ID	*ActionFlowToAllocatedBlocks*
Description	For each source element of a pair of Actions connected via Control Flow, and where the Actions are allocated to different Blocks via Activity Partition containment, verify that its allocated Block has a Part property, the type of which corresponds to the Block typing the Partition containing the target Action. The Part is typically created by EA as the result of connecting two Blocks via Association.
Notes	None.
Rationale	The intent is to check if Block-to-Block interactions, as specified in Activity diagram Control Flow connectors, are also reflected at the Block Definition level.

Rule ID	*ActionPinFlowToAllocatedBlocks*
Description	For each source element of a pair of Action Pins connected via Object Flow, and where the parent Actions of the Pins are allocated to different Blocks via Activity Partition containment, verify that its allocated Block has a Part property, the type of which corresponds to the Block typing the Partition containing the target Pin's parent Action. The Part is typically created by EA as the result of connecting two Blocks via Association.
Notes	None.
Rationale	The intent is to check if Block-to-Block interactions, as specified in Activity diagram Object Flow connectors, are also reflected at the Block Definition level.

Rule ID	*InterruptFlowConnectors*
Description	Verify that Interrupt Flow connectors are compliant with the UML/SysML specification
Notes	Verifies that the source of the connector is inside an Interruptible Activity Region, and the target is either a child of a Partition or of an Activity. Note that the specification allows an Interrupt Flow to connect two Actions, or two Action Pins. EA only supports the former.
Rationale	As per the UML/SysML specification.

Block Definition rules

Rule ID	*BlockAssociationEndProperties*
Description	Verify for each (navigable) connector end that role name and multiplicity are set.
Notes	Checks that for both Reference Association and Part Association connectors the role name and multiplicity properties defined.
Rationale	Since each navigable connector end represents a Property of the source Block, that Property should be assigned a name, the same way all manually created Properties are named. Multiplicity is assumed 1 when not specified, but assigning a multiplicity removes ambiguity in the model.

Rule ID	*BlockWithNoDescription*
Description	Report Blocks with no description/notes.
Notes	The validation does not currently check for a Linked Document associated with the Block.
Rationale	It is recommended to provide at least a minimal description of the Block, and not just to rely on the name which can be a source of ambiguity.

Rule ID	*PartsWithNoClassifier*
Description	Ensure Parts (of a Block) have a Classifier defined.
Notes	Includes checking if the specified Classifier exists in the current repository.
Rationale	Properties should have a Classifier assigned (typically a primitive, a Value Type, or a Block) to document their actual structure.

Rule ID	*PortsWithNoClassifier*
Description	Ensure Ports (of a Block) have a Classifier defined.
Notes	Includes checking if the specified Classifier exists in the current repository.
Rationale	Both Full and Proxy Ports should have a Classifier assigned (typically a Block for Full Ports, and an Interface Block for Proxy Ports) to document their actual structure and/or behavior.

Rule ID	*FullPortClassifier*
Description	Ensure Full Ports (of a Block) are classified by a Block (not an Interface Block).
Notes	None.
Rationale	This is currently not enforced by the tool.

Rule ID	*InterfaceBlockUsage*
Description	Verify that an Interface Block is used as a Classifier of at least one Proxy Port
Notes	None.
Rationale	Unused Interface Blocks should be removed from the model. Otherwise, they indicate a gap in the structural model.

Feature related rules

Rule ID	*UnresolvedAttributeDataType*
Description	Report Signal Attributes with unresolved Classifiers.
Notes	Verifies the Attributes defined in Signal elements. Includes checking if the specified Classifier exists in the current repository.
Rationale	Ensuring the complete specification of Signals is important since they are mapped to Receptions, and also used in Send Signal and Accept Event Actions.

Rule ID	*UnresolvedBlockOperation*
Description	Report unclassified Operations (i.e., with an unresolved return type).
Notes	Verifies the Operations defined in Block and InterfaceBlock elements. By default, it reports Operations where the return type Classifier: - Is null (set the type to 'void' instead, to remove any ambiguity). - No longer exists in the current repository. To check only for the second case, set the *AllowNullReturn* option to True.
Rationale	Verify that the Operation signature is fully specified with regards to the return type.

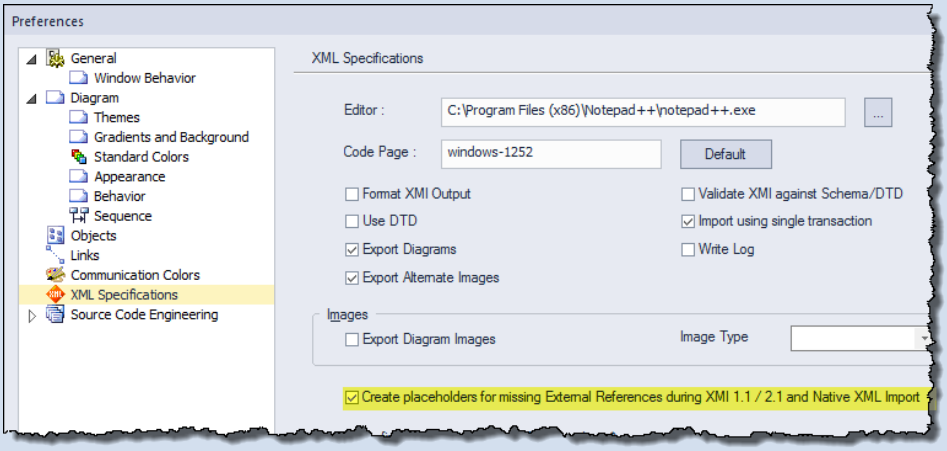
Rule ID	*UnresolvedBlockOperationParameter*
Description	Report Operations with unresolved parameter Classifiers.
Notes	Verifies the Parameters of Operations defined in Block and InterfaceBlock elements. It reports Operations Parameters where the Classifier: - Is null (unassigned). - No longer exists in the current repository.
Rationale	Verify that the Operation signature is fully specified with regards to its input/output arguments (if any).

Rule ID	*UnreferencedBlockOperation*
Description	Report Operations which are not used in a Sequence message, Call Operation Action, Accept Event Action, State Machine behavior (entry/do/exit) or Transition trigger/effect.
Notes	None.
Rationale	Report: 1. Possible gaps in the behavior models of the architecture. 2. Obsolete or deprecated Operations.

Rule ID	*UnmatchedBlockReceptionToSendSignalAction*
Description	Report (Signal) Receptions for which there is no corresponding Send Signal Action present in the model
Notes	The rule looks for Send Signal Actions in the entire model, not just the currently selected context.
Rationale	Report: 1. Possible gaps in the behavior models of the architecture. 2. Obsolete or deprecated Signals.

