Modeling Object State Using State Machine Diagrams

Objectives

Upon completion of this module, you should be able to:

- Model object state
- Describe the essential elements of a UML State Machine diagram

Additional Resources

Additional resources – The following references provide additional information on the topics described in this module:

- Folwer, Martin, with Kendall Scott. *UML Distilled (2nd ed)*. Reading: Addison Wesley Longman, Inc., 2000.
- Booch, Grady, James Rumbaugh, and Ivar Jacobson. *The Unified Modeling Language User Guide*. Reading: Addison Wesley Longman, Inc., 1999.
- Gamma, Erich, Richard Helm, Ralph Johnson, and John Vlissides. *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley, 1995.
- Rumbaugh, James, Jacobson Ivor, and Booch Grady. *The Unified Modeling Language Reference Manual (2nd ed)*. Addison-Wesley, 2004.
- The Object Management Group. "OMG Unified Modeling Language™ (OMG UML), Superstructure," [`http://www.omg.org/spec/UML/2.2/Superstructure/PDF/`], Version 2.2, February 2009.

Note – The mathematical basis of the Statechart diagram is founded in the state machine theories of Meale and Moore. (See Booch, Rumbaugh, and Jacobson UML User Guide page 336).

Process Map

This module describes the next step in the Design workflow: modeling the state behavior of a complex object. Figure 9-1 shows the activity and artifact discussed in this module.

