Department of Computer Science

UNIVERSITEIT VAN PRETORIA
UNIVERSITY OF PRETORIA
YUNIBESITHI YA PRETORIA

Tackling Design Patterns
Chapter 10: UML State Diagrams

Contents

10.1 Introduction **2**

10.2 Notational Elements **2**
10.2.1 State Nodes 2
10.2.2 Transition edges 3
10.2.3 Control Nodes 4
10.2.4 Actions 5
10.2.5 Signals 5
10.2.6 Composite States 6

10.3 Examples **6**

10.4 Exercises **7**

References **8**

10.1 Introduction

UML State diagrams are used to describe the behaviour of nontrivial objects. State diagrams are good for describing the behaviour of one object over time and are used to identify object attributes and to refine the behaviour description of an object.

A state is a condition in which an object can be at some point during its lifetime, for some finite period of time [4]. State diagrams describe all the possible states a particular object can get into and how the objects state changes as a result of external events that reach the object [1]. The notation for state diagrams was first introduced by Harel [2], and then adopted by UML.

10.2 Notational Elements

A UML State diagram is a graph in the mathematical sense of the word. It is a diagram consisting of nodes and edges. The nodes can assume a variety of forms each with specific meaning, while the edges are labeled arrows connecting these nodes. In Section 10.2.1 we discuss the basic nodes. The basic nodes are called state nodes. In Section 10.2.2 we discuss the syntax for the edges. They are called transitions. To be able to model complicated transitions special nodes called control nodes are introduced in Section 10.2.3.

10.2.1 State Nodes

There are three kinds of state nodes; initial nodes, state nodes and end nodes.

Figure 1: Initial Node

Figure 1 shows the symbol used to indicate the initial node. It is a filled circle. The initial node is the starting point of a state diagram. It indicates the default state of an object whose behaviour may change over time. A state diagram may at most have one initial node. Although a diagram may be drawn without indicating the starting point it is considered good practice to always have a starting point.

Figure 2: End Node

Figure 2 shows the symbol used to indicate an end node. It is a filled circle with another open circle around it. It indicates where the system terminates. A system may be running infinitely while sometimes changing state. In such case its UML state diagram may have no end nodes. It is also permissible for a state diagram to have many final nodes. Since a state diagram is used to model behaviour that is dependent on events that may or may not happen, it is conceivable that the system may terminate in a variety of ways.

Figure 3: State Node

Figure 3 shows the symbol used to indicate a state. It is a rounded square. In a state diagram each state other than the initial state (initial node) and the final state (end node), should be named. The name of a state is a short descriptive name that can be used to refer to the state. The name of the state in this figure is `Active`. We discuss more detail about states in Section 10.2.4.

10.2.2 Transition edges

Figure 4: Transition Edge

Transition edges in a state diagram are used to indicate transition between states. If an object in a system changes state as a reaction to some event, it is indicated by connecting the current state with a target state with an arrow labeled with the name of the event that triggers such transition. The following detail may be shown on a transition in a UML state diagram. These are all optional i.e. if not required, they may be omitted.

- The event that triggers the transition (text). If the event shown on the transition is detected, the transition will fire and the state of the object will change to the state at the end of the transition. If a transition is shown without an event it triggers immediately after all actions that are associated with entering the state are completed.
- A guard condition that is a prerequisite for transition (text in [ ]). If the guard condition is true the the transition will fire, otherwise it will not fire. When there are more than one transition leaving the same node, the value of the guards my determine which one of them will fire. If no guard condition is shown for a transition, there are no preconditions required for the transition to fire.
- An action that is executed during transition (method name after /). Activities that are executed while moving from one state to the other is shown in the form of a method call on a transition label. Such method call must be preceded by a /. If the transition from one state to another is not associated with some action, no action is specified.

10.2.3 Control Nodes

There are two kinds of control nodes; decision-and-merge nodes to model alternate flows, and fork-and-join nodes to model parallel flows.

Alternate Flows

To model alternate flows, decision-and-merge nodes are used. They are diamonds. They are used both at the beginning and the end of alternate flows. The diamond at the beginning of an alternate flow is called a decision node, while the diamond at the end of the alternate flow is called a merge node. When a decision node is included in a UML State diagram guard conditions are required to indicate the conditions that determine which of the alternative flows will fire.