Rule ID	*UnmatchedBlockReceptionToAcceptSignalActionOrTransitionTrigger*
Description	Report (Signal) Receptions for which there is no corresponding Accept Event Action or State Transition Trigger present in the model
Notes	The rule reports cases where: 1) No corresponding Accept Signal Action can be found. 2) The Accept Signal Action is the child of an Activity (directly or indirectly via either an unclassified Partition or an Interruptible Activity Region) that itself is **not** owned by the Block with the Reception. 3) The Accept Signal Action is a child of a classified Partition that itself is **not** referencing the Block with the Reception. 4) There is **no** SysML Allocate relationship from the Accept Signal Action to the Block. 5) The Accept Signal Action is **not** graphically enclosed in a Partition classified by the Block with the Reception (e.g., the Action is owned by an Interruptible Activity Region, but is graphically located within the boundaries of the Partition). 6) There is **no** State Machine owned by the Block with a Transition Trigger of type Signal matching the Reception.
Rationale	Find possible gaps in the behavior models of the architecture, or obsolete or deprecated Signals.

General (non SysML specific) rules

Rule ID	*UnresolvedConnectorEnds*
Description	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. **Caveat:** the rule can only detect this condition if the *Create Placeholders for Missing External References* property is set! 
Notes	None
Rationale	These missing elements may affect the integrity of the model.

Rule ID	*OrphansWithNoConnectors*
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	By default, Action Pins named ‘target’ or ‘result’, automatically created by EA for simulation purposes, are filtered out. Verifies that the element: • 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.

Internal Block rules

Rule ID	*IncompatibleConnectedPortClassifiers*
Description	Validate connected Ports have compatible Classifiers.
Notes	By default, the validation takes into account the compatibility of: • The direction of Flow Properties (if any). • The direction of Directed Features (if any). • Item Flows (if any) against Flow Properties (if any). There are options in the rule to enable/disable these checks. Comparison between Properties of two Classifiers is based on: 1) A matching Property name for Directed Properties. 2) A matching Property Classifier for Flow Properties and Directed Features. Comparison between Directed Operations of two Classifiers is based on the concatenation of Operation name and return type being the same.
Rationale	Ensure model coherence.

Rule ID	*IncompatibleConnectedPartClassifiers*
Description	Validate connected Parts have compatible Classifiers.
Notes	By default, the validation takes into account the compatibility of: • The direction of Flow Properties (if any). • The direction of Directed Features (if any). • Item Flows (if any) against Flow Properties (if any). Comparison between Properties of two Classifiers is based on: 1) A matching Property name for Directed Properties. 2) A matching Property Classifier for Flow Properties and Directed Features. Comparison between Directed Operations of two Classifiers is based on the concatenation of Operation name and return type being the same.
Rationale	Ensure model coherence.

Rule ID	*MatchingIBDPartToBDDRelationship*
Description	For each pair of connected Parts, verify that there is a corresponding relationship between the two distinct Classifier Blocks.
Notes	The relationship between the Classifier Blocks can be the result of a direct Reference Association or a Part Association, but can also be indirect via a Property typed by the Block on the other side of the Connector. Consequently, the rule is implemented by looking for matching child Parts on either side of the Classifier Blocks. IBD Connectors between Parts are considered bi-directional by default in SysML, so direction is currently not taken into account for this rule.
Rationale	If an IBD shows connectivity between two Blocks (via their Parts), then for consistency, a direct or indirect relationship should exist at the BDD level as well.

Rule ID	*MatchingIBDPortToBDDRelationship*
Description	For each pair of connected **Ports** (of Parts), verify that there is a corresponding relationship between the two distinct Classifier Blocks of the **Parts**.
Notes	See previous rule.
Rationale	If an IBD shows connectivity between two Blocks (via their Ports and Parts), then for consistency, a direct or indirect relationship should exist at the BDD level as well.

Rule ID	*MatchingIBDPartToBehaviorRelationship*
Description	For each pair of connected Parts, verify that there is a corresponding connector between either two Activity diagram Actions (as classified by their Partitions) or two classified Objects of a Sequence diagram.
Notes	Connectivity between Actions is assessed by looking for both Control Flow and Object Flow (via Action Pins) connectors. IBD Connectors between Parts are considered bi-directional by default in SysML, so direction is currently not taken into account for this rule.
Rationale	The IBD level connectivity between two Blocks (via their Parts), documents that these components interact with one another, but not when this happens, and under which conditions. Only an Activity and/or Sequence diagram can provide this context information.

Rule ID	*MatchingIBDPortToBehaviorRelationship*
Description	For each pair of connected Ports (of Parts), verify that there is a corresponding connector between either two Activity diagram Actions (as classified by their Partitions) or two classified Objects of a Sequence diagram.
Notes	Connectivity between Actions is assessed by looking for both Control Flow and Object Flow (via Action Pins) connectors. For Sequence diagrams, both Sequence-to-Sequence and Port-to-Port connectivity is checked. IBD Connectors between Parts are considered bi-directional by default in SysML, so direction is currently not taken into account for this rule.
Rationale	The IBD level connectivity between two Blocks (via their Ports and Parts), documents that these components interact with one another, but not when this happens, and under which conditions. Only an Activity and/or Sequence diagram can provide this context information.

Requirement rules

Rule ID	*UnsatisfiedRequirements*
Description	Verify Requirements are satisfied/realized using a connector type that is SysML compliant.
Notes	By default, the rule specifically checks for the SysML *Satisfy* relationship targeting the Requirement. The source of the relationship can be any element.
Rationale	Every system Requirement should have a relationship with one or more elements to document how it is being met by the architecture. Realization of end User Requirements sometimes occurs not from these Requirements directly, but from derived (lower level) Requirements. In that case, add the string *deriveReqt* to the Connector stereotype filter list.

Rule ID	*UntestedRequirements*
Description	Verify Requirements are verified/tested using a connector type that is SysML compliant.
Notes	By default, the rule: • Specifically checks for the SysML *Verify* relationship targeting the Requirement • Excepts the source of the relationship to be a SysML Test Case element.
Rationale	Every system Requirement should have a relationship with one or more elements to document how it is being verified by the architecture. Testing of end User Requirements sometimes occurs not from these Requirements directly, but from derived (lower level) Requirements. In that case, add the string *deriveReqt* to the Connector stereotype filter list.

Sequence Interaction rules

Rule ID	*SequenceMessageNoOperation*
Description	Report Sequence messages with no associated Operation.
Notes	An option is provided to filter out Sequence messages: • Targeting one or more element types. • Targeting one or more element stereotypes.
Rationale	Sequence Messages should refer to actual Operations of the Block, Interface Block, or Port classifying the Sequence, and not just represent made up text strings.

