# Activity simulation engine

**On this page**



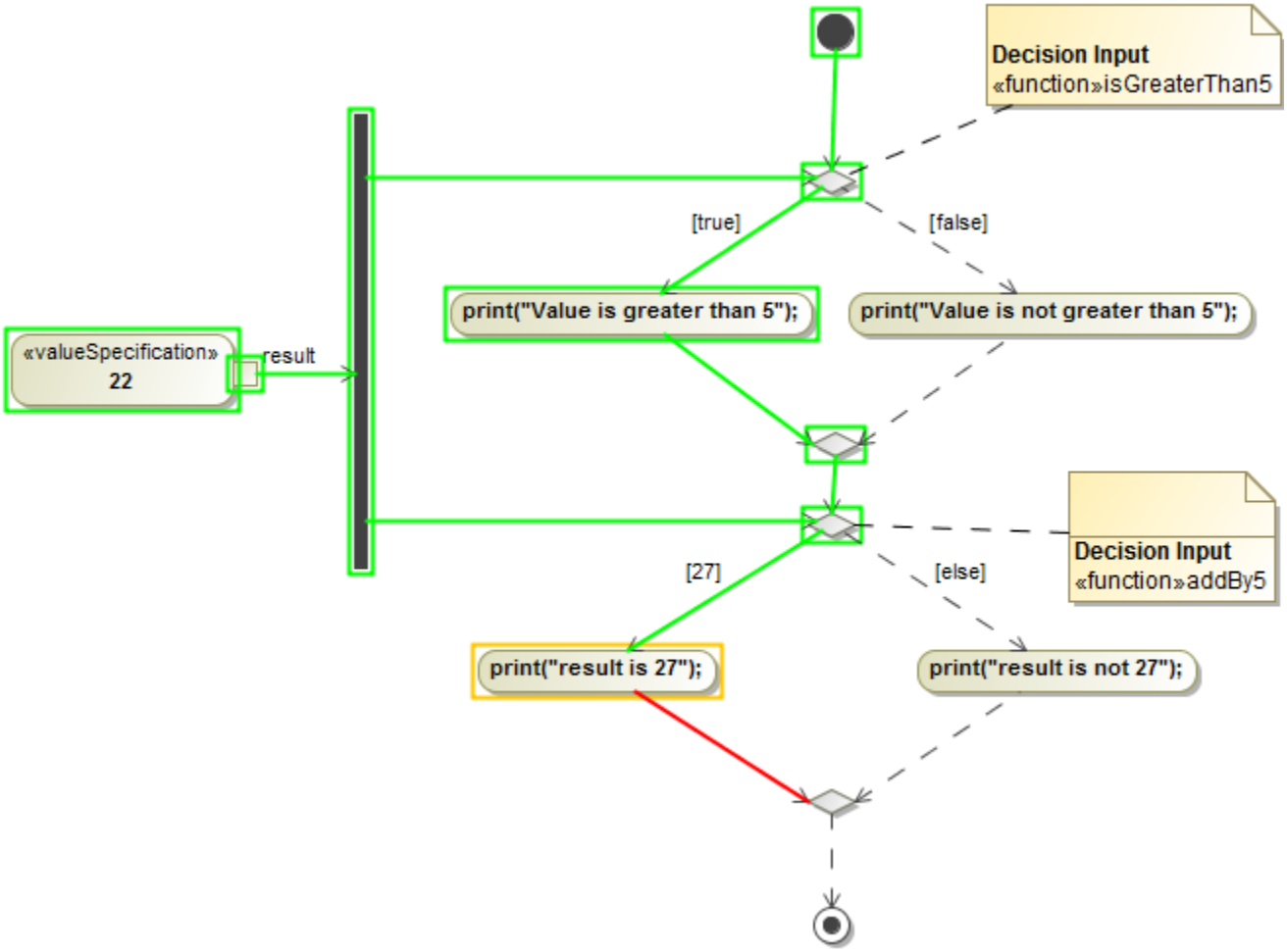
Cameo Simulation Toolkit provides an Activity simulation engine that allows you to run an Activity Simulation on Activity diagrams or Activity Elements. Cameo Simulation Toolkit also includes the implementation of OMG Semantics of a Foundational Subset for Executable UML Models (fUML), an executable subset of standard UML, that can be used to define the structural and Behavioral semantics of systems. fUML defines a basic virtual machine for the Unified Modeling Language and supports specific abstractions enabling compliant models to be transformed into various executable forms for verification, integration, and deployment.

