

States and Transitions in RSARTE

Mattias Mohlin
IBM

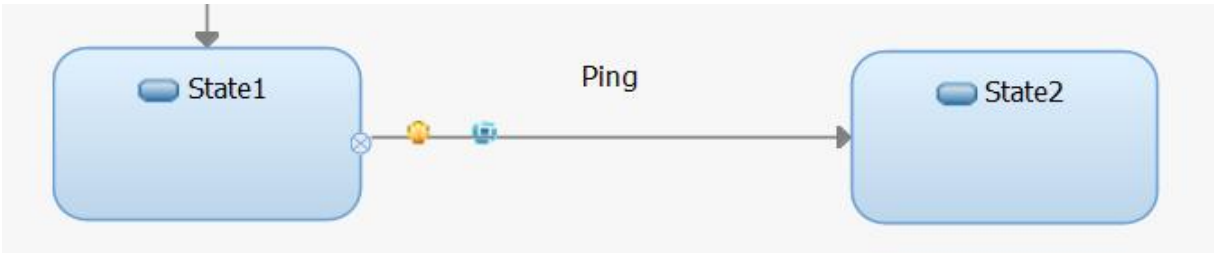
1. External State Transitions....2
   1.1. Effect Code....3
   1.2. Transition Guard....3
   1.3. State Entry and Exit Code....3
   1.4. Creating a Trigger for a Transition....4
   1.5. Receiving a Trigger Event....5
2. Explanation of Symbols....6
3. Sub States....9
   3.1. State Entry Points....10
   3.2. State Exit Points....11
   3.3. Summary of Sub State Transitions....11
   3.4. Code Execution Order....12
4. Kinds of Transitions....14
   4.1. External Self Transitions....14
   4.2. Internal Transitions....15
   4.3. Local Transitions....16
   4.4. Example Use of Different Kinds of Transitions....18
5. State Diagram Appearance....23

This document explains how states and transitions (including self-transitions) work in Rational Software Architect Real-Time Edition (RSARTE). It also covers composite states and state entry and exit points. Note that this is an explanation of what is possible with the tool and is not meant to be a design guide.

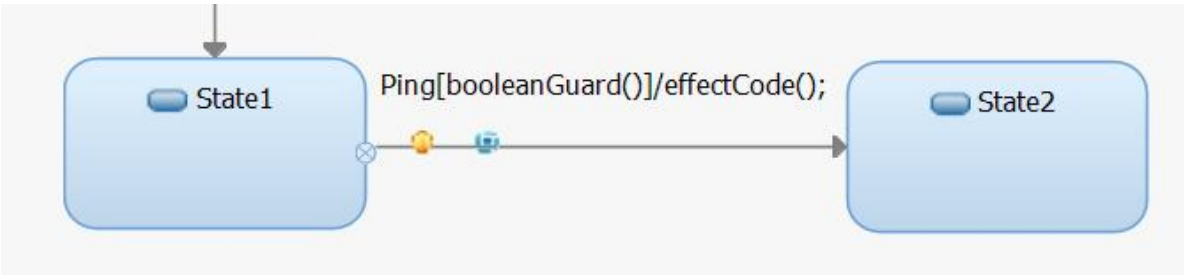
The document was last updated for RSARTE 10.2. All screen shots were captured on the Windows platform.

1. External State Transitions

External state transitions cause an object to change from one state to another. Refer to the picture below. Assume that the state machine is in State1. If the transition Ping is triggered it will cause a state change from State1 to State2. The transition Ping is considered to be an external transition because it causes the state machine to leave one state (State1) and then enter another (State2).



The transition Ping may have a guard (a boolean expression) that determines if the transition is enabled. The transition is enabled when the guard expression evaluates to true. Only enabled transitions can be triggered. A transition may also have effect code that executes when the transition is triggered. An example of guard and effect code is shown in the picture below.



The UML syntax for the text that is shown for a transition is:

```
Transition name [guard] / effect
```

which in our example is: **Ping [booleanGuard()] / effectCode();**

Thus,

- **Ping** is the name of the transition
- **booleanGuard()** is the guard; a boolean expression that evaluates to either true or false
- **effectCode();** is the effect code; code that executes when the transition Ping is triggered

Note the following:

1. The two pictures above come from the same state machine diagram. The only difference is that in the lower picture we have turned on the preferences to show the guards and effects of transitions. See State Diagram Appearance for more information.
2. The actual syntax to use in RSARTE for the above example guard is
return booleanGuard();
That is, the boolean expression must be returned. However, the return was left off for clarity.

In addition to transitions, UML allows each state to have entry and exit code that executes when a state is entered or exited. Transitions, guards, and entry and exit code are further explained in the following sections.

1.1. Effect Code

Effect code is code that executes when the transition is triggered. In our example above, the effect code consists of a call to the operation **effectCode()**. The effect code can be any valid C++ code that you can write in a function body. If it's short (one or two lines) you can consider showing it on the state start diagram. But if it's longer it's usually better to not show it since the diagram can be too cluttered otherwise. Use the state chart diagram filtering preferences (*UML Development - Real Time - Diagrams - State Chart - State Chart Filtering Options*) to define what is visible on a state chart diagram.

It is important that the effect code is not blocking (for example waiting for some input) because this will block the entire capsule, as well as other capsules that run in the same physical thread. All code that executes when a transition triggers should be as quick as possible. If you need to perform longer running operations, you should consider placing such code in its own capsule so that it can be run by a separate thread.

1.2. Transition Guard

A transition guard is a boolean expression that is evaluated before the transition is actually triggered. If the guard is true then the transition is enabled and can be triggered, otherwise it is not triggered and the object remains in the same state.
Examples of guards are:

- X == 1
- isReady() Assuming that isReady() is a boolean function.

For our above example, we used a guard **effectCode()**. This means that if this operation returns true, the transition would trigger and take the object from State1 to State2. However, if the function returns false, then the object would stay in State1.

As mentioned above RSARTE requires that the transition guard expression is returned using a return statement.

1.3. State Entry and Exit Code

States may have entry and exit code. Entry code is code that is executed **every** time the state is entered, and exit code is executed **every** time that the state is exited, regardless of which transition is triggered.