Rule ID	*SequenceMessageMissingOperation*
Description	Report Sequence messages with an unresolved Operation reference (e.g., the Operation is missing from the model).
Notes	An option is provided to filter out Sequence messages: • 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	*SequenceWithNoClassifier*
Description	Verify Sequence elements (called Lifelines in UML) have a Classifier defined.
Notes	None.
Rationale	Sequences should have a Classifier defined, and not refer to anonymous Objects.

Use Case rules

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

Rule ID	*ActorToUseCaseConnectors*
Description	Verify Actor to Use Case connector types are SysML compliant.
Notes	By default, checks if the connector type is *Association*.
Rationale	As per the UML/SysML specification.

Rule ID	*ActorWithNoDescription*
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 Actor, and not just to rely on the name which can be a source of ambiguity.

Rule ID	*UseCaseWithNoDescription*
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	It is recommended to provide a minimum description of the context in which the Use Case is set to execute, even if it is supported in the model by a Structured specification or Activity diagram.

Rule ID	*DuplicateActorName*
Description	Report duplicate Actor elements.
Notes	None.
Rationale	Actors should have a unique name and be defined one time in the repository (typically in a library) and reused wherever needed.

Rule ID	*MissingUseCaseToRequirement*
Description	Report Use Cases with no SysML compliant relationship to one or more Requirements.
Notes	By default, checks for the SysML r*efine* relationship from the Use Case to the Requirement. Type and Stereotype filters are available at the Use Case and Requirement level.
Rationale	Use Cases should relate back to the functional Requirements that they help document.

Rule ID	*UseCaseWithNoConstraints*
Description	Report Use Cases with no pre- and post-conditions defined.
Notes	Type and Stereotype filters are available at the Use Case and Constraint level. By default, the rule checks for both pre- and post-conditions to be present.
Rationale	A well-defined Use Case states: • The conditions whereby the Use Case may not 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. Constraints are particularly useful for defining black box test cases.

State Machine rules

Rule ID	*SignalTransitionToSendSignal*
Description	Verify Transitions with Signal Triggers have a corresponding Send Signal Action defined in the model.
Notes	The Send Signal Action can be defined anywhere in the model.
Rationale	Without a Send Signal Action, the Trigger would never fire.

Rule ID	*TransitionTriggerSpecified*
Description	Verify that any defined Trigger for a Transition has a specification value.
Notes	The rule verifies that the Trigger: 1) Has a type defined (either Call, Signal, Change, or Timer). 2) For a Call, that there is an Operation specified for the Trigger, and the Operation reference is not unresolved. 3) For a Timer, that there is non-empty text specification defined. 4) For a Change, that there is non-empty text specification defined. 5) 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 fully document the event that will cause the Trigger to fire.

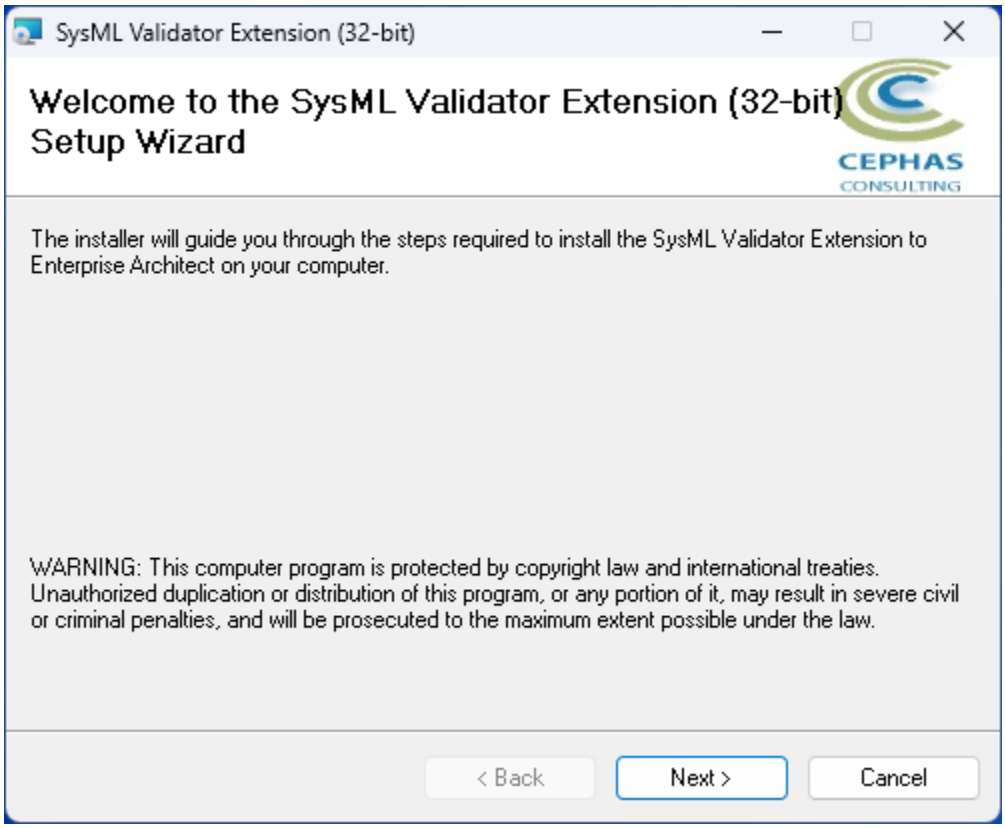
Rule ID	*SignalTransitionWithNoMatchingBlockReception*
Description	Verify Transitions with a Signal Trigger have a matching Reception defined in the Block owning the State Machine.
Notes	An error is reported if the State Machine is not the child of a Block.
Rationale	Without a Reception defined in the Block owning the State Machine, the Trigger would never fire.

Rule ID	*AssignedBehaviors*
Description	Check for States with entry/do/exit behaviors that do not reference specified model behaviors.
Notes	The rule verifies that for each declared behavior (entry, do, or exit) of a State, the *Behavior* and the *Code* fields of the (pseudo) Operation are not both null. 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. If a reference to a model behavior is specified, the rule verifies that the reference can be resolved in the model.
Rationale	A State behavior should have some implementation information defined, beyond just its name.