Figure 5 models a simple heating device. It sensors the current temperature in its sensing state. It remains in sensing state until one of the conditions of the decision node is true. The choice of which alternative flow to follow is determined by the guard conditions. If `[temp < 20]` is true, the top flow will trigger. The device will be turned on as indicated in the action on this transition. The device will remain in the `heating` state until the event `2 min elapsed` occurs. It will then follow through the merge node back to the `sensing` state where it will stay until one of the guard conditions become true. Similarly the device will be turned off when `[temp > 25]` is true whereafter the device will be in `idle` state for a while before returning to `sensing` state.

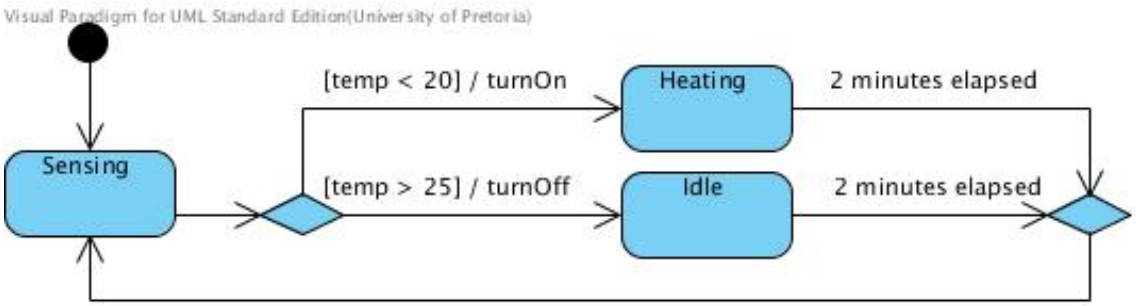

Figure 5: Heating device with alternate flows

Parallel Flows

To model parallel flows, fork-and-join nodes are used. They are heavy vertical or horizontal lines. They are used both at the beginning and the end of parallel flows. The node at the beginning of a number of parallel flows is called a fork node, while the node at the end where the parallel flows meet again is called a join node. When a fork node is included in a UML State diagram, all transitions leaving the fork node fire at the same time creating different threads that execute simultaneously. The join node is used where these parallel threads synchronise. It is also called a synchronisation point. The transition leaving a join node fires only after all threads in the parallel flows meeting at the join have reached the join.

Figure 6 models the stages of a system from implementation through testing to operation. This diagram indicates that the hardware and software testing are executed in parallel. The system will only enter its operation stage after both hardware and software testing are completed.

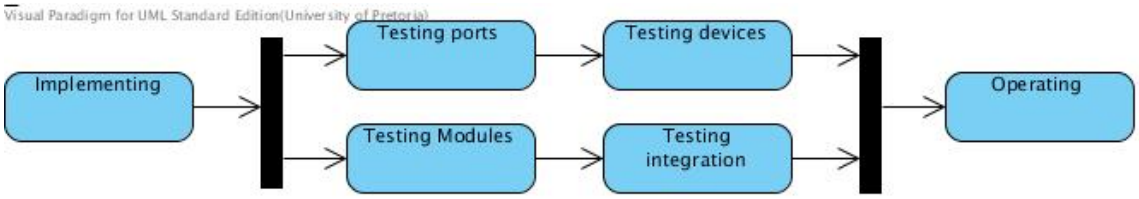

Figure 6: Stages of a system with parallel flows

10.2.4 Actions

Actions can be executed during transition as was explained in Section 10.2.2. It is also possible to indicate actions that are executed while the system is in a state. More often such actions are triggered on entering a state or when leaving a state. Figure 7 shows the syntax for indicating such actions. In this figure the `pickUp` action will execute when the `Receiving` state is reached and the `disconnect` action will execute before a transition to a next state is performed.

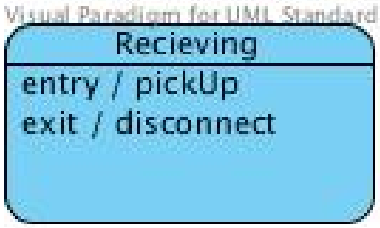

Figure 7: Actions that are executed in a state

10.2.5 Signals

Figure 8: Signal node

In event driven programming events are generated while a system is running. Many of these events are generated by the users of the system, for example when selecting a menu

option or clicking a button. Some of the events can also be generated by actions executed by the system. For example when a timer times-out or when some critical threshold is reached. These events can be modelled using a signal node. Figure 8 shows the syntax for a signal node. When a signal node is reached in a UML State Diagram an event named by the label on the signal node is generated. This event can trigger a transition in any other state in the diagram to fire as a result of this event. When a signal is reached, the event is generated and all transitions in the diagram that is associated with the event will trigger.

