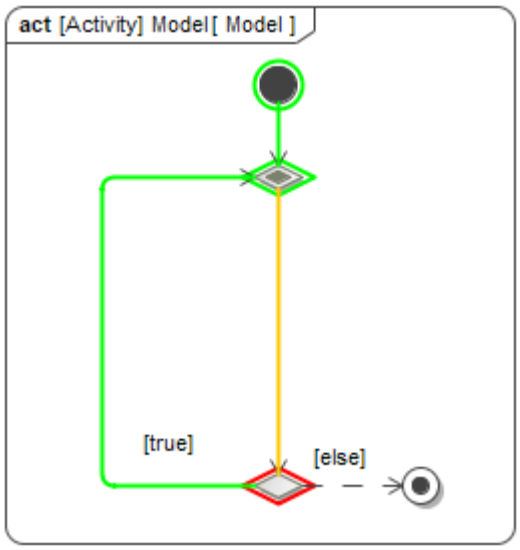
Simulation modeling: Do's and Don't's

On this page

Don’t: Create an fUML loop without any Action Activation

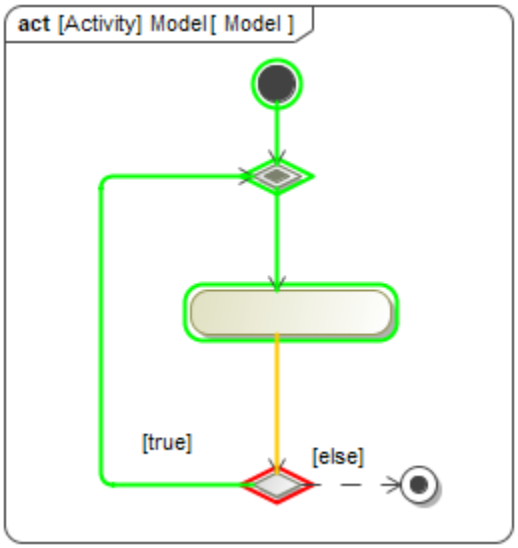
When the fUML model has a loop without any Action Activation in the loop as shown in the figure below, the code stack continually increases until **StackOverflowError** occurs, causing the execution to be unexpectedly terminated. You can see **StackOverflowError** in the **magicdraw.log** file.



The model runs an infinite loop until StackOverflowError occurs, and the execution is terminated.

Do: Add an Action Activation in the loop

To avoid **StackOverflowError**, you must add at least an Action Activation, e.g., **CallBehaviorAction** in the loop as shown in the figure below. The code stack will not increase continually without **StackOverflowError** because Simulation has code to cut the stack loop and recall the Action at **ActivityNodeActivation.receiveOffer()** and **ActionActivation.fire()** according to fUML v1.3 specification. The execution will be continuously run without unexpected termination.