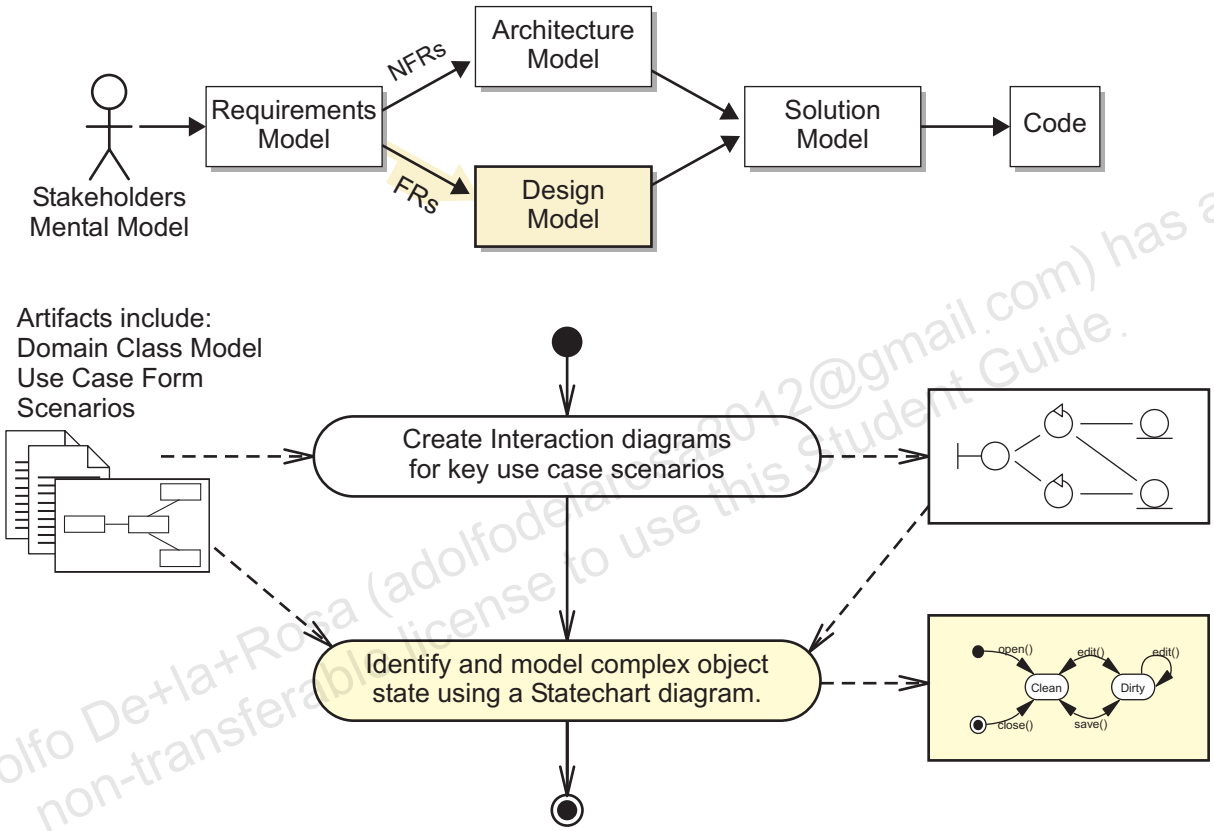


Figure 9-1 Design Workflow Process Map

Modeling Object State

This section discusses what object state is and how you can model state information using the UML.

Introducing Object State

State is “1a: mode or condition of being” (Webster)

There are two ways to think about object state:

- The state of an object is specific collection of attribute values for the object.

  Typically people think of object state as the state of the attributes in an object. This view is certainly valid, but it misses an important understanding about object behavior.

  For example, imagine a software system that is controlling a heating, ventilating, and air-conditioning system (HVAC). You could think of the state of the HVAC system in terms of whether the power is on, and whether the current (real) temperature is within a certain range (upper bound and lower bound temperatures).

- The state of an object describes the behavior of the object relative to external stimuli.

  This view is behavior focused rather than data focused. In the HVAC example, there are four basic states: Initial (meaning the unit has not been turned on), Idle (the unit is on but not active), Cooling (the unit is on and cooling), and Heating (the unit is on and heating). These states talk about behaviors of the HVAC.

  There are also triggers to change states. For example, if the HVAC unit is on and the real temperature is recorded as being above the upper bound, then the system should begin cooling (transition to the Cooling state).

This module describes how to model and implement complex objects with state. This module will use the HVAC system example throughout.

Identifying the Elements of a State Machine Diagram

> A state machine is "A behavior that specifies the sequences of states that an object or an interaction goes through during its life in response to events, together with its responses and actions." (UML v1.4 page B-18)

A State Machine diagram *State Machine diagram* represents the set of states for an object or class, the activities of a given state, and the triggers that transition an object from one state to another. Figure 9-2 shows an example State Machine diagram.

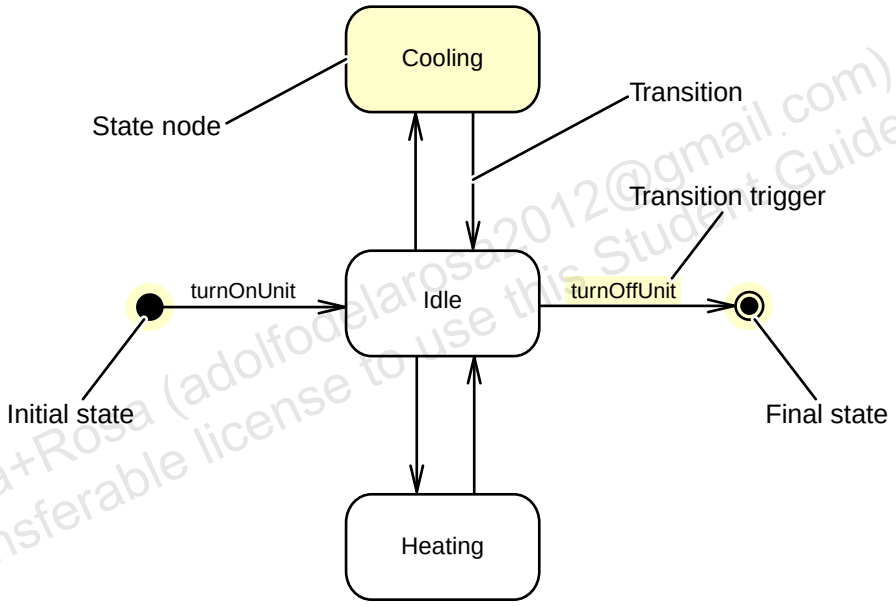


Figure 9-2 Example State Machine Diagram

Every State Machine diagram must include Initial and Final state nodes. Typically, these nodes do not have any behavior associated with them. A state node is represented with a rounded-corner rectangle. A transition from one state to another is represented with a solid line with a stick arrowhead. A transition usually has a trigger associated with it. This trigger is usually an external stimulus on the object. In the HVAC example, turning the HVAC unit on triggers a transition from the Initial state to the Idle state.