10.2.6 Composite States

A UML State Diagram can be nested in a state. The two diagrams in Figure 9 was taken from [5]. They model the same states. In the left diagram the UML State Diagram showing the sub-states of the `Check-PIN` state is shown in detail, while they are hidden in the diagram at the right. Notice how the ∞ symbol us used to indicate that a state is a composite state.

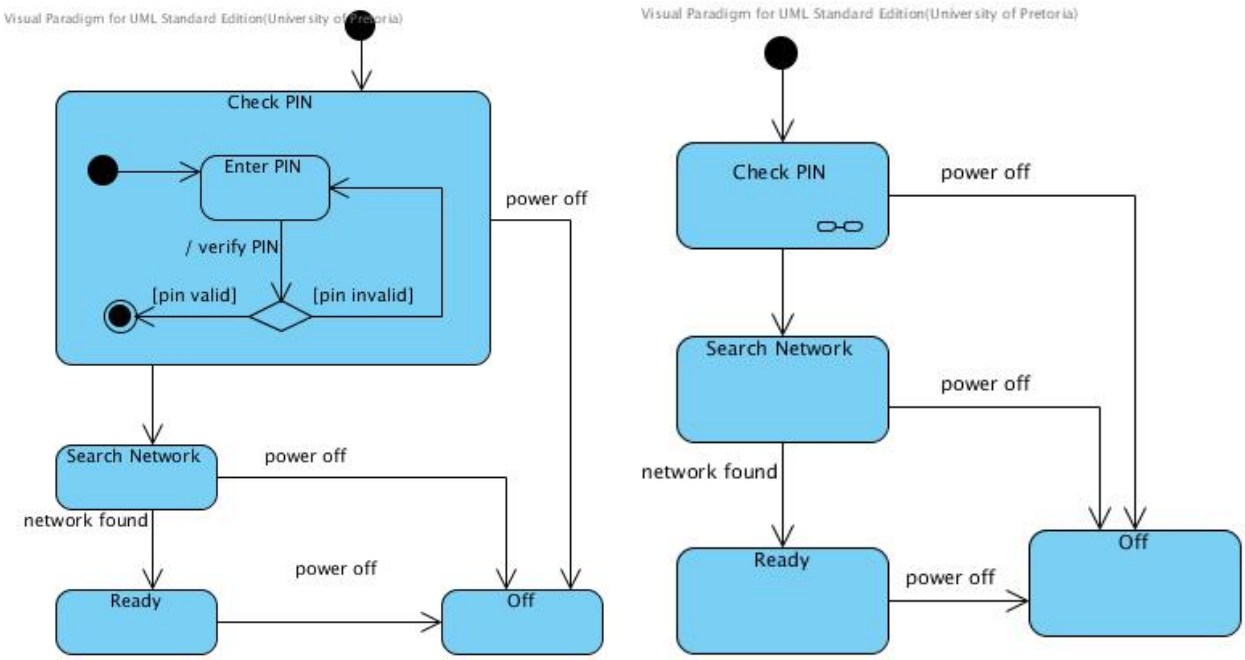

Figure 9: Composite

10.3 Examples

Figure 10 shows a UML State Diagram for a player's turn in Monopoly taken from [6]. It shows all the possible actions and conditions required before taking these actions while making transitions between the states one can be in during one's turn in the Monopoly game. This diagram serves as a good example by itself if you are familiar with the game.