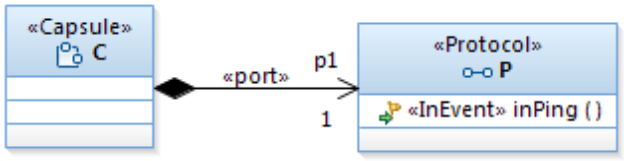
Let's return to our example above. If State1 had exit code and State2 had entry code, then the code would be executed in the following order:

1. State1 Exit Code
2. Ping Effect Code
3. State2 Entry Code

1.4. Creating a Trigger for a Transition

Transitions that originate from a state must have at least one trigger. A trigger represents the arrival of an event on one of the ports of the container capsule. When the specified event arrives on the specified port, the transition may be triggered by the trigger. Triggers may be created for a transition using the Properties view for the transition.

Assume we have a capsule C with a port p1 that is typed by a protocol P which has an in-event inPing.

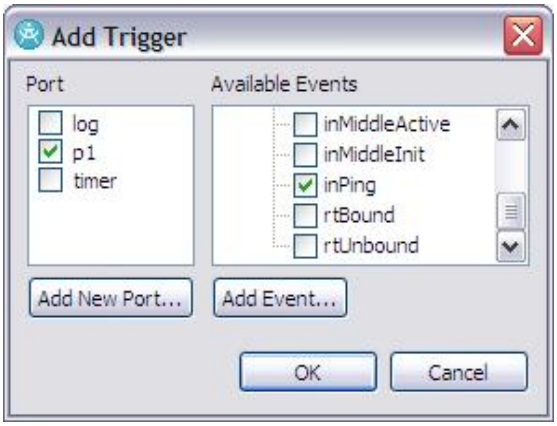


When the inPing event arrives on port p1, then we want it to cause the Ping transition on the state chart to fire. We should therefore create a trigger for the Ping transition.

1. On the state chart, click on the transition.
2. In the **Properties** view, click the **Triggers** tab.
3. Click the **Insert New Transition Trigger**  icon.



4. In the "Add Trigger" dialog select the port and the event.



You can also create a trigger from the context menu of a transition that is selected in a state chart diagram (**Add UML - Trigger**).

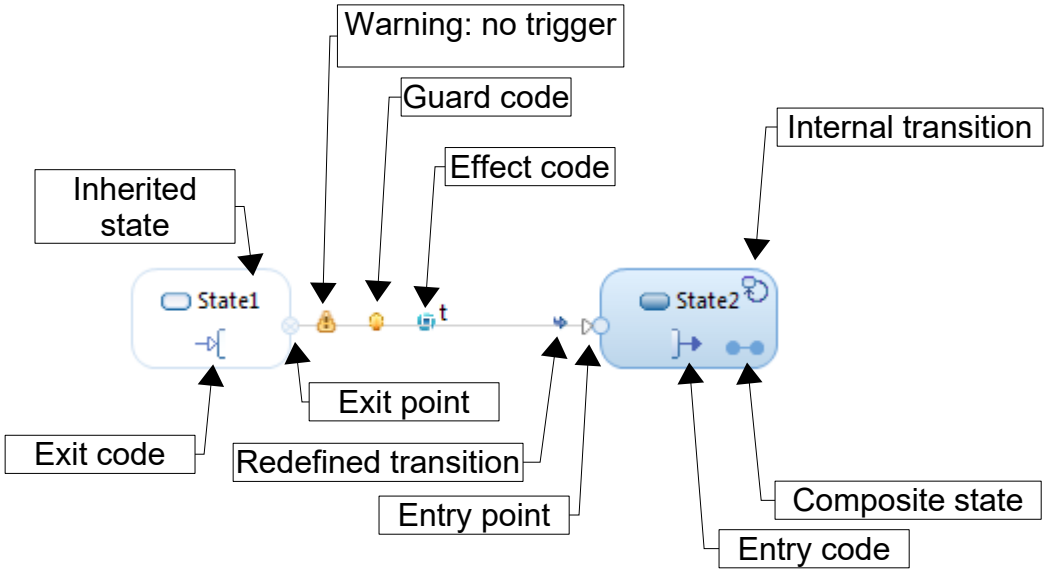
1.5. Receiving a Trigger Event

When the inPing event arrives at the p1 port, the following happens (assuming that the object is initially in State1):

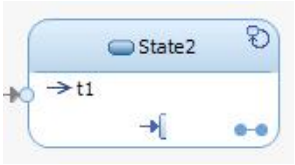
1. Event inPing arrives on port p1
2. The guard on transition Ping is evaluated (either true or false)
3. If the guard is true:
   - State1 exit code is executed
   - Ping effect code is executed
   - State2 entry code is executed
   - The capsule object is now in State2
4. If the guard is false:
   - The capsule object remains in State1 and no more code is executed

2. Explanation of Symbols

The following visual marking symbols may appear on states and transitions:



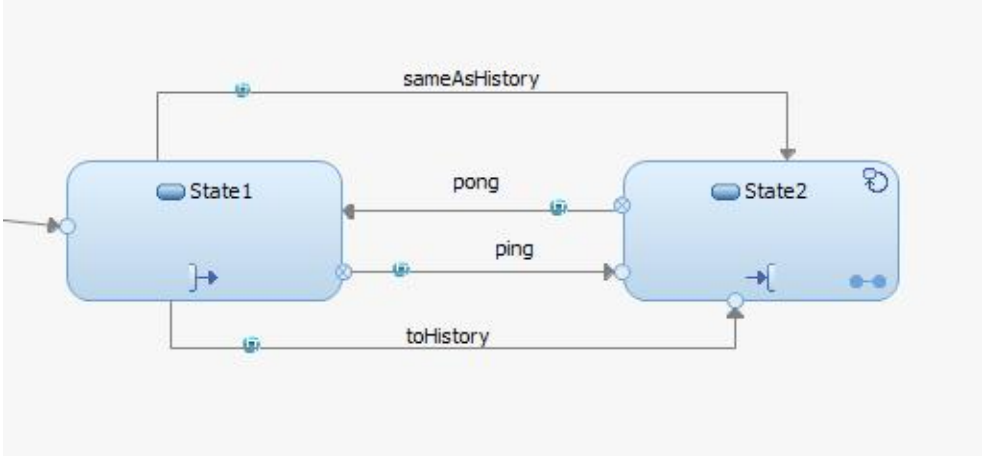
Effect code	The blue symbol on a transition indicates that the transition contains effect code that is executed when the transition is triggered. □ Hovering the mouse over the symbol will display the effect code in a popup window. □ Single clicking the symbol will display the effect code in the code view. □ Double clicking the symbol will open the effect code in an editor.
Entry code	The "incoming arrow" indicates that the state contains entry code that will execute whenever the state is entered. □ Hovering the mouse over the symbol will display the entry code in a popup window. □ Double clicking the symbol will allow you to open the entry code in an editor. Note that if the state has both an entry code an an exit code the symbol looks like this: 
Entry point	The hollow circle on the edge of a state indicates that the transition is entering the state via a specific entry point and can then proceed to a specific sub state. The entry point only has meaning if the state has a sub state (i.e. is a composite state). Otherwise, it is the same as drawing the transition to the edge of the state. See Sub States for more information.