This diagram is incomplete without the triggers on the Idle to Cooling or Heating states.

State Transitions

A state transition represents a change of state at runtime. A state transition usually includes a trigger event declaring when an object changes from one state to another. For example, if the HVAC system is in the Idle state and the real temperature rises above the upper bound, then the HVAC object transitions to the Cooling state. A transition can also have a guard condition and a set of actions. In this example, the HVAC can only begin cooling if the AC subsystem has been installed for this particular HVAC system; if there is no AC, then no cooling can occur. There is also an action on this transition to turn the HVAC fans on. Figure 9-3 shows the state transitions for the HVAC system.

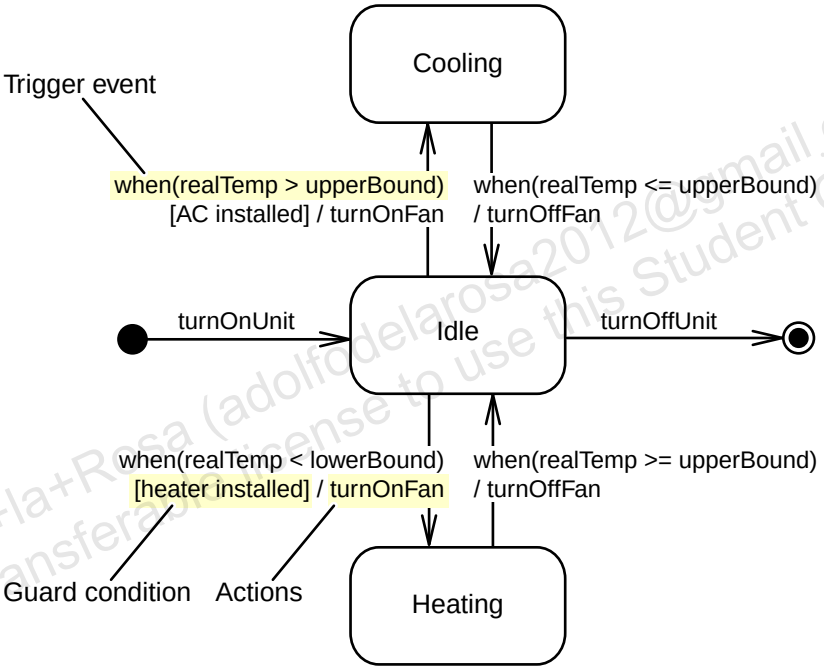


Figure 9-3 HVAC State Machine Diagram With Transitions

Internal Structure of State Nodes

State nodes represent a state of a single object at runtime. The internal structure of a state node can include specific actions to be taken in the case of object events that do not transition out of the given state. Figure 9-4 shows the internal structure of state nodes.

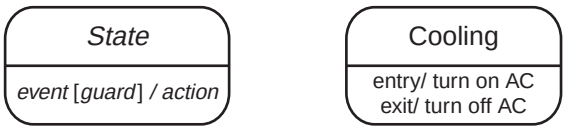


Figure 9-4 Internal Structure of State Nodes

The types of events include:

- Entry – Specifies actions upon entry into the state.
- Exit – Specifies actions upon exit from the state.
- Do – Specifies ongoing actions.

  You can also specify specific events with corresponding actions.

Entry, exit, and do events are specified by the UML.

Figure 9-5 shows a complete State Machine diagram for the HVAC example.