Various UML Activity diagram concepts are supported, including Object and Control Flows, Behavior and Operation Calls, sending Signals via Connectors with or without Ports in Internal Structure, accepting Signals and Time Events, Pins, Parameters, Decisions, Structured Activity Nodes, and many more.

The Activity simulation engine features include the following

- fUML 1.3 specification support.
- Any Action languages in opaqueBehaviors, opaqueExpressions, Decisions, Guards, and Constraints (see Integration with MATLAB® for more details).
- CallBehaviorAction with nested diagrams simulation and animation.
- SendSignalAction to send a Signal to a global Event queue to be used by the system level but not by the subsystem, e.g., in State Machine.

  **Note**
  - SendSignalAction has the **Target** input Pin for specifying the signal receiver. If this Pin is omitted, the signal instance will be sent to the context itself.
  - The **On Port** property on SendSignalAction is used for sending signals via the specified port.
- CallOperationAction through a Port.
- Sending Signals through a Port.
- Support for Decision Nodes with probabilities over all outgoing edges.
- Support for Decision Nodes with a Decision input that provides input to Guard specifications on outgoing edges from each Decision Node.



The supported Decision input for Decision nodes.

> **Note**
>
> - You can simulate only Activities that are owned by a Package or a Class. As a workaround, the CallBehavior Actions, owned by the Call Behaviors in a Package, will be used for the entry/do/exit Behaviors in States.

Most of the elements on an Activity diagram are supported as follows.

- Control Flow
- Object Flow
- Input Pin (Illegal type is not supported)
- Output Pin
- Activity Final Node
- Flow Final Node
- Activity Parameter Node: The parameter of the Activity Parameter Node must be defined. If the Activity Parameter Node violates the rule, Simulation will print a warning message in the **Console** pane.
- Decision Node
- Merge Node
- Join Node
- Fork Node
- Structured Activity Node
- Conditional Node
- Loop Node: the setup part will not be executed as the fUML specification. The number of Input Pins of the Loop Node must be equal to Output Pins. If the Loop Node violates the rule, Simulation will print an error message in the **Console** pane.
- Expansion Region
- Expansion Node
- Object Node
  - Central Buffer Node
  - Data Store Node
- Actions
  - AcceptEventAction
  - AddStructuralFeatureValueAction
  - CallBehaviorAction
  - CallOperationAction
  - ClearAssociationAction
  - ClearStructuralFeatureAction
  - CreateLinkAction
  - CreateObjectAction
  - DestroyLinkAction
  - DestroyObjectAction
  - OpaqueAction
  - ReadExtentAction
  - ReadIsClassifiedObjectAction
  - ReadLinkAction
  - ReadSelfAction
  - ReadStructuralFeatureAction
  - ReclassifyObjectAction
  - ReduceAction
  - RemoveStructuralFeatureValueAction
  - SendSignalAction
  - StartObjectBehaviorAction
  - TestIdentityAction
  - ValueSpecificationAction

- The Guards on an Object flow are not Boolean expressions in fUML. They should contain a value that matches the runtime value that flows on the Object Flow during simulation.
- A decision input can be passed as a parameter of Behavior of an opaque expression through a pin of Action. If a valid return value is from an assigned Behavior of a Guard, the body of the Guard will be ignored. However, if the Behavior is <unspecified>, the body of the Guard will be used.

## fUML engine settings

You can customize Behaviors of the fUML engine for Simulation to match your project Requirements through the **Project Options** dialog as shown below.