Exit code	The "outgoing arrow" indicates that the state contains exit code that will execute whenever the state is exited. □ Hovering the mouse over the symbol will display the exit code in a popup window. □ Double clicking the symbol will allow you to open the exit code in an editor.
Exit point	A circle with an X inside indicates that the transition is leaving the state from a specific exit point. This allows you to connect a transition to a specific sub state of a state. An exit point only has meaning if the state has a sub state (i.e. is a composite state). Otherwise, it is the same as drawing the transition from the edge of the state. See Sub States for more information.
Guard code	The yellow symbol on a transition indicates that the transition contains a guard expression (boolean) that is evaluated before the transition is triggered. The transition may be triggered only if the guard is true. Otherwise, there is no change of state. □ Hovering the mouse over the symbol will display the code in a popup window. □ Single clicking the symbol will display the code in the code view. □ Double clicking the symbol will open the code in an editor.
Internal transition	The symbol in the upper right corner of a state indicates that the state contains at least one internal transition. □ Double clicking the symbol will display the internal transitions inside the state as shown below. In this example, t1 is an internal transition.  See Internal Transitions for more information about internal transitions.
Composite state	The two connected circles in the lower right hand corner of a state indicate that it is a composite state, and therefore may contain sub states. See Sub States for more information.

Warning: no trigger	The yellow warning triangle on a transition indicates that the transition does not have any triggers defined, and that it must have at least one trigger. In this case the Triggers tab of the Properties view for the transition is empty:  How to create triggers for a transition is described in Creating a Trigger for a Transition.
Redefined transition	The small blue arrow shows that a transition is redefined. This means that there is an inherited version of the transition in the state machine of one of the super classes. The redefined transition contains some modification compared to this inherited transition. For example, it may have a different target state. The redefinition arrow may appear on all elements that may be redefined, not just transitions.
Inherited state	Inherited states are shown as grayed or dashed (depending on the preference *UML Development – Real Time – Diagrams – State Chart – Line style to be used for inherited elements*) compared to locally defined states. This applies to all elements that are inherited, not just states.

3. Sub States

States may contain sub states that allow you to break down the behavior of a state (composite state) into smaller states (sub states). The picture below shows two states, State1 and State2. State 2 contains a sub state that describes the behavior when the state machine is in State2.

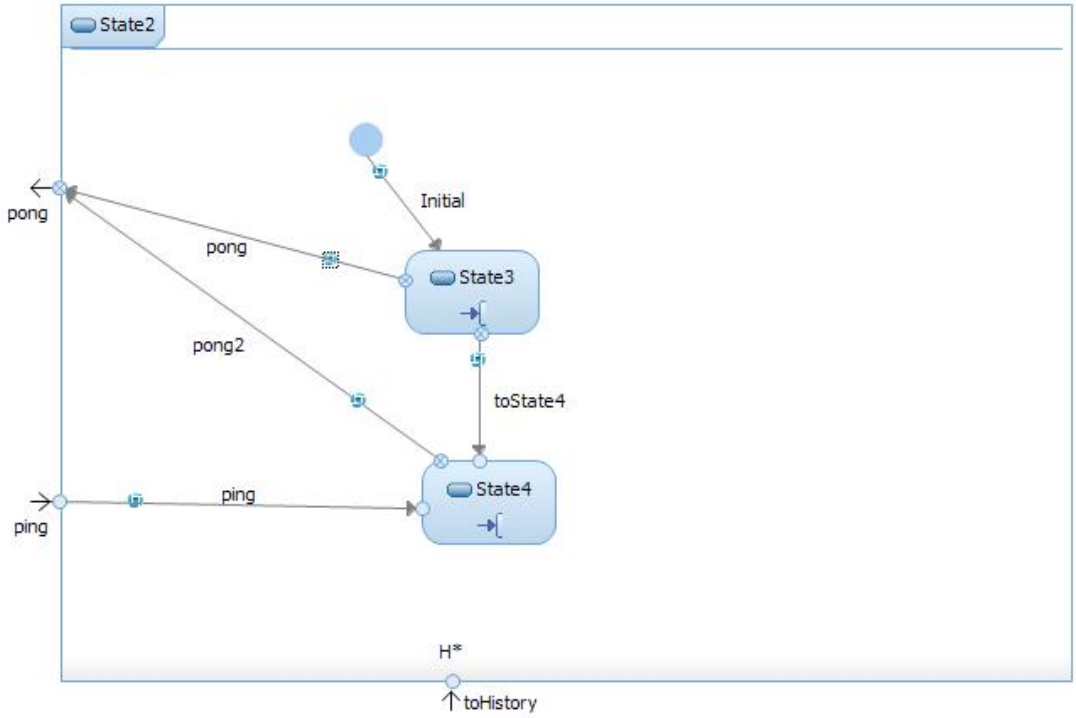


There are three transitions from State1 to State2:

- ping: to an entry point on State 2 (labeled **ping** in the sub state)
- toHistory: to an entry point (labeled **toHistory, H*** in the sub state)
- sameAsHistory: to the edge of State2 (which behaves the same as history)

There is one transition from State2 back to State1:

- pong: leaving an exit point of State2 (labeled **pong** in the sub state)



We'll go through the modeling constructs used in this example in the sections below.

3.1. State Entry Points

Entry and exit points allow transitions at different levels in a state machine hierarchy to be connected. A transition can terminate at an entry point on the edge of a composite state. Another transition can then connect the entry point with a sub state inside the composite state. Thereby it becomes possible to enter a composite state and proceed directly to a certain sub state.

There are three ways in which a transition can enter a composite state:

1) Via an entry point
2) Via a history point
3) Directly to the edge of the state.

Enter via an entry point

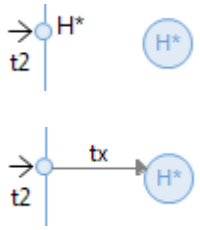
In the above example the transition ping enters State2 via an entry point. If ping is triggered then the sub state State4 will be entered.