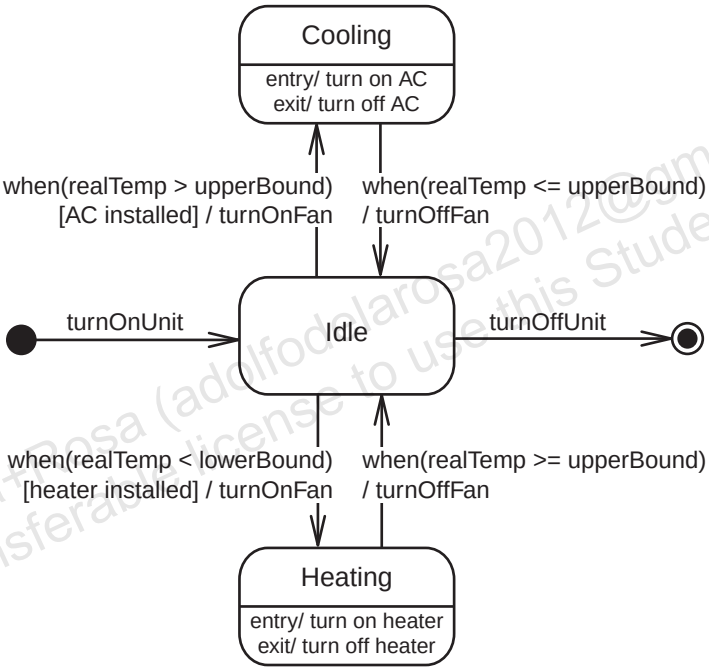


Figure 9-5 Complete HVAC State Machine Diagram

Creating a State Machine Diagram for a Complex Object

Creating a State Machine diagram requires a deep understanding of the object that is modeled. This analysis is beyond the scope of this course.

You can use the following steps to simplify the creation of a State Machine diagram after the states of the object are understood:

1. Draw the initial and final state for the object.
2. Draw the stable states of the object.
3. Specify the partial ordering of stable states over the lifetime of the object.
4. Specify the events that trigger the transitions between the states of the object. Specify transition actions (if any).
5. Specify the actions within a state (if any).

Step 1 – Start With the Initial and Final States

The first step is easy. Place the initial and the final state nodes in the diagram. Do not forget to place these states, because every State Machine diagram must include them. Figure 9-6 illustrates how to place the initial and final nodes.

Figure 9-6 Step 1 – Start With the Initial and Final States

Step 2 – Determine Stable Object States

Then determine the stable states of the object (besides the Initial and Final) and create state nodes for each of these object states. Figure 9-7 illustrates this step.

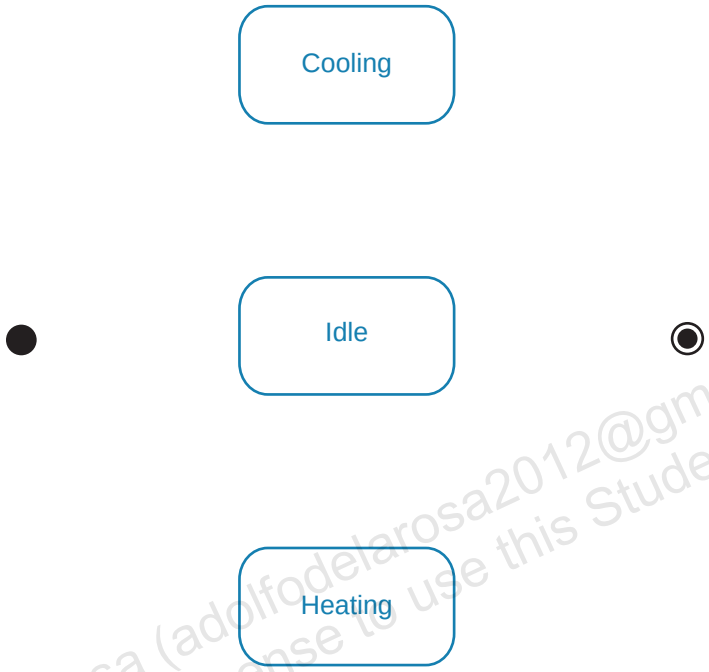


Figure 9-7 Step 2 – Determine Stable Object States

A stable state is the set of conditions and behaviors in which an object can reside for a significant length of time. A transitory state is one in which the object upon entering, immediately exits to some other state. You should model transitory states if there is any significant behavior of the object during entry and exit.