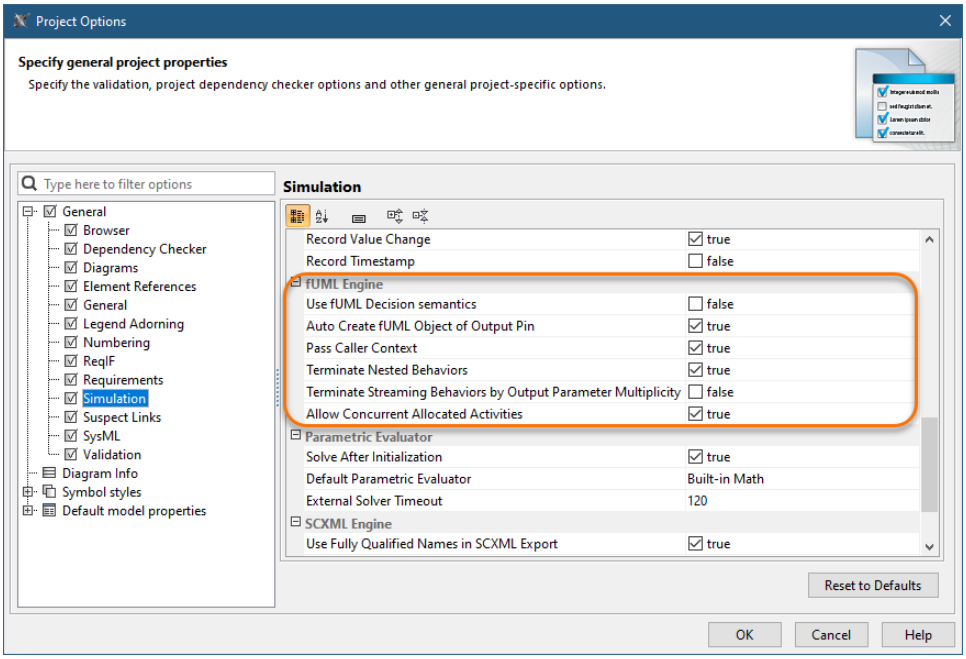
To customize the fUML engine settings in the **Project options** dialog

1. On the main menu, click **Options** and select **Project**. The **Project options** dialog opens.
2. On the left pane, click **General** > **Simulation**.

To change to a regular UML (Boolean expression)

1. On the main menu, click **Options** > **Project** and select **Simulation** on the left of the **Project Options** dialog.
2. Select the **Use fUML DecisionSemantics value** check box so that the value becomes false. The value is false by default in the UML mode.

3. Go to the **fUML Engine** group and set any of the properties as desired.




The following table describes the project options of the fUML engine.

| Property | Function |
|---|---|
| **Use fUML Decision semantics** | If set to *true* (false by default when in the UML mode), all Guards will be solved to true when the flowing token value matches the Guard value (instead of evaluating every Guard as Boolean operation). |
| **Auto Create fUML Object of Output** | If set to *true*, automatically create an fUML object of output. |
| **Pass Caller Context** | If set to *true*, pass the caller context to the called Behavior if it does not have its own context specified. Otherwise, use the Behavior itself as context. |
| **Terminate Nested Behaviors** | If set to *true*, terminate nested Part Behaviors if their parent object Behavior is terminated. |
| **Terminate Streaming Behaviors by Output Parameter Multiplicity** | If set to *true*, a streaming Activity terminates when each of its output parameters receives a cumulative number of values equal to the upper bound of the parameter multiplicity. If set to *false*, a streaming Activity terminates only when forced by the Activity final node or termination of the Activity that invoked it. |
| **Allow Concurrent Allocated Activities** | If set to *true*, Activities can be executed in parallel even if the allocated resource is busy. If set to *false*, only one allocated Activity is executed for an object represented in an Activity Partition.<br>If the **Allow Concurrent Alllocated Activities** option is set to *true*, it is recommended to use a time event to control the execution of parallel activities. A time event will increase the simulation performance in case one parallelly executed activity needs to get a response from another in order to start. |