Note that the transitions in the sub state diagram that originate at entry points do not have any triggers themselves. These transitions cannot be triggered directly on their own, but only indirectly when the transitions that connect at the entry points in the enclosing state machine get triggered. They hence behave similar to a transition that leave a junction point, and may have effect code and guard, but no triggers.

Enter via a history point

In the above example the transition toHistory enters State2 via a history point. This is a special kind of entry point which is not connected by a transition inside the sub state diagram. It is also marked by "H*" which denotes that it is a history point.

The RSARTE notation to mark entry points with "H*" to denote a history point is a shorthand for using an explicit deep history pseudo state. Hence the following two ways of modeling are equivalent (provided that the transition tx does not have a guard or effect code):



When entering a state via a history point the sub state that was most recently active will be entered to become active again. The history is "deep" which means that if the activated sub state also is a composite state, it too will be entered using a history point. This process continues recursively until a non-composite state is entered. If a composite state has not been entered before, so that there is no historically active sub state, then its initial transition will be used to decide which sub state that will become active.

Let's look at what these rules mean for our example. The first time the toHistory transition is triggered so that State2 is entered via the history point, then the sub state State3 will be entered, because the initial transition targets this sub state. Then, assume the transition toState4 gets triggered and then the pong2 transition so that we leave State2. The next time the toHistory transition triggers we will now enter the sub state State4, since that is the most recently active sub state.

Enter without using an entry point
If you draw a transition to the edge of a composite state, without using an entry point, it actually means the same as if you would have connected it to a history point. The only difference is that the incoming transition is not visible from the sub state diagram.

In our example the sameAsHistory transition connects directly to the edge of State2. When it triggers the same thing happens as described above for the toHistory transition.

RSARTE encourages the usage of entry points for composite states, since it becomes more clear what will happen when the state is entered. Therefore, if you have a non-composite state with incoming transitions that connect directly to its edge, and then turn this state into a composite state, then entry points will be automatically created for all such incoming transitions.

It should be noted that the interpretation to treat a transition that ends on the edge of a composite state to be equivalent with entering the state using a history point is a non-standard extension of UML2. According to the UML2 standard such a transition performs a "default entry" of the state which should cause the initial transition to execute. In RSARTE this only happens if this is the first time the composite state is entered.

3.2. State Exit Points

An exit point behaves very much like a junction point. When a transition that targets an exit point gets triggered, then the composite state will be exited, and then the transition that originates at the exit point in the enclosing state chart diagram will execute. Hence, a transition that originates at an exit point may not have a trigger, but can have effect code and a guard.

In our example the transition pong leaves State2 via an exit point. If it is triggered the transition pong in the enclosing state machine will then execute, which will cause State1 to be entered.

A composite state can also be exited without using an exit point, through a transition that is directly connected to the edge of the composite state. Such a transition can be triggered regardless of which sub state that is active.

In our example the transitions toHistory and sameAsHistory are drawn directly from the edge of State1. Now, State1 is not a composite state, but if it was, these transitions could be triggered regardless of which sub state of State1 that was active.

3.3. Summary of Sub State Transitions

The following table summarizes the behavior of different kinds of sub state transitions using the above example.

Transition	Example	Description
To an entry point	ping	When the ping transition from State1 to State2 is triggered, then State2 will enter sub state State4 directly.

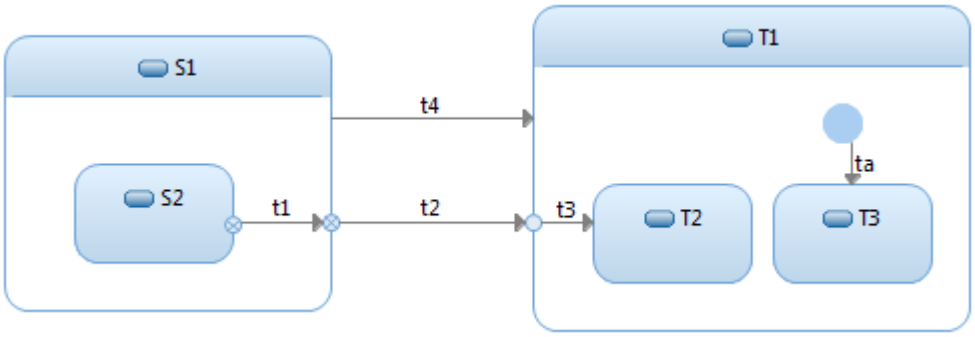
To a history point	toHistory	When the toHistory transition is triggered from State1 to State2, the sub state of State2 that gets activated will be the same state as was active when State2 was last exited. If this is the first time that State2 is entered then the sub state State3 will be activated since this is the state targeted by the initial transition.
To the edge of a state	sameAsHistory	This behaves exactly the same as the to-History transition as described above. Note that this transition is not visible in the sub state diagram.
From an exit point	pong	The pong exit transition will be taken if either of the following occurs: ▪ The active sub state is State3 and the transition pong gets triggered ▪ The active sub state is State4 and the transition pong2 gets triggered
From the edge of a state	toHistory	If State1 had a sub state, then the toHistory transition could be triggered regardless of which of its sub states that was active.

3.4. Code Execution Order

When an event arrives that matches a trigger for a transition that exits and/or enters a composite state, there are many code snippets that will execute. The first thing that happens is that several guard conditions are evaluated, starting with the triggered transition. The goal with this guard evaluation procedure is to find a path of transitions (a so called compound transition) with fulfilled guard conditions, starting with the triggered transition and ending with the last transition that would execute if that triggered transition would trigger. If during this evaluation procedure a transition is found along the path which has a false guard condition, then the current path of transitions is not enabled, and the evaluation procedure backtracks and tries to find another enabled path.

When a path of enabled transitions have been found the next thing that happens is that the effect codes of the transitions execute. For all transitions which exit a state, the exit code of that state is first executed, before executing the effect of the transition. And when reaching a transition on the path which enters a state, the entry code of that state executes after executing the effect of the transition.

Let's look at an example which illustrates the above mentioned rules:



Assume the following:

- The active state is S2. This means that S1 also is active, since it encloses S2. We talk about an active state configuration that consists of a set of active states; in this case {S2, S1}.
- Both t1 and t4 triggers on the same event E received on the same port P.