Step 3 – Specify the Partial Ordering of States

Then specify the transitions between the object states; this is known as a partial ordering of the states. For example, an HVAC object can transition from the Idle state to the Cooling or Heating states, but the Cooling state cannot transition to the Heating state. Figure 9-8 illustrates this.

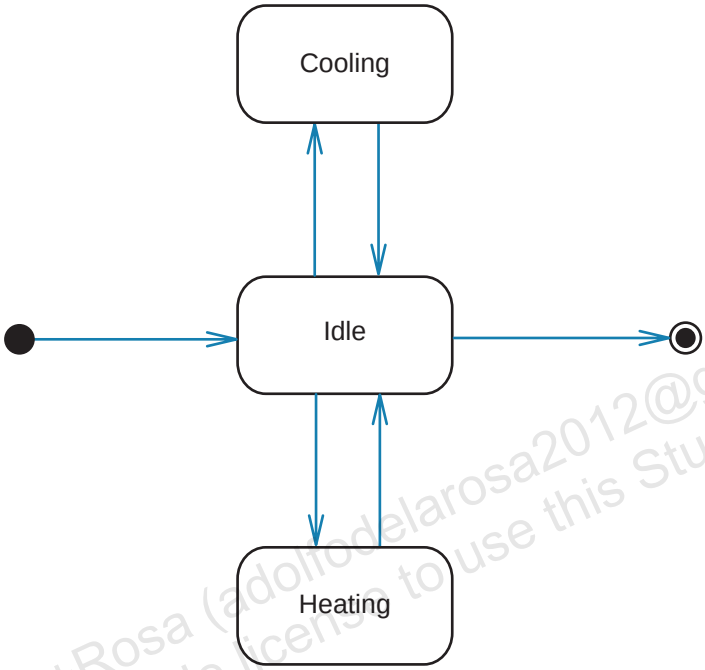


Figure 9-8 Step 3 – Specify the Partial Ordering of States

Step 4 – Specify the Transition Events and Actions

Then specify the triggers, guard conditions, and actions (if any) for each transition. If no trigger is specified, then the transition occurs immediately (unless guarded). If no guard condition is specified, then the transition occurs when the trigger occurs. The guard condition can halt a state transition even if the trigger occurs. If no actions are specified, then the transition occurs without affecting the object (other than changing state). If actions are specified, then these actions act upon the object before the new state is entered. Figure 9-9 illustrates this step of the HVAC state model.

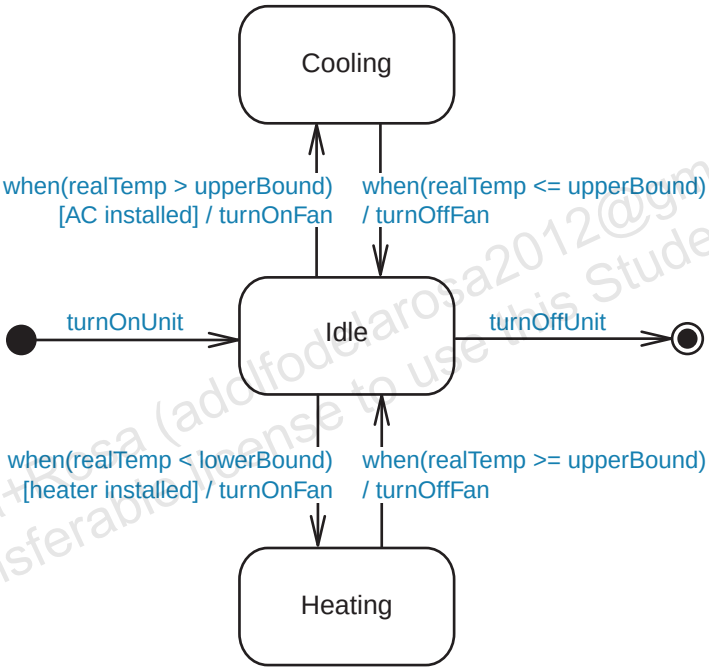


Figure 9-9 Step 4 – Specify the Transition Events and Actions

Step 5 – Specify the Actions Within a State

Finally, specify any events and corresponding actions that occur within each state (if any). For example, the Cooling state indicates that the AC subsystem is turned on after this state is entered, and the AC is turned off after the state is exited. Figure 9-10 illustrates this step of the HVAC state model.