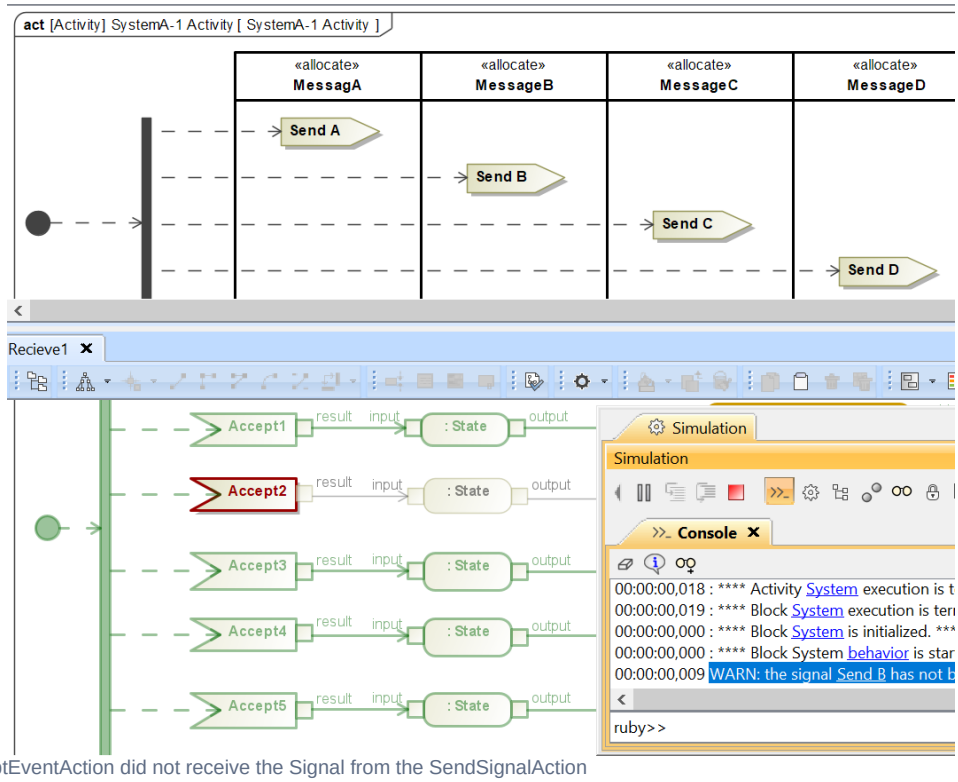


The model runs an infinite loop without StackOverflowError, and the execution is not terminated.

Don't: Send multiple signals and multiple AcceptEventActions at the same time without synchronization at all

When the fUML model has multiple threads of sending signal and multiple threads of PREPARING TO READ the signal at the same time, with no synchronization at all, it may send a signal before other thread starts AcceptEventAction to listen for it in the event pool.

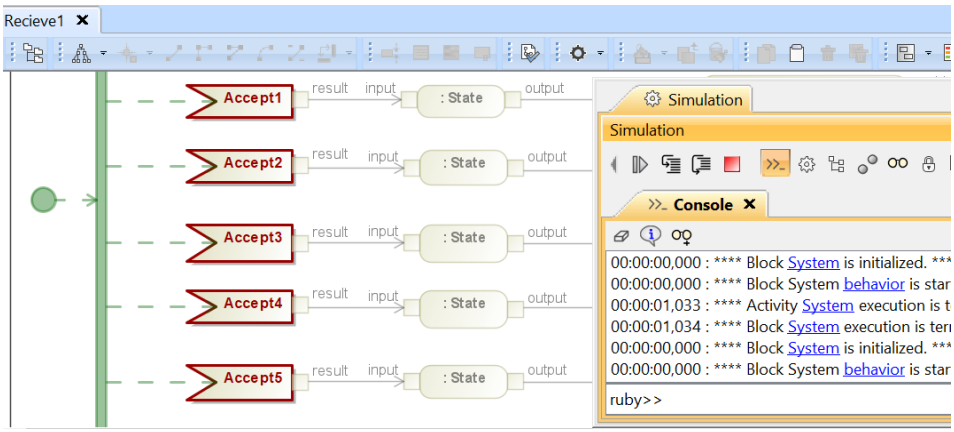
According to fUML semantics, an event is lost and removed if such is a case: According to fUML semantics, once an event occurrence is selected for dispatch, it is matched against the list of waiting event accepters for the active object. If a match is found, the event occurrence is passed to the event accepter using its accept operation. If no matching event acceptor is found, the event occurrence is not returned to the event pool and is lost.