The following happens when E arrives on P if the guard of t2 is false, and all other guards are true:

1. The guard of t1 and t2 are evaluated. Since the guard of t2 is false, guard evaluation of this transition path is aborted, and the guard of t3 does not get evaluated.
2. There are no more outgoing transitions from S2, so next the transitions of the enclosing state S1 are traversed which means that the guard of t4 is evaluated. This transition is the first path that is found where all the guard conditions (just one) are fulfilled, so this transition is enabled and will be triggered.
3. The exit code of S2 executes.
4. The exit code of S1 executes.
5. The effect code of t4 executes.
6. The entry code of T1 executes.
7. The effect code of ta executes.
8. The entry code of T3 executes.

Here we have assumed that T1 is entered for the first time, and hence would use its initial transition to arrive at sub state T3.

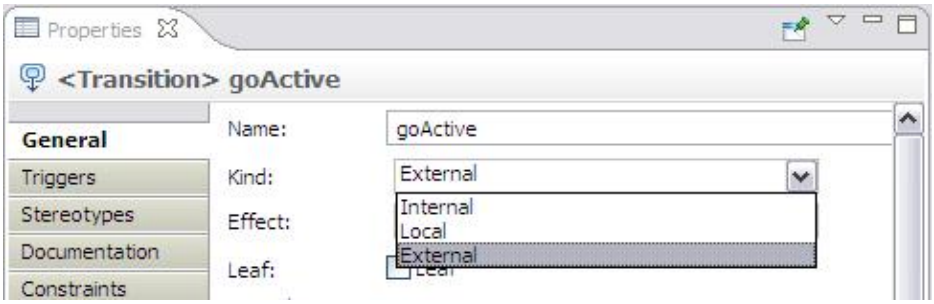
If instead all guard conditions are true at the time when E arrives on P, then the following will happen:

1. The guard of t1, t2 and t3 are evaluated. All are fulfilled which means that this path of transitions is enabled and t1 will be triggered.
2. The exit code of S2 executes.
3. The effect code of t1 executes.
4. The exit code of S1 executes.
5. The effect code of t2 executes.
6. The entry code of T1 executes.
7. The effect code of t3 executes.
8. The entry code of T2 executes.

4. Kinds of Transitions

There are three different kinds of transitions: external, internal, and local. They differ in how the entry and exit code is handled.

The transition kind is set in the **Kind** field in the Properties view of the transition.

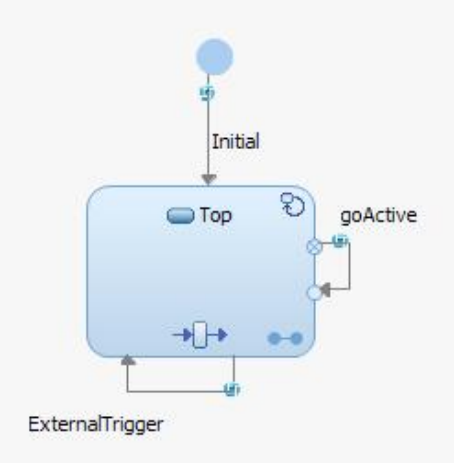


A special category of transitions are self-transitions. These are transitions that have the same state as source and target. All internal transitions are self-transitions, while external and local transitions may be, but do not have to be, self-transitions.

4.1. External Self Transitions

External self transitions behave like other external transitions (see External State Transitions), i.e. the state is exited and then it is entered again. This means that the code execution order is the same as for all other external transitions; first the exit code of the state executes, then the effect code of the transition and finally the entry code of the same state.

The picture below shows a state Top with two external self-transitions, goActive and External-Trigger.



Top is a composite state which means that when either of these self-transitions is triggered, sub states will first be exited (from inside out), then Top will be exited, then the transition effect will

execute, then Top will be entered, and finally sub states will be entered (from outside in). In general the following happens:

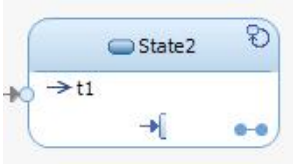
- Exit code for the innermost state is executed
- Exit code for the next outer state is executed
- .....
- Exit code for the outermost state is executed
- Transition effect code is executed
- Entry code for the outermost state is executed
- Entry code for the next inner state is executed
- .....
- Entry code for the innermost state is executed

See Code Execution Order for the detailed rules of code execution when entering and exiting a composite state.

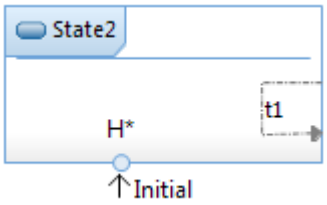
4.2. Internal Transitions

When an internal transition triggers the state machine stays in the current state. This means that the only code snippet that executes is the effect code of the internal transition. The exit and entry code of the state does not execute.

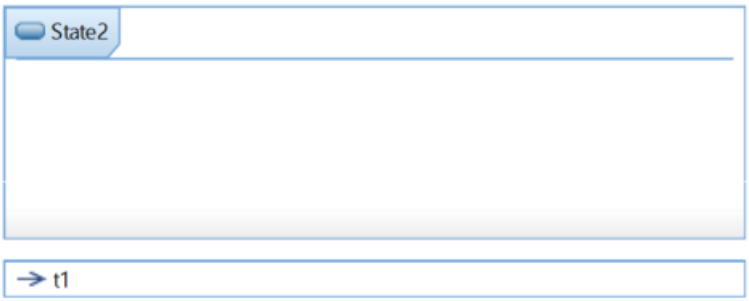
Internal transitions are normally shown in a separate compartment in a state symbol. An icon in its top-right corner shows the presence of internal transitions, and you can double-click this icon to show or hide the internal transitions compartment. Here is an example of a state with an internal transition t1:



Inside State2 the internal transition can be shown as a border transition using a dashed line. It is placed inside the state chart diagram to further emphasize that an internal transition does not leave the state when it is triggered.



However, RSARTE also supports an alternative visualization of internal transitions where they are shown in a special compartment below the state chart diagram (either inside or outside the diagram frame), similar to how they are shown on the enclosing state.



Which visualization to use is controlled by a preference (*UML Development – Real Time – Diagrams – State Chart – Enclosing State Internal Transitions*).

An internal transition can be used as an event error handler. For example, assume that you have a state machine with many states and transitions. If an event arrives when the state machine is in an unexpected state, then the event could go unhandled. To avoid this you can create an enclosing composite state (let's call it Top) with an internal transition for every event that can be received by the state machine. If everything is functioning as it should, these internal transitions will never execute since the transitions that go between the sub states will be triggered instead of the internal transitions on Top. However, if an event arrives which does not trigger any of the other transitions (because the state machine is in an unexpected state) then an internal transition in Top will be triggered, and can handle the unexpected situation in its effect code.