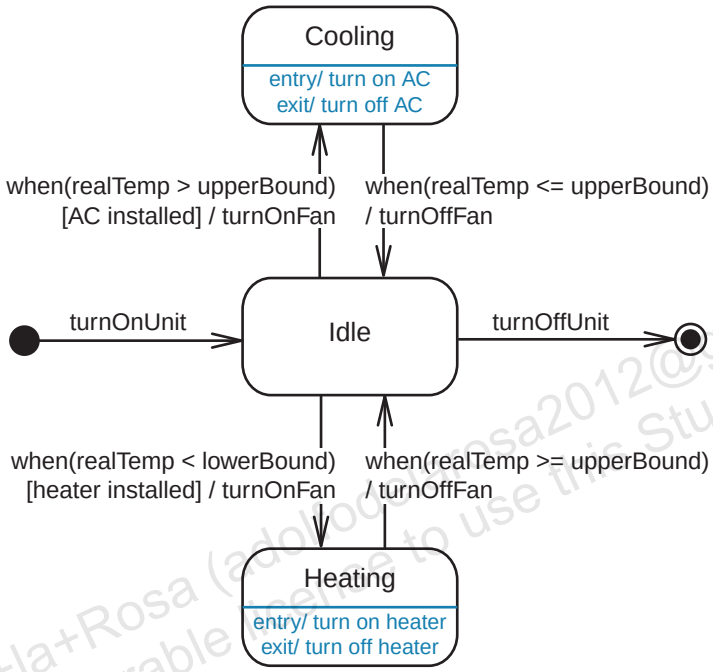


Figure 9-10 Step 5 – Specify the Actions Within a State

After Trigger Event

Events that occur after a period of time are shown by using the *after* trigger event.

To fire a transition after 10 minutes, specify the event on the transition as `after(10 mins).`

This event is often used for timeouts.

Self Transition

A self transition is a state transition with the same state for the source and the destination of the transition. As a result of self transition, the exit and entry internal actions also fire.

Figure 9-11 shows an example Stack in which the push and pop trigger events transition to another state or transition back to the same state.

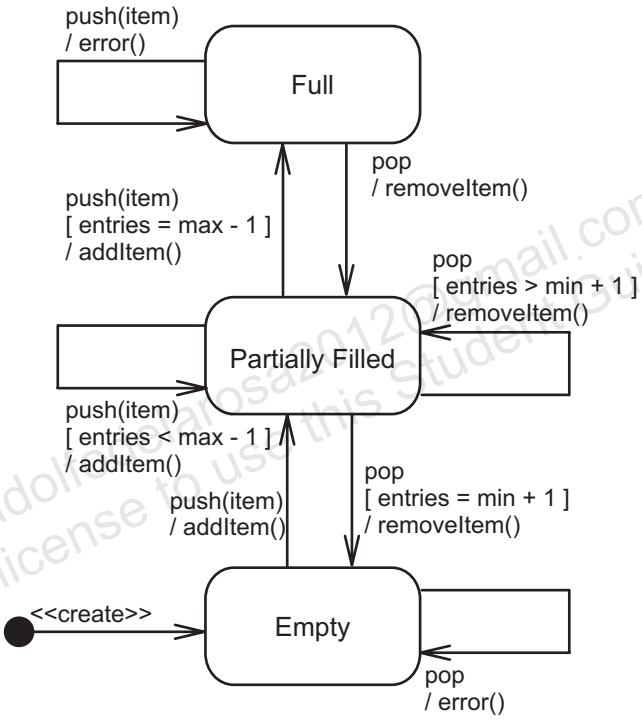


Figure 9-11 Stack Example Showing Self Transition

Junction

In UML, a Junction is used to simplify diagrams by breaking the transition into several fragments, thereby reducing the duplication of trigger events, guards and actions.