Installation

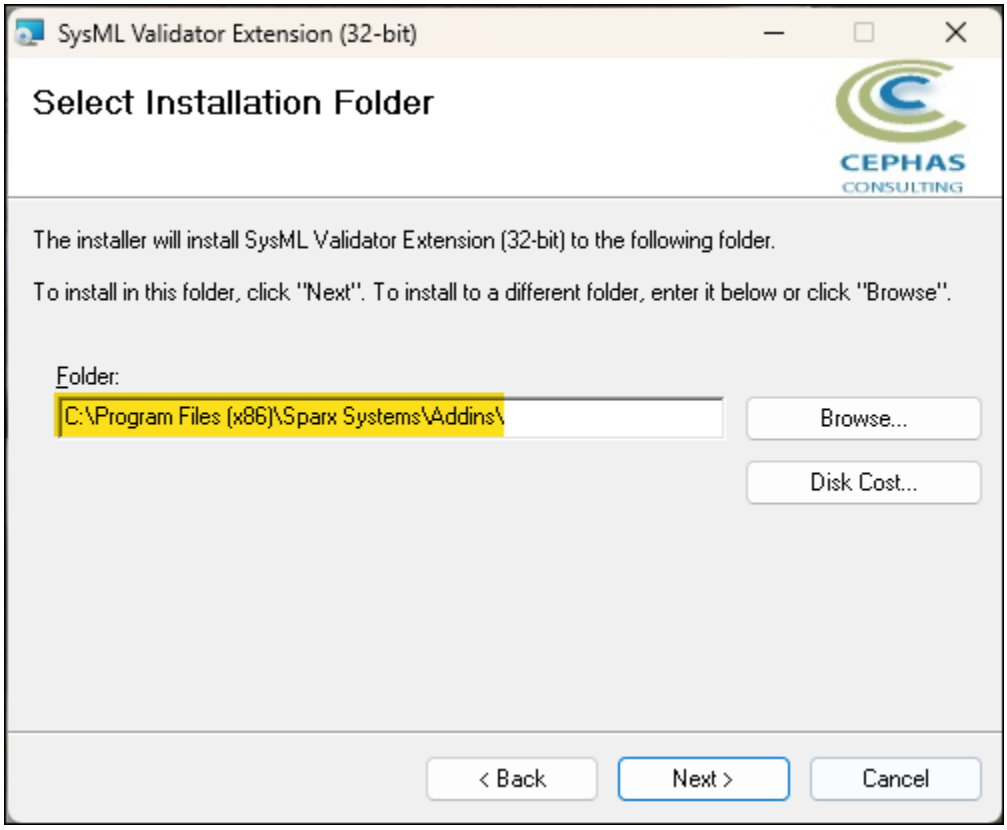
The installation process is the same for both the trial and the full version.

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



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 32-bit version:

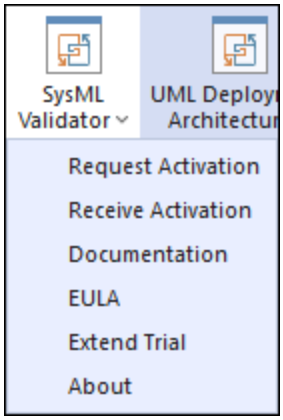


Note that while older versions of the software can be upgraded, it is recommended to uninstall and reinstall the extension.

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.

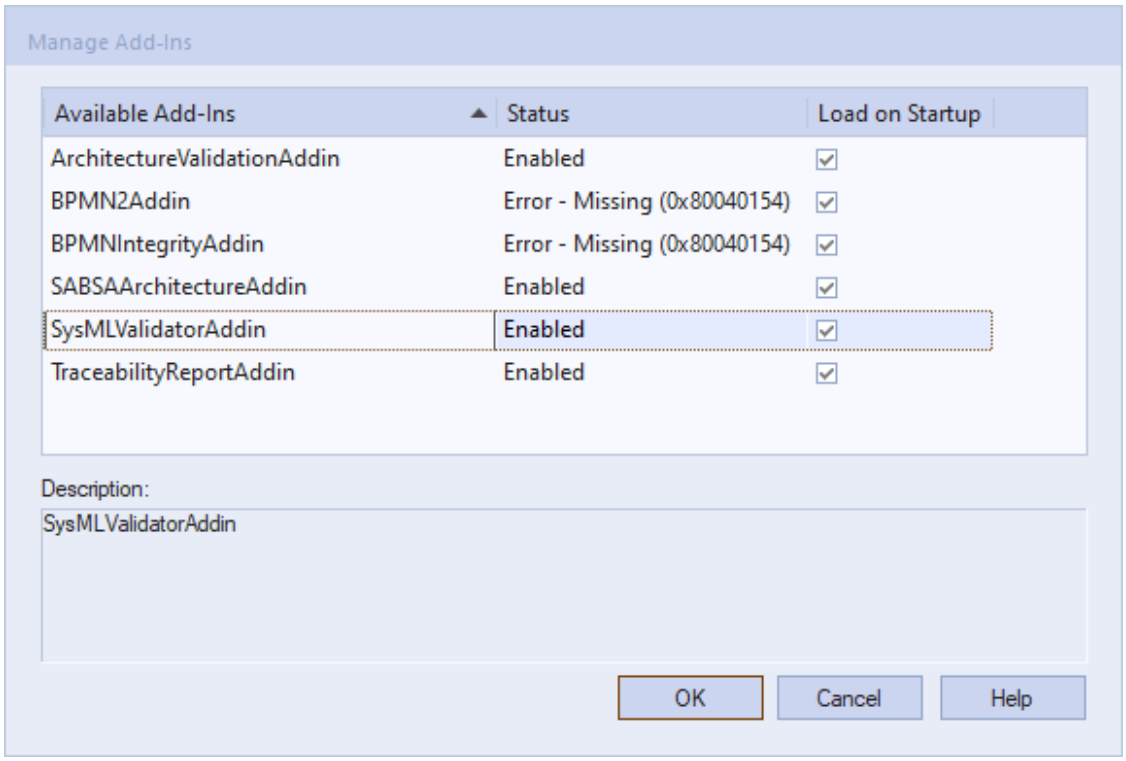
Verifying the installation

Bring up Enterprise Architect, without necessarily opening a repository, and verify that the *SysML Validator* 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 *SysML Validator* extension is present and enabled:



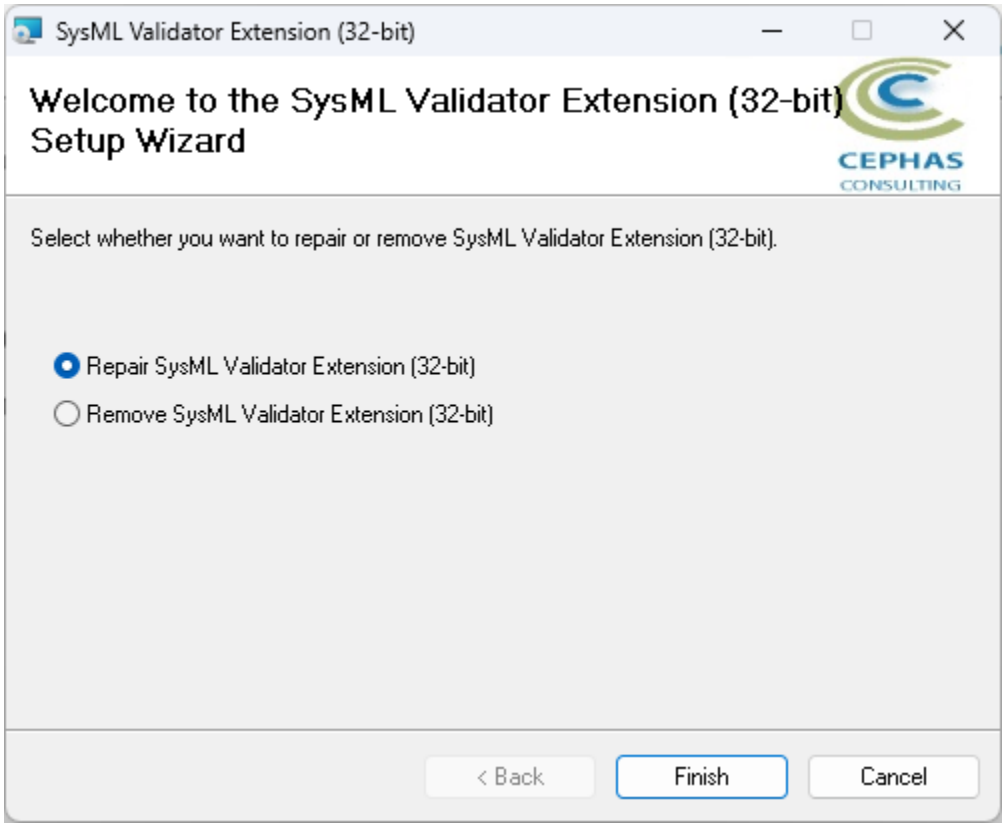
If an error status is shown, as in the example above for other extensions, this typically means that either:

- 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 *SysML Validator* entry itself is not found, then the extension installation did not complete successfully.

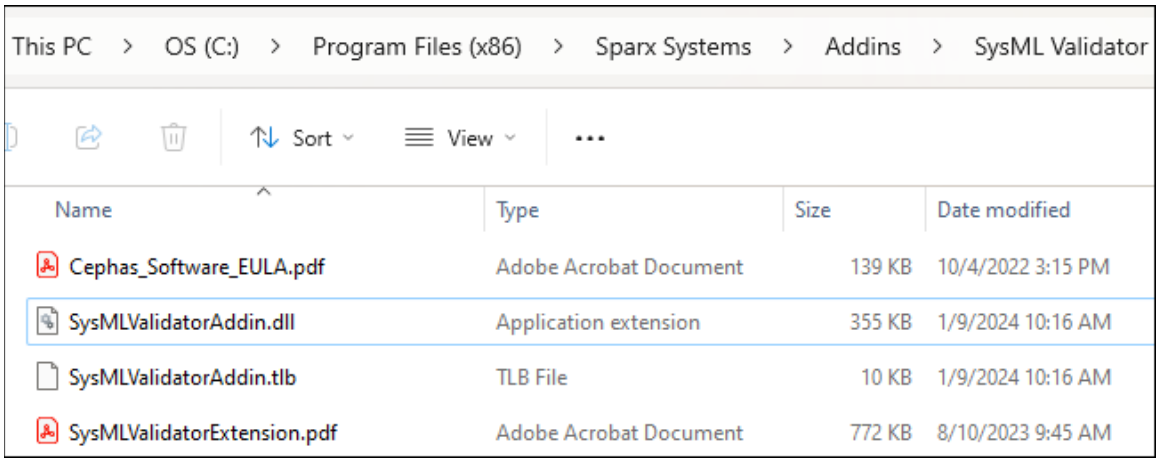
To fix an incorrect installation:

- 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 *SysML Validator* extension is still not loaded correctly in Enterprise Architect, remove the program through the Windows control panel and start the installation process over.

At the completion of a successful installation the following files are installed in the selected directory:



Licensing

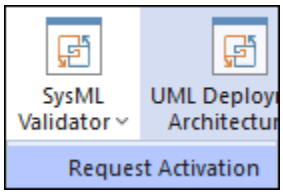
Trial version

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

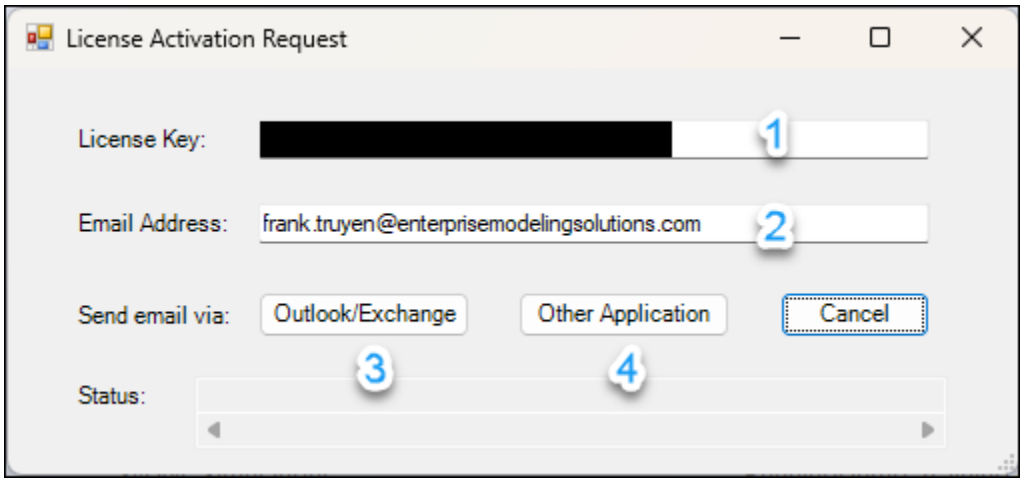
Licensed version

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:

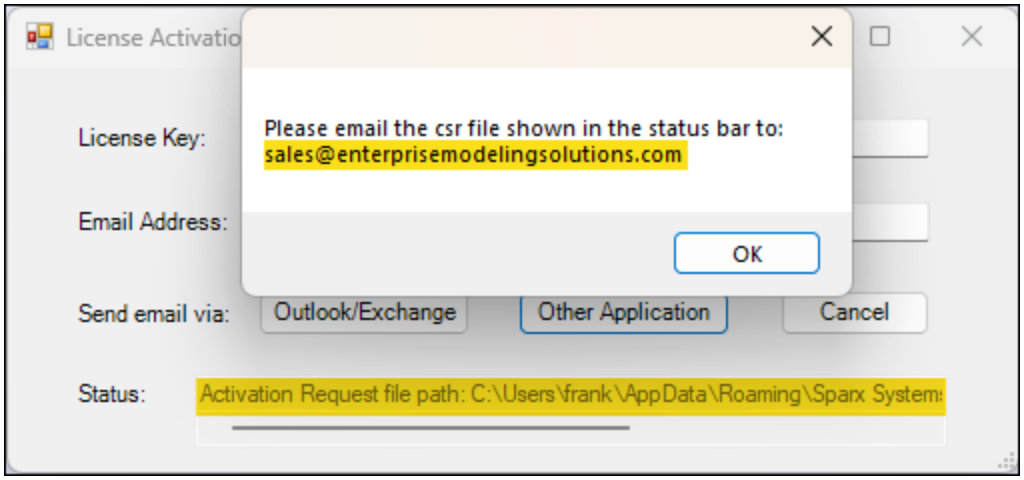
Request the license activation



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:

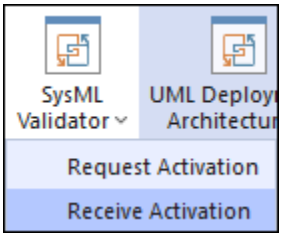
sales@enterprisemodelingsolutions.com

Receive the license activation

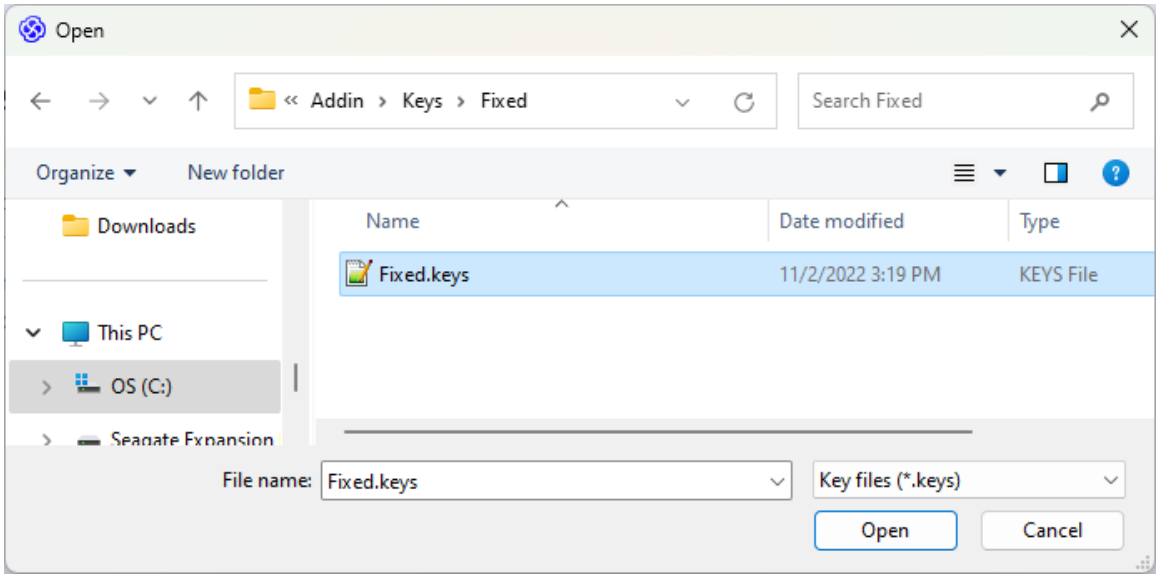
Upon receiving your activation request, Cephas Consulting replies with a license key file named “Fixed.keys”.

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

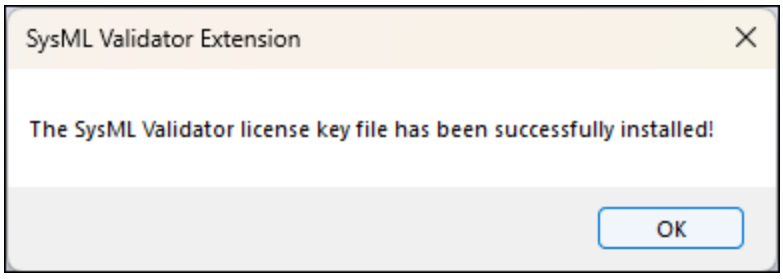
To install the license key file, open Enterprise Architect and in the *Specialize → Add-Ins* ribbon panel, select:



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



A confirmation of the license activation is then displayed:



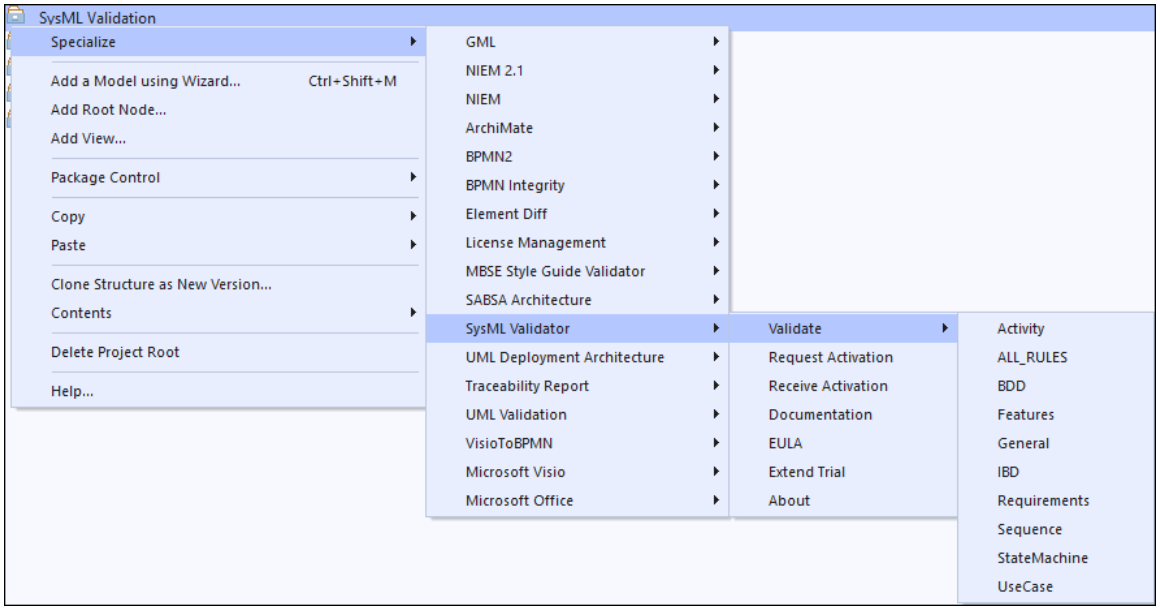
Next **exit the Enterprise Architect tool** and **restart it** in order to fully enable the extension.

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).