AcceptEventAction did not receive the Signal from the SendSignalAction

Do: Send signals only when all AcceptEventActions are activated

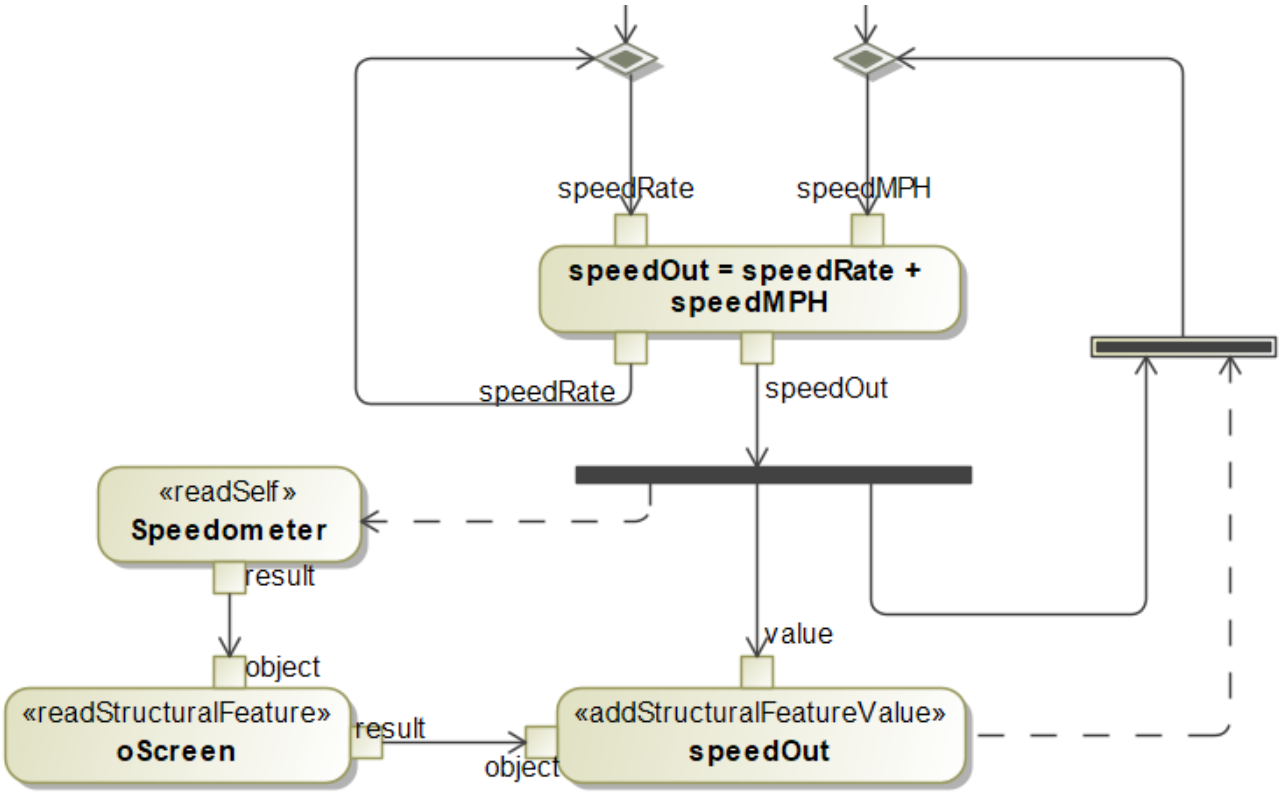
To avoid off-synchronization, you must synchronize sending and receiving activities, to make sure that you start sending only when all AcceptEventActions are activated. For example, you may send "Ready" signals after they are all activated to the sender.



All AcceptEventActions are activated for receiving the Signal from the SendSignalAction

Don't: Create fUML loops with Fork nodes

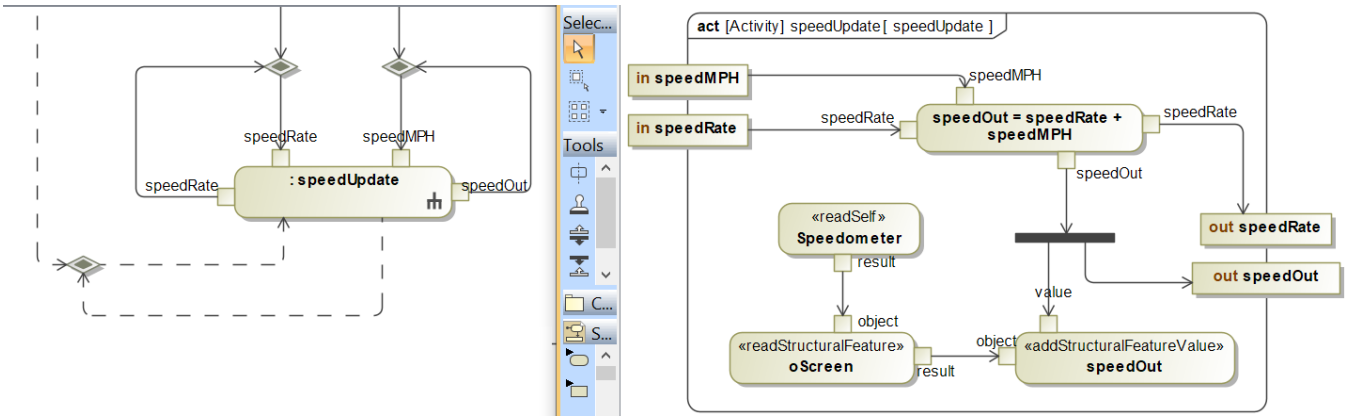
When the fUML model has loops with Fork nodes as shown in the figure below, execution threads continually increase until the internal error occurs, causing the execution to be unexpectedly terminated.



The model has loops with Fork nodes

Do: Use a CallBehaviorAction to encapsulate Fork nodes

To avoid the internal error, you must add an activity with CallBehaviorAction to encapsulate Fork nodes in the loops as shown in the figure below. The thread will not increase continually because Simulation can clear the memory after finishing the activity that contains the CallBehaviorAction.



The model uses an activity to encapsulate a fork node

Related pages

- Action
- Execution
- Behavior
- Call Behavior Action