**Note**
You can instantiate a nested composite structure using the standard PSCS construction mechanism by default.

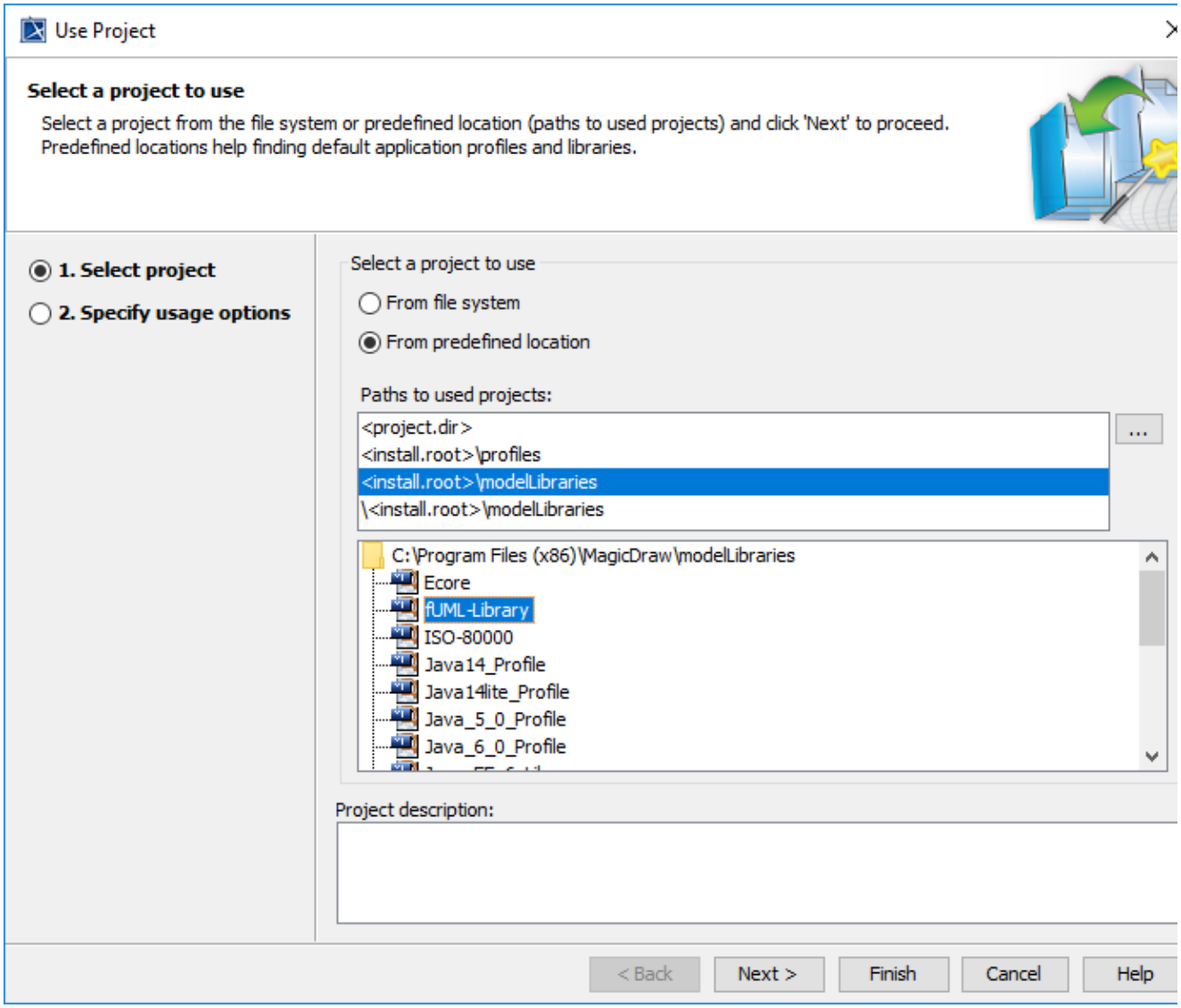
## ReadLine support

ReadLine is a function that allows the user to enter value through the input line on the **Console** pane. A Call Behavior Action can be set Behavior as **Read Line [fUML_Library::BasicInputOutput]** using **fUML_Library.mdzip** from the **Use Project** dialog. Before using the ReadLine function, you need to include **fUML_Library.mdzip** in the project first.