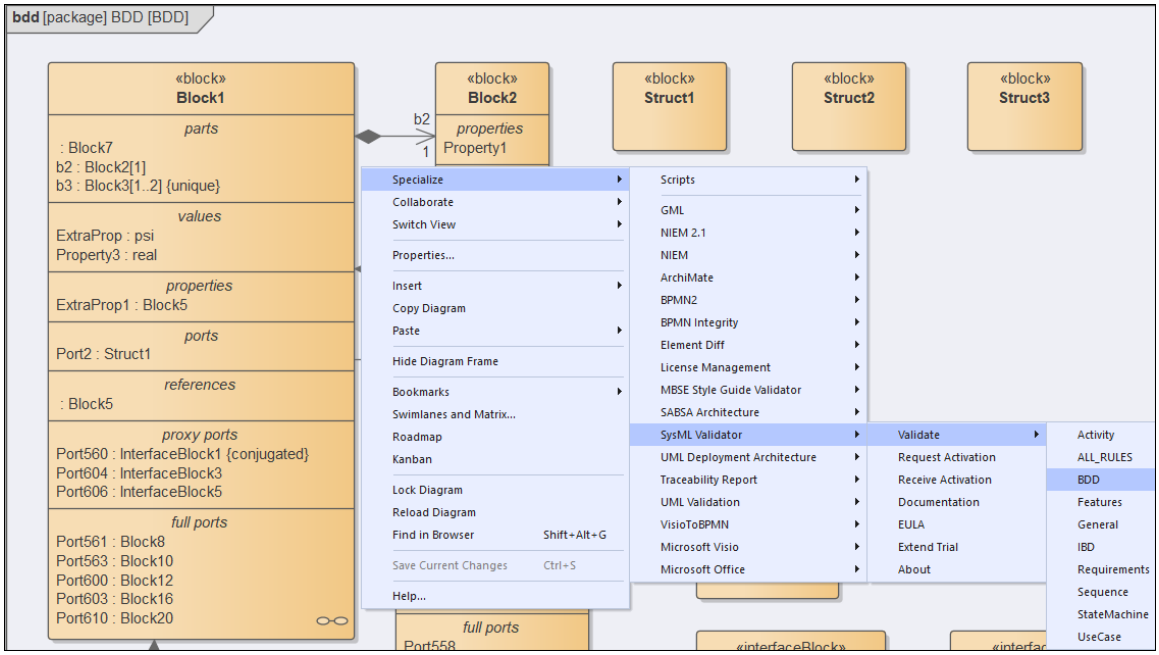
Right click a Package, or the top of a Package hierarchy, and select any of the available rule sets:



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

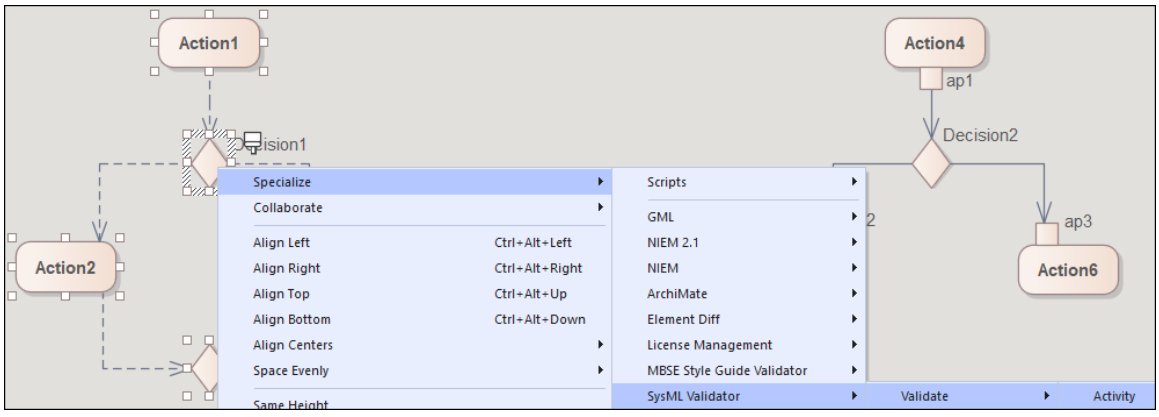
Diagram level validation

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



Element level validation

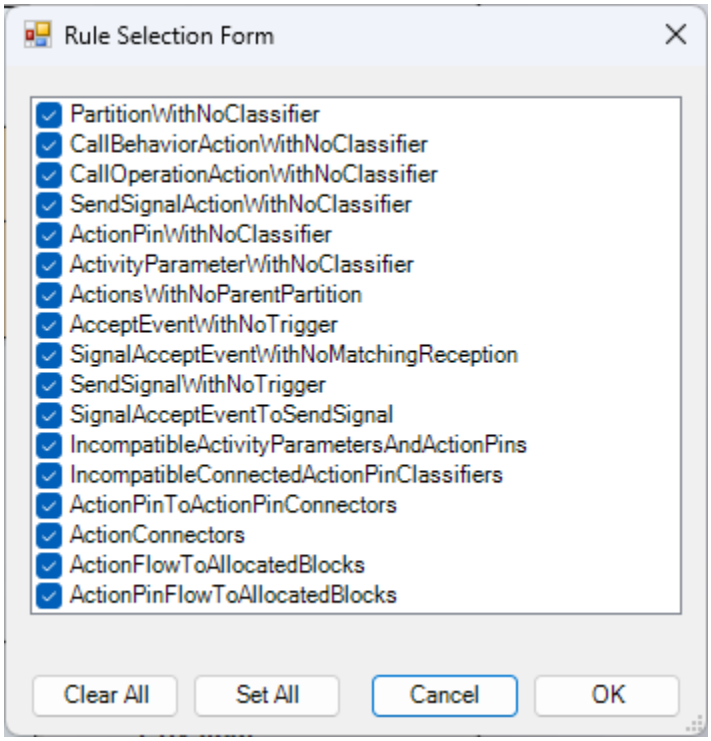
Select one or more diagram elements, then **right click** and select the rule set appropriate for this type of selection. For example, in the context of a SysML Activity diagram:



If the set contains rules for validating connectors, ensure that both the source and the target elements are part of your selection.