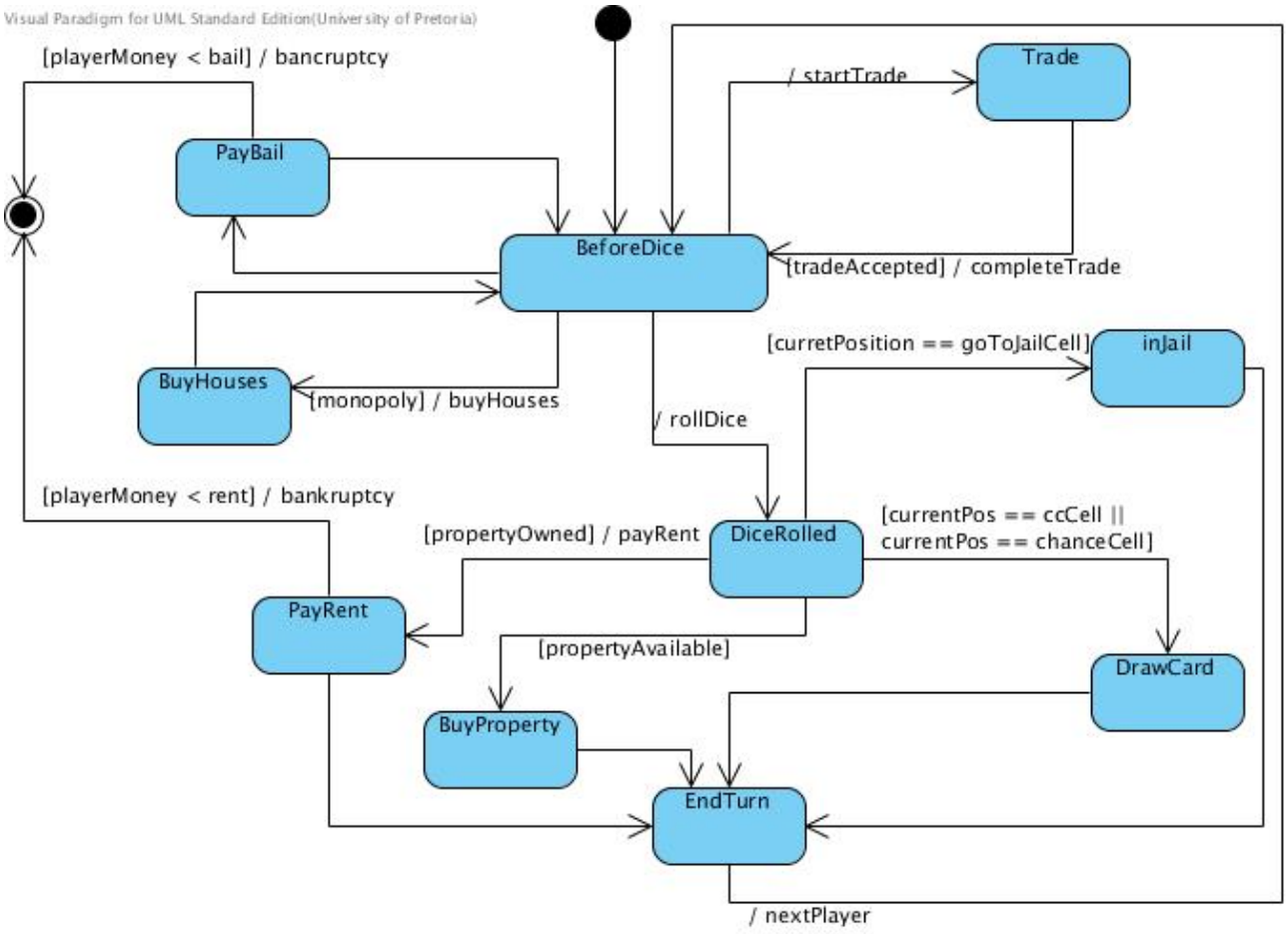

Figure 10: UML State Diagram for a turn in Monopoly

Figure 11 shows a UML State Diagram of a very fancy toaster for toasting bread taken from [3]. It has a timer like most of the common toasters we use in our houses today. In parallel with the timer it uses an on-off cycle similar to the device modelled in Figure 5. Along with all this it features a safety feature that monitors both the absolute colour of the toast as well as the change rate of the colour and will trigger a bomb-out (`Done`-signal) if one of these measures are not OK.

10.4 Exercises

1. Describe all actions and transitions that will execute in the diagram in Figure 10 if a player rolls the dice and lands on a property for which the rent is higher than the value of `playerMoney`.

2. Extend the diagram in Figure 11 model an `Unplugged` state. It should be the initial state.When a `plug-out` event occurs in any of the existing states it should immediately transition to this state. When a `plug-in` event occurs it should transition from `Unplugged` to `Idle` unless it's sensor detects that there is a slice of bread in it. In such a case it should start toasting the bread as if the `start` event had happened.

3. Draw a `UML State Machine` diagram to visualise the states of a torch that is operated with an on-off switch. When the torch is switched on, it shines yellow. It can only be switched off if the switch is turned off while the torch is shining red. The torch will start shining red when it is switched on while it was shining yellow. If the torch is switched off when shining yellow, it start shining white. If it is turned off while shining white, it starts shining yellow.

References

[1] Martin Fowler and Kendall Scott. *UML Distilled: A Brief Guide to the Standard Object Modeling Language*. Addison-Wesley, Reading, Mass, 2003. ISBN 0-321-19368-7.

[2] David Harel. Statecharts: a visual formalism for complex systems. *Science of Computer Programming*, 8(3):231 – 274, 1987.