Figure 9-12 shows the order of state transitions, order of events, evaluation of guards, and actions.

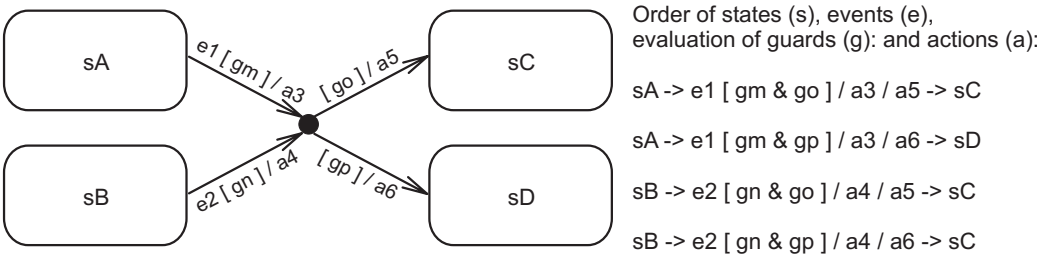
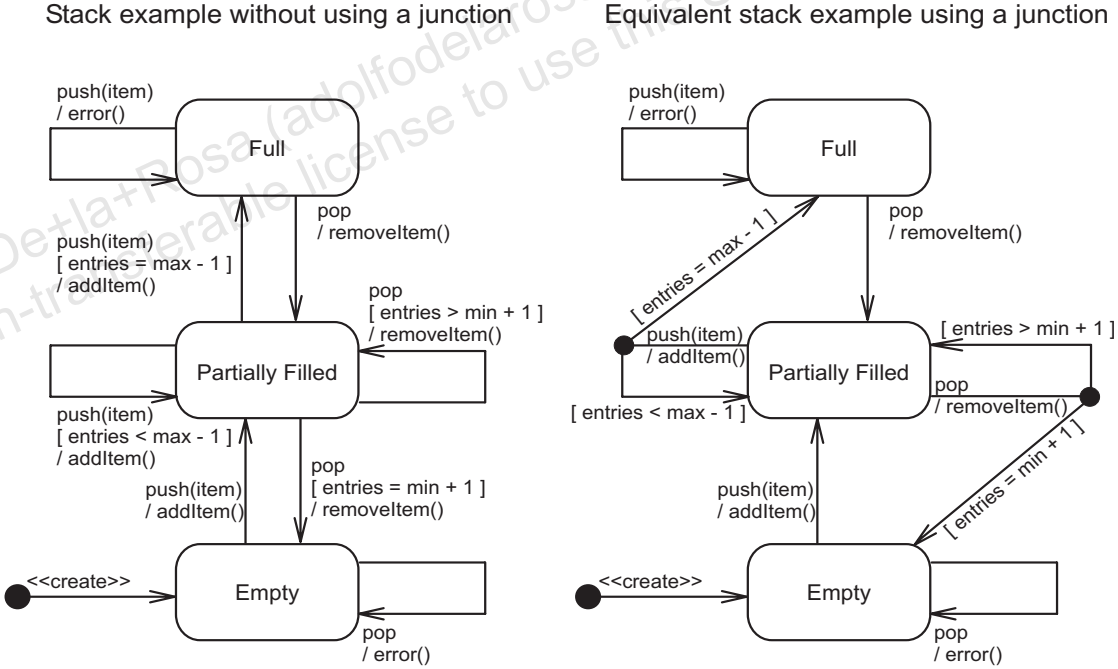


Figure 9-12 Order of Processing by Using Junction

Junctions Example

Figure 9-13 shows two examples of a stack. One example uses the original state transitions notation. The other example shows the advantages of



using a junction in which the number of duplicate trigger events and actions are reduced.