If states do not contain exit or entry code, an external self-transition is functionally equivalent with an internal transition. In this case it's recommended to use internal transitions:

- Diagrams get less cluttered with lines on the border of states
- Generated code becomes more efficient since there is no need to invoke any exit or entry code snippets.

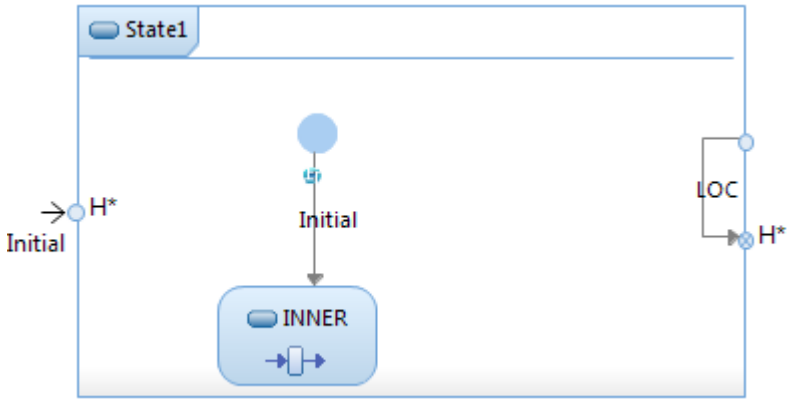
You can convert external self-transitions to internal transitions using a command that is available on the transition in the Refactor context menu: **Convert to Internal Transition**. A similar command is also available on a state in case you want to convert all its external self-transitions to internal transitions.

4.3. Local Transitions

Local transitions execute the effect code for the transition and the exit and entry code for all sub states but **not** the exit and entry code for the state to which it is connected. Local transitions are hence only meaningful for composite states (if the state is non-composite they behave the same as internal transitions).

A local transition always originates from the border of a sub state chart diagram. They are drawn using solid lines to distinguish them from internal transitions, in case the local transition is a self-transition.

Here is an example of a composite state State1 with a local self-transition LOC:



If the state INNER is active when the LOC transition is triggered the following will happen:

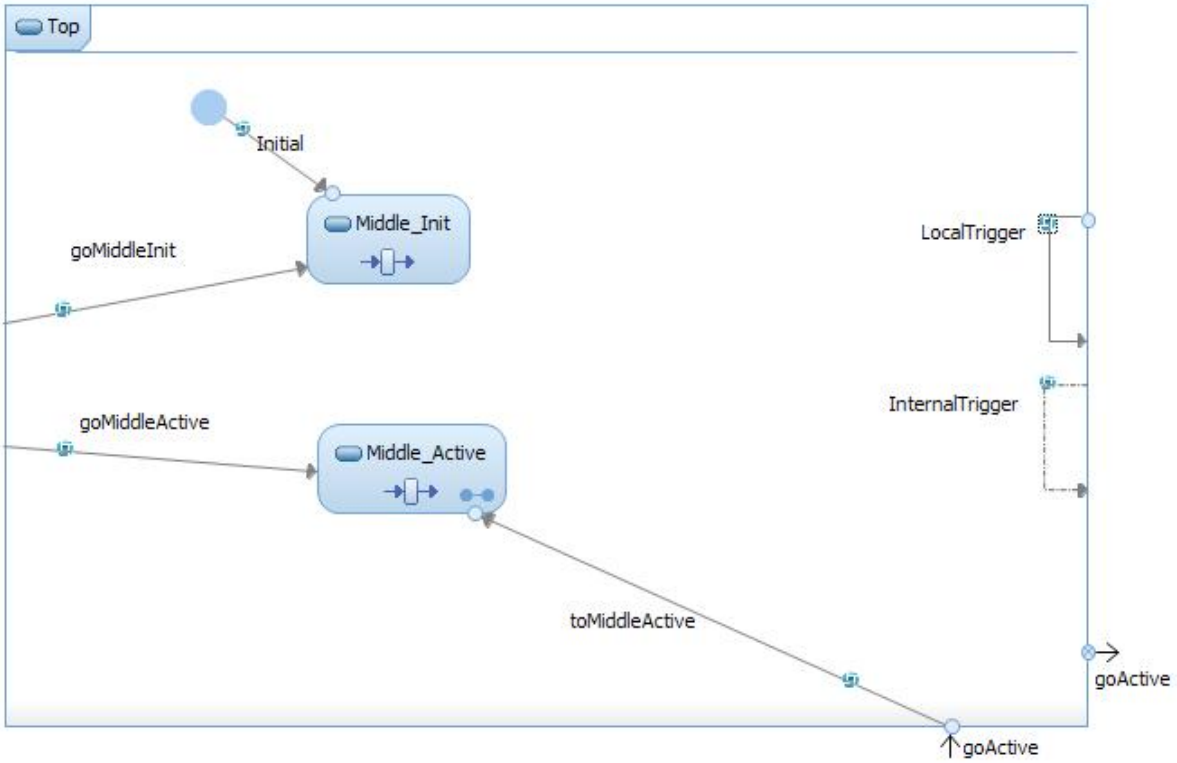
1. The exit code of INNER executes.
2. The effect code of LOC executes.
3. The entry code of INNER executes. Note that the local transition terminates at a history point, and since INNER was previously active it will be entered again without executing the initial transition.

In general the following happens when a local transition is triggered:

- Exit code for the innermost state is executed
- Exit code for the next outer state is executed
- .....
- Exit code for the outermost state is NOT executed
- Transition effect code is executed
- Entry code for the outermost state is NOT executed
- Entry code for the next inner state is executed
- .....
- Entry code for the innermost state is executed