To open the **Use Project** dialog and include **fUML_Library.mdzip**

---

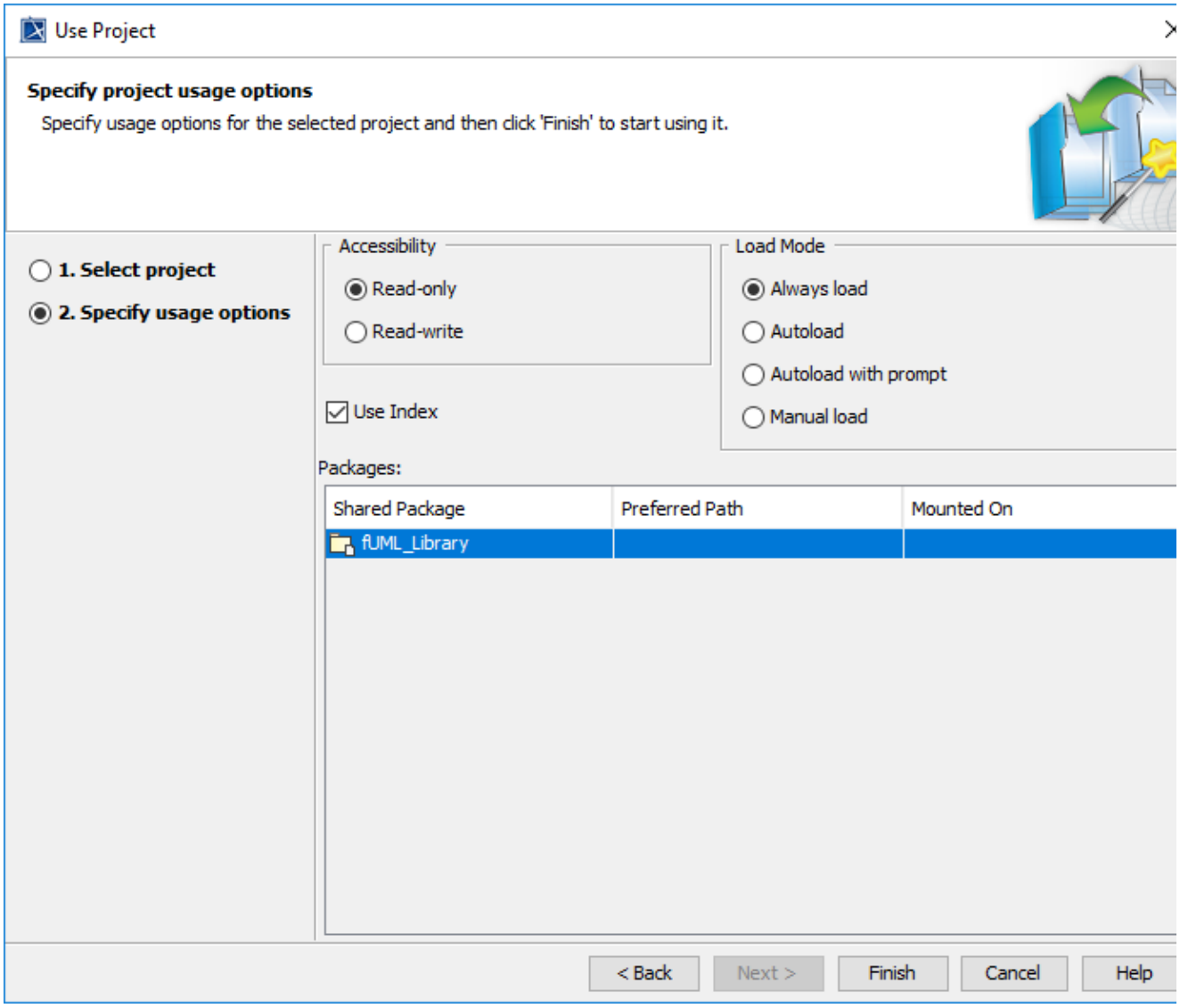
1. Click **File** > **Use Project** > **Use Local Project** from the main menu to open the **Use Project** dialog.