Figure 9-13 Comparing Stacks With Junction and Without Junction

Choice

In UML, a Choice is used to simplify diagrams by breaking the transition into several fragments, thereby reducing the duplication of trigger events, guards, and actions. Choice also enables dynamic evaluation of the guards after the previous transition fragments actions are executed.

Figure 9-14 shows the order of state transitions, order of events, evaluation of guards, and actions.

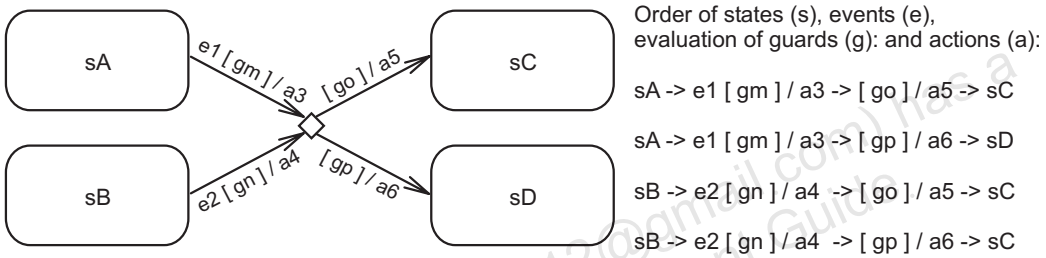


Figure 9-14 Order of Processing by Using Choice

Choice Example

Figure 9-15 compares two examples of a stack. One example uses the Junction state transitions notation and the other example shows the differences by using a choice state transition notation.

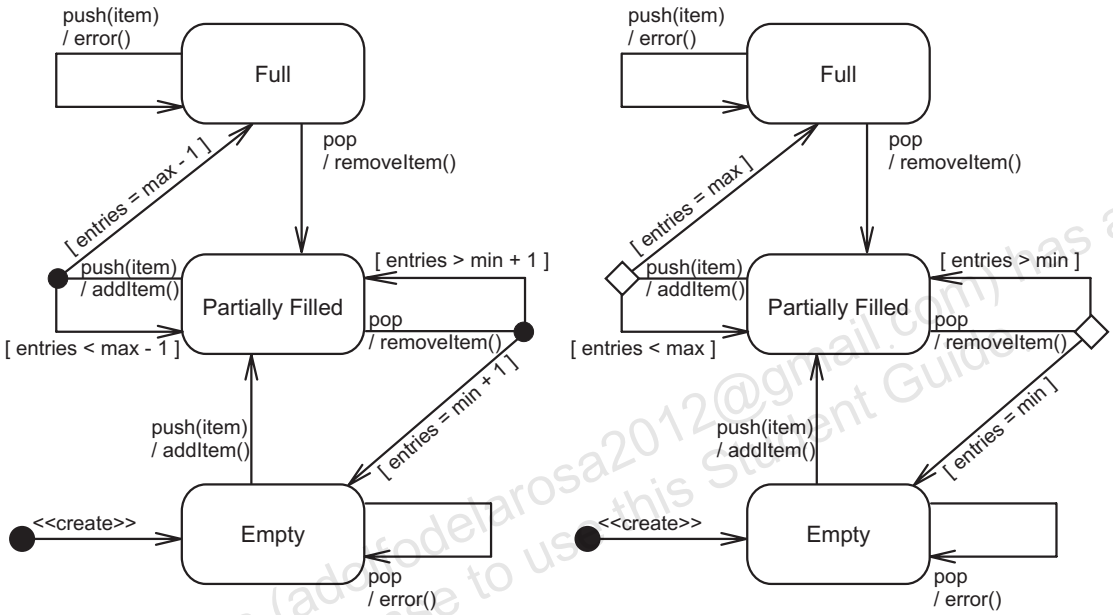


Figure 9-15 Comparing Stacks With Choice and Without Choice

Summary

In this module, you were introduced to the UML State Machine diagram and the concepts behind objects with a complex state. Here are the important concepts:

- A object might have states that define unique behaviors for the object.
- The State Machine diagram provides a mechanism for modeling the states and transitions of an object.