User Interface

Unless the *RunSilent* option is enabled, a form displays when launching the validation for a specific rule set (except for “ALL_RULES”). For example, for Activity:



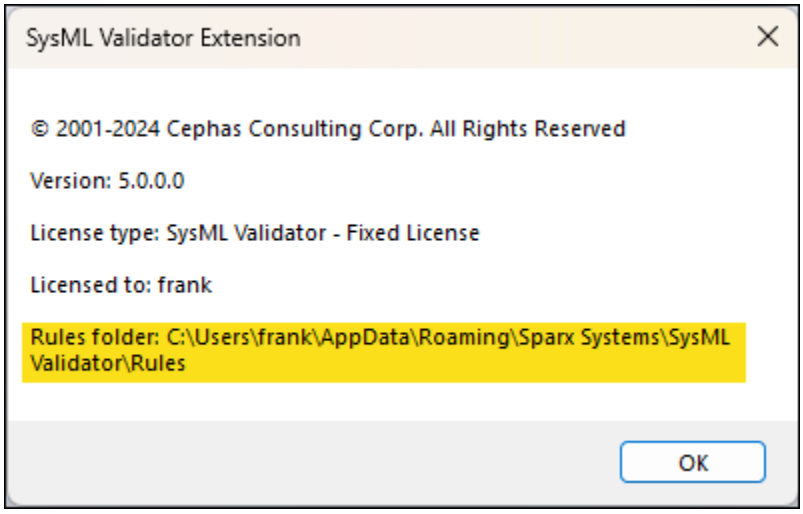
This allows rules to be enabled/disabled for a specific validation. To disable rules by default, create a custom rule set (see next section).

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 new file in any XML editor and make the required changes in that copy.

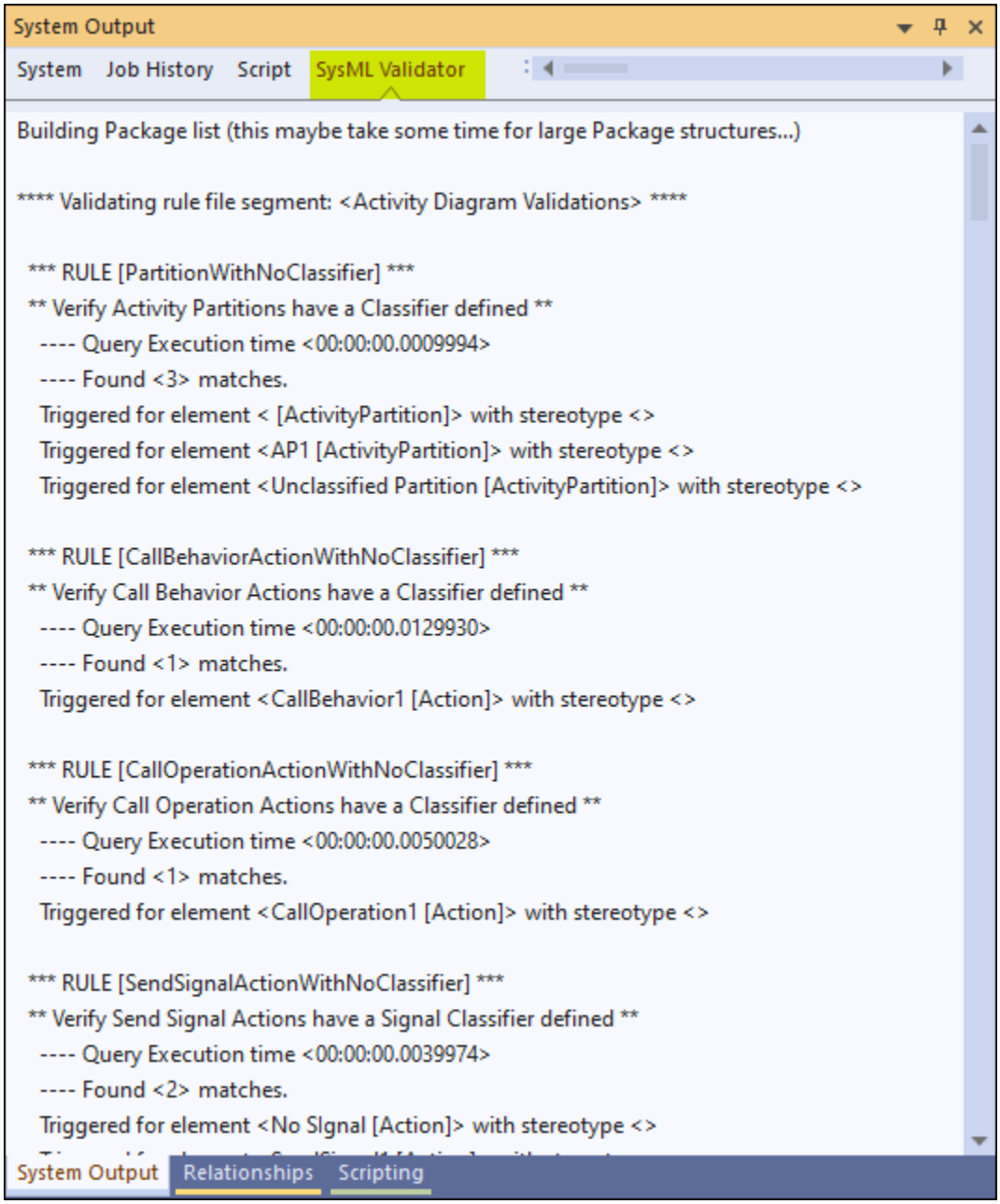
When adding a new rule to a set, ensure that it is given a unique ID (i.e., name) value.

Custom rule files are automatically detected and made available for selection.

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

Verifying the result set

During execution, the “SysML Validator” tab in the System Output window automatically opens and displays the results of the validation. For example:



Every rule match listed in this window uses 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.

Active output lines are the ones that are NOT:

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

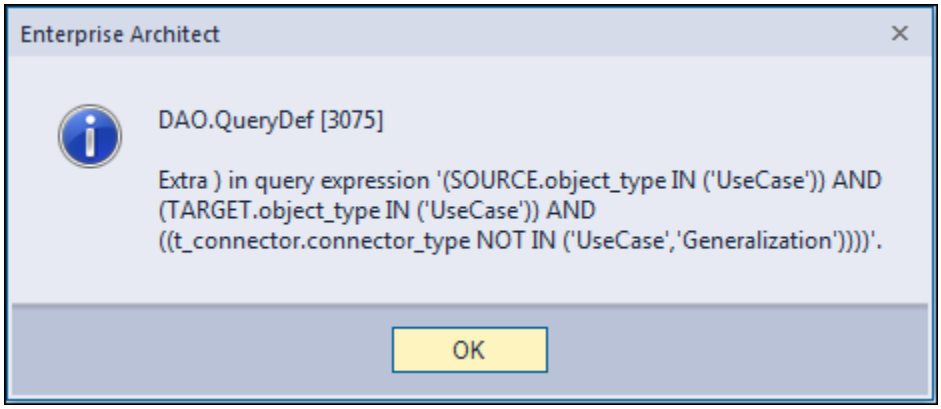
Single click an active output line to automatically locate its related element, Attribute, Operation or diagram in the Browser.

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, 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 to report any error encountered:

- 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 error notification, look at the System Output window to determine the rule being executed at the time of the failure. For example:

  *** VALIDATING RULE [UseCaseToUseCaseConnectors] on connector type <'UseCase','Generalization'> ***

- 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 SysML Validator 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.