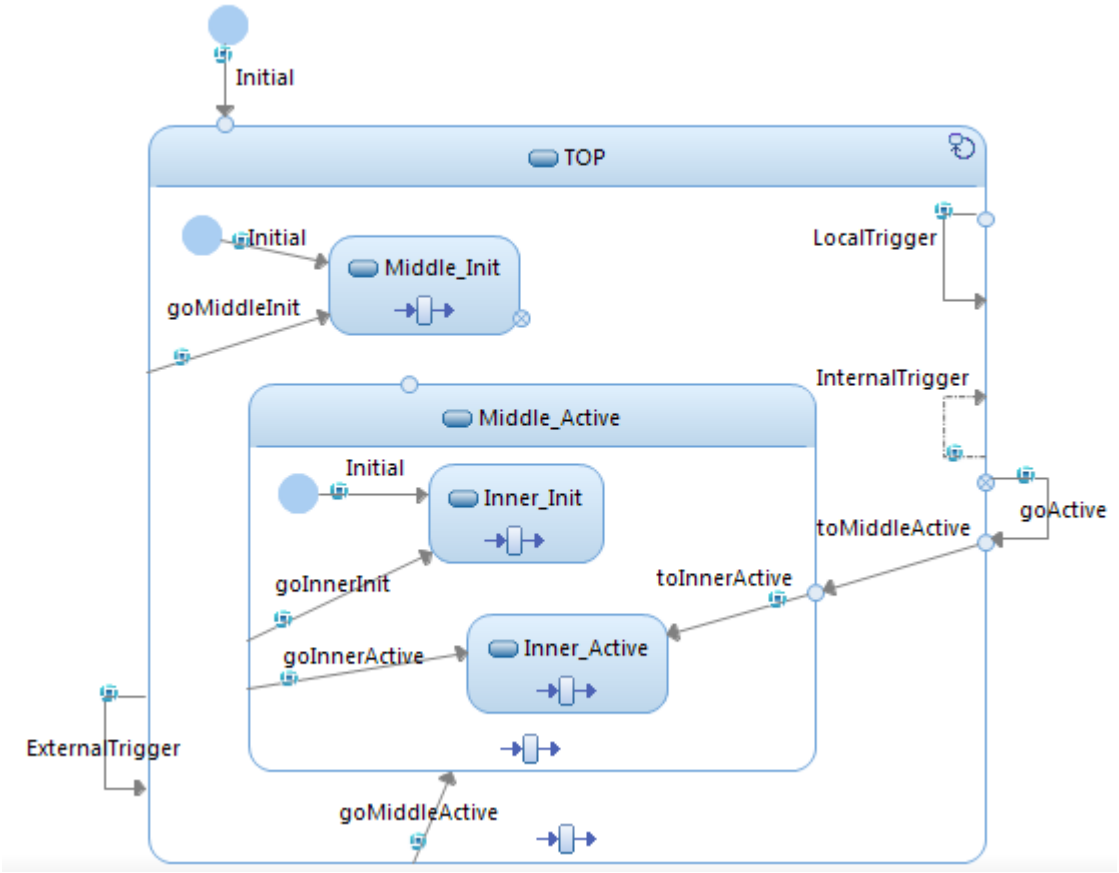
Local transitions may be used to simplify a state diagram. If we have many states and want to be able to go to a specific state regardless of which state we are in, then we could create an enclosing composite state and draw a local transition from that state to the state that we want to go to. Without the enclosing composite state, we would have to create a transition from every state to the desired target state.

An example of this is the goMiddleActive transition shown in the picture below. It originates from the edge of the Top state and terminates at the Middle_Active sub state. When the transition is taken, it will cause the Top state to go to the Middle_Active sub state regardless of which sub state of Top that is active. The transition is local to Top and will not cause any exit or entry code of Top to execute, but only the entry code of Middle_Active and its sub states.



4.4. Example Use of Different Kinds of Transitions

As an example of using the different kinds of transitions let's look a little more on the example with the Top state that was shown previously. To get a better overview look at the picture below where the Top state is shown with its sub states visible inline.



The Top state has the following transitions:

External transitions:

- ExternalTrigger
  Leaves state Top and returns back to the edge of state Top. Therefore, it is the same as entering Top via a history point.

- goActive
  Leaves an exit point and returns to a specific entry point which will go directly to the Middle_Active sub state via the toMiddleActive transition and then further to the Inner_Active sub state via the toInnerActive transition.

- Initial
  Makes the Top state the default state of its state machine. When the capsule is incarnated, the Initial transition is taken and its effect code executes. Then Top is entered.

Top contains two sub states, Middle_Init and Middle_Active, and the following transitions:

Internal Transition

- InternalTrigger
  When this transition triggers, only its effect code is executed. No entry or exit code is executed.

Local Transitions

- LocalTrigger

This transition will execute its effect code and the exit and entry code for the active sub state (direct or indirect), but it will not execute the exit and entry code for Top.

- goMiddleInit
  When this transition triggers it will cause a transition to state Middle_Init regardless of if Middle_Init or Middle_Active was the active state. The entry code for Middle_Init will be executed but the exit and entry code for Top will not be executed.

- goMiddleActive
  When this transition triggers it will cause a transition to state Middle_Active regardless of if Middle_Init or Middle_Active was the active state. The entry code for Middle_Active will be executed but the exit and entry code for Top will not be executed.

- toMiddleActive
  This transition originates at the entry point where the goActive transition comes in from the Top state in the enclosing state machine. Therefore this transition may not have any triggers. It can only execute when the goActive transition is triggered. However, this transition may have effect code and a guard condition.

External Transition

- Initial
  This initial transition is taken when entering the Top state without using one of the entry points. This happens when incarnating the capsule since the initial transition of the enclosing state machine goes to Top.

Finally, let's look at the Middle_Active state machine. It contains two sub states, Inner_Init and Inner_Active, and the following transitions:

Local Transitions

- goInnerInit
  When this transition triggers it will cause a transition to state Inner_Init regardless of if Inner_Init or Inner_Active was the active state. The entry code for Inner_Init will be executed but the exit and entry code for Middle_Active will not be executed.

- goInnerActive
  When this transition triggers it will cause a transition to state Inner_Active regardless of if Inner_Init or Inner_Active was the active state. The entry code for Inner_Active will be executed but the exit and entry code for Middle_Active will not be executed.

- toInnerActive
  This transition originates at the entry point where the toMiddleActive transition comes in. If we look in the enclosing state machines we see that this transition path originates in the Top state and consists of the transitions goActive, toMiddleActive and toInnerActive. All these transitions may have their own effect codes and guard conditions, but only the first one (goActive) may have a trigger.

External Transition

- Initial

This initial transition is taken when entering the Middle_Active state without using one of the entry points. This happens when incarnating the capsule since the initial transition of the enclosing state machine goes to Middle_Active, and the initial transition of the top-level state machine goes to Top.