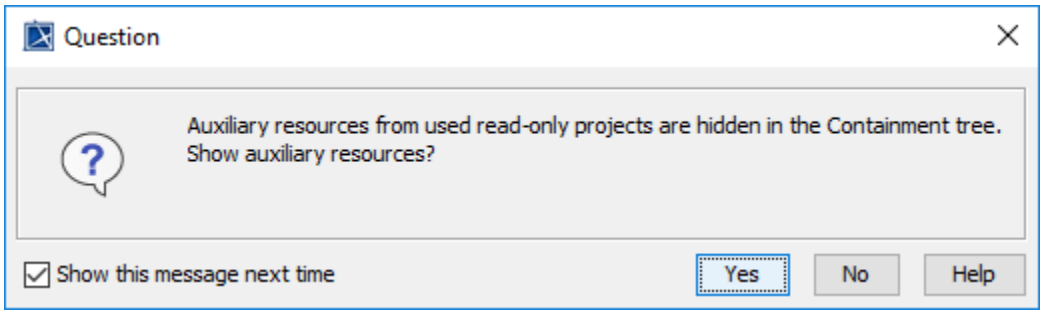

2. In the **Select a project to use** area, select the **From predefined location** option.

3. In the **Paths to used projects** list, select **<install.root>\modelLibraries**.

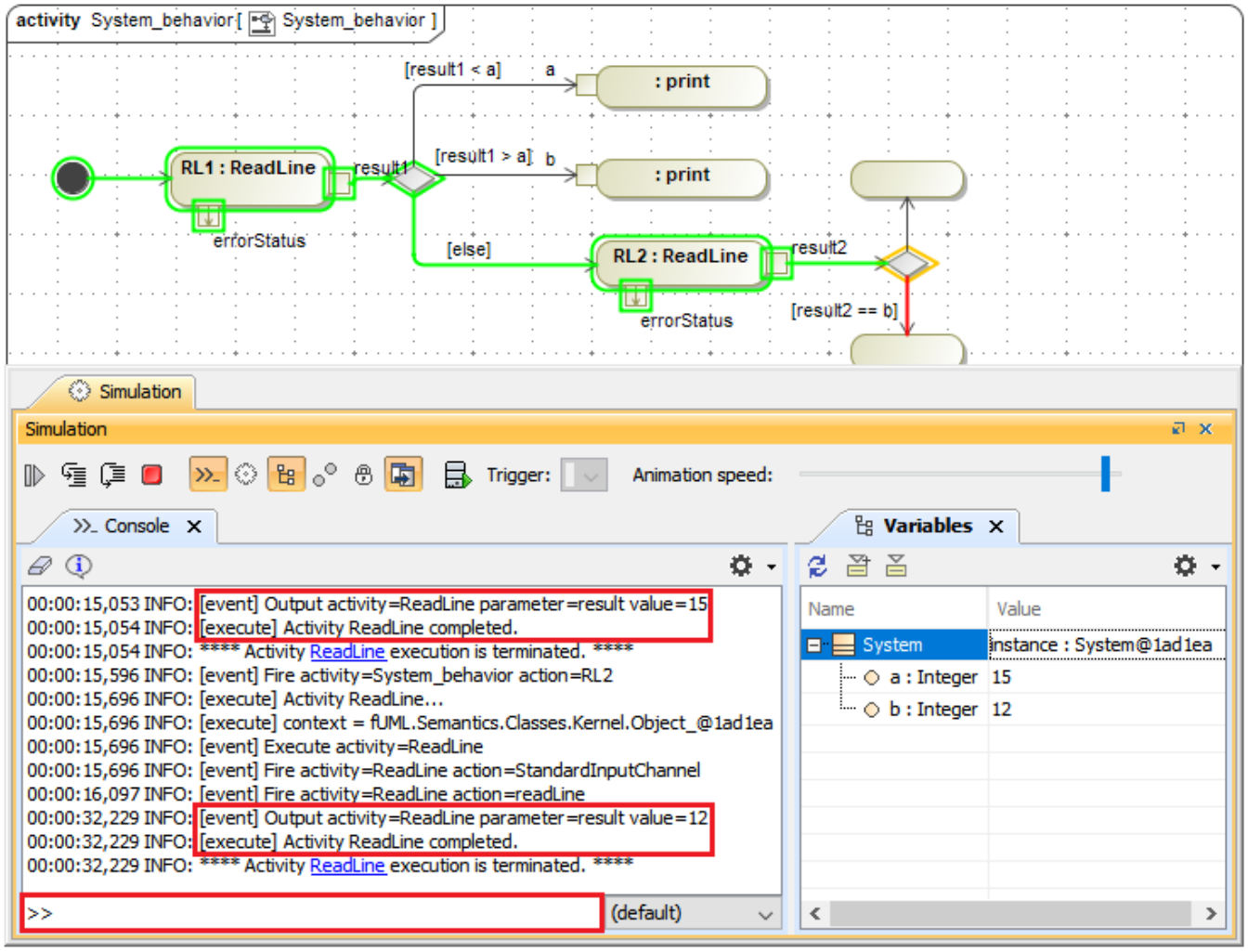
4. In the directory tree list, select **fUML-Library** and click **Next>** to proceed to the next step.