[3] Robert C. Martin. Uml tutorial: Complex transitions. `http://www.objectmentor.com/resources/articles/cplxtrns.pdf`, 1998. [Online: Accessed 29 June 2011].

[4] Kendall Scott. *UML Explained*. Addison-Wesley, Boston, Massachusetts, 2001.

[5] n.a. Sparx Systems Pty Ltd. Uml 2 state machine diagram, 2001.

[6] Laurie Williams. An introduction to the unified modeling language: A picture is worth a thousand words. `agile.csc.ncsu.edu/SEMaterials/UMLOverview.pdf`, 2004. [Online; accessed 30-June-2011].

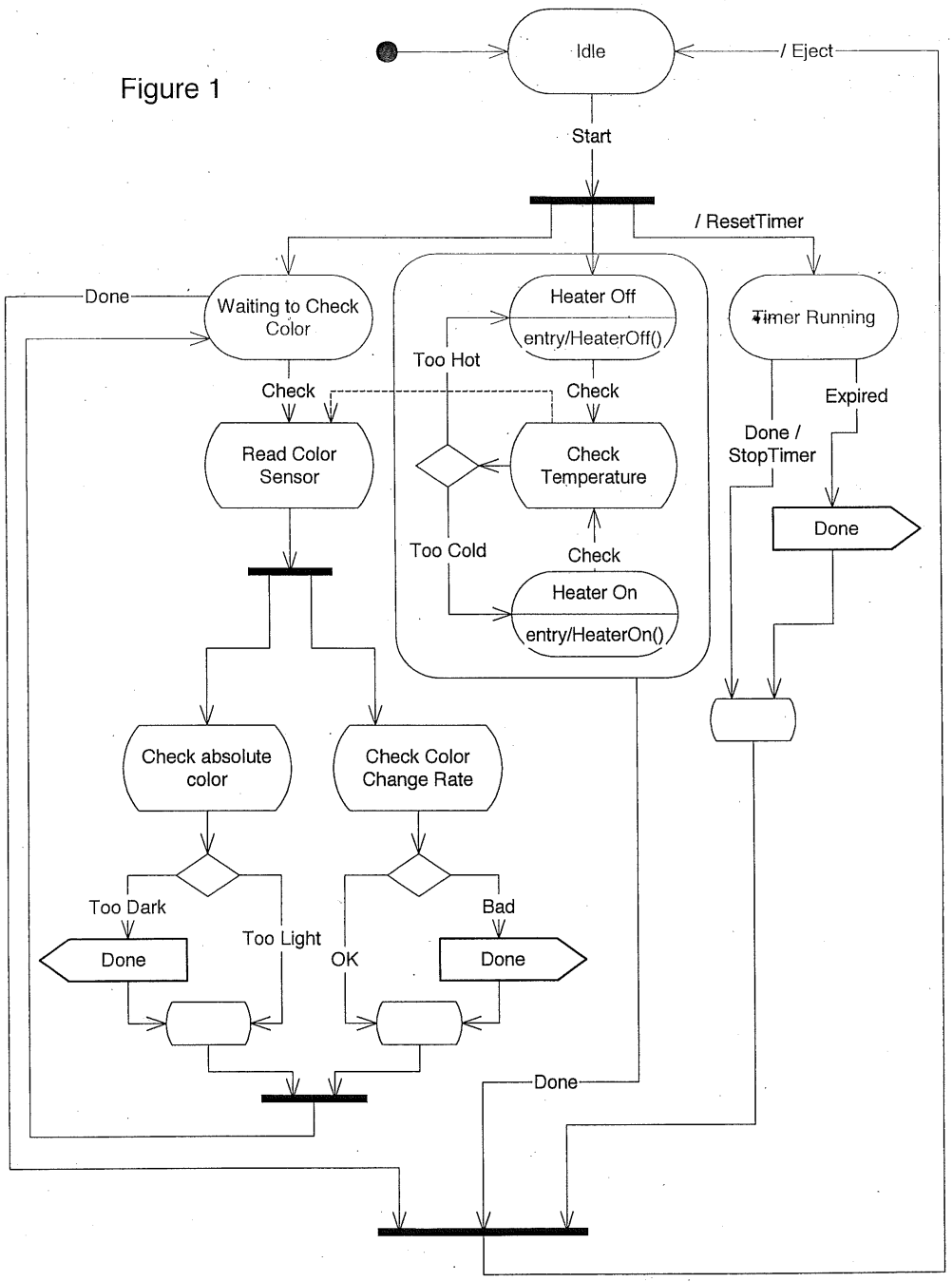

Figure 11: UML State Diagram modeling a toaster