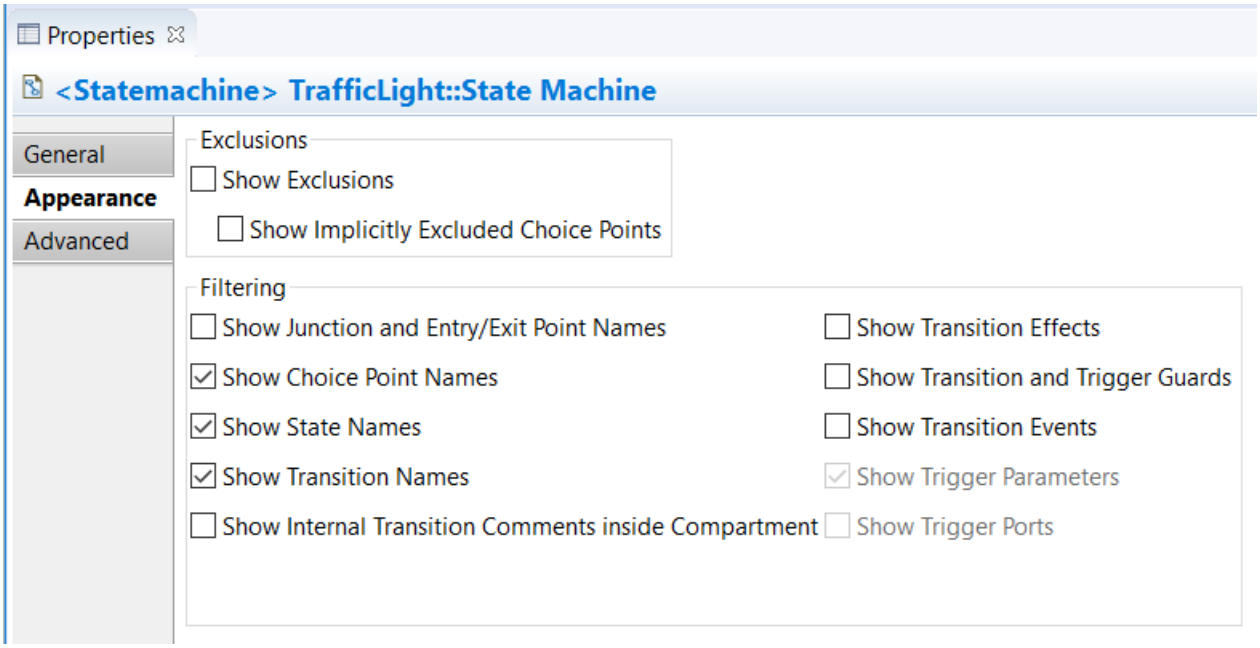
The table below shows some scenarios for the above example and the order of code execution that takes place in each case:

Scenario	**Initially Active States**	**Code Execution Sequence**
Capsule incarnation	None	▪ Effect code of Initial (to Top state) ▪ Entry code of Top ▪ Effect code of Initial (to Middle_Init state) ▪ Entry code of Middle_Init
The local transition goMiddleActive is triggered	Top, Middle_Init	▪ Exit code of Middle_Init ▪ Effect code of goMiddleActive ▪ Entry code of Middle_Active ▪ Effect code of Initial (to Inner_Init state) ▪ Entry code of Inner_Init Note that the Top state is never left since goMiddleActive is a local transition.
The local transition LocalTrigger is triggered	Top, Middle_Active, Inner_Init	▪ Exit code of Inner_Init ▪ Exit code of Middle_Active ▪ Effect code of LocalTrigger ▪ Entry code of Middle_Active ▪ Entry code of Inner_Init Note that the Top state is never left since LocalTrigger is a local transition. Both Inner_Init and Middle_Active are however exited and then entered again.
The internal transition InternalTrigger is triggered	Top, Middle_Active, Inner_Init	▪ Effect code of InternalTrigger Since InternalTrigger is internal no states will be exited and entered when it triggers. Only its effect code executes.
The local transition goMiddleInit is triggered	Top, Middle_Active, Inner_Init	▪ Exit code of Inner_Init ▪ Exit code of Middle_Active ▪ Effect code of goMiddleInit ▪ Entry code of Middle_Init
The external transition goActive is triggered	Top, Middle_Init	▪ Exit code of Middle_Init ▪ Exit code of Top ▪ Effect code of goActive ▪ Entry code of Top ▪ Effect code of toMiddleActive ▪ Entry code of Middle_Active ▪ Effect code of toInnerActive ▪ Entry code of Inner_Active goActive is an external transition so all states are exited all the way up to Top. The path of transitions that start with goActive consists of 3 transitions and leads to the Inner_Active state.

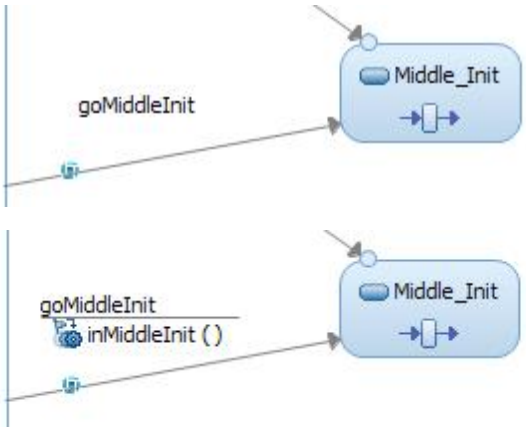
The external transition ExternalTrigger is triggered	Top, Middle_Active Inner_Active	▪ Exit code of Inner_Active ▪ Exit code of Middle_Active ▪ Exit code of Top ▪ Effect code of ExternalTrigger ▪ Entry code of Top ▪ Entry code of Middle_Active ▪ Entry code of Inner_Active Note that the ExternalTrigger transition terminates at the border of the TOP state. This is equivalent of entering the state using a history point, which is why the states Top, Middle_Active and Inner_Active are entered again, without executing the initial transitions.

5. State Diagram Appearance

The appearance of a state chart can be changed by using the Properties view of the selected state chart diagram. As a reader of a state chart diagram you may find it useful to temporarily change some of the appearance settings. For example, turning **Show Transition Effects** on could help to debug the transition effect code. Another setting that could be useful is the **Show Transition Events**. It will display the event(s) that trigger a transition. The picture below shows the Appearance tab of the Propertites view for a state diagram.



The picture below shows the effect of the "Show Transition Events" property. When it is checked you can see in the diagram that the goMiddleInit transition triggers on the inMiddleInit event.



If you only temporarily want to change the state diagram appearance, make sure to have the preference *UML Development – Real Time – Diagrams – Persist workspace filters* turned off. Only turn this preference on in case you want to save the appearance settings in the diagram.