5. You will be at the **Specify usage options** step. Click **Finish**. The **Question** dialog opens to ask you about showing auxiliary resources in the Containment tree. Click **Yes**.



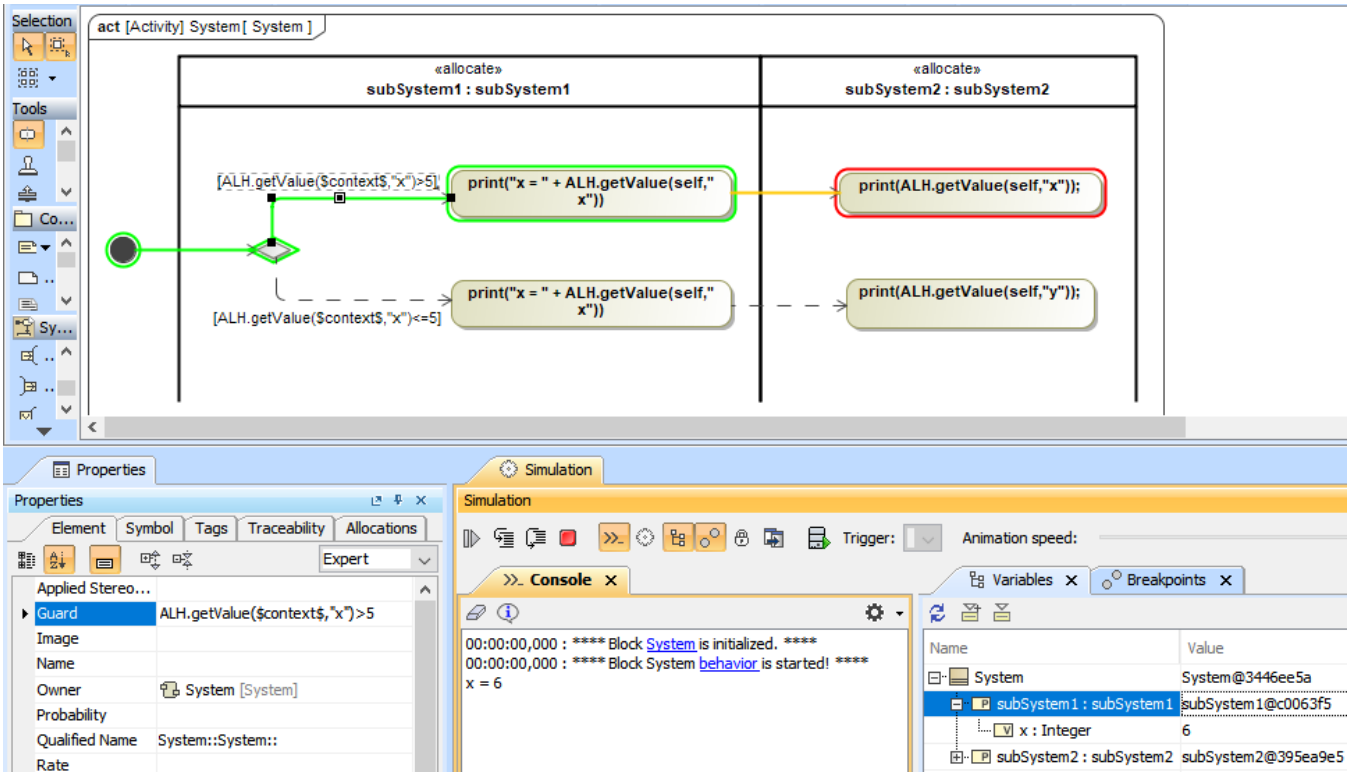

Then a Call Behavior Action can now be set Behavior as a ReadLine Element. The ReadLine Element will be shown with two default Pins, i.e., result and errorStatus. During the simulation, the ReadLine Element is executed to allow entering value through the input line on the **Console** pane. The result of the ReadLine Element can be used by other Elements with any proper data types, e.g., Guard, as in the following figure




ReadLine support allows entering values through the input line on the Console pane.

## Guards in Swimlanes

Simulation supports Guards in Swimlanes in the Activity diagram as shown in the figure below. See also Using Guards on Transitions and Swimlane.




Guards in Swimlanes are supported in the Activity diagram.

**Related pages**

- Decision and Merge
- Behavior
- Action