CHAPTER 6 Activity Diagram and State Chart Diagram

6.1 INTRODUCTION TO ACTIVITY DIAGRAM

In object-oriented analysis & design, activity diagrams, in many ways, are the equivalent of flow charts and data flow diagrams (DFDs) from structured development showing flow of control from activity to activity. An activity diagram is typically used for modelling the logic captured by a single use-case or usage scenario. The diagram can also be used to model a specific actor's workflow within the entire system. Activity diagrams can also be used independent of use-cases for other purposes such as to model business process of a system, to model detailed logic of business rules, etc.

An activity diagram shows all potential sequence flows in an activity. Activity diagrams are used for simple and perceptive illustration of what happens in a workflow, what activities can be done in parallel, and whether there are alternative paths through the workflow.

An activity diagram is a dynamic diagram that shows the activity and the event that causes the object to be in the particular state. It shows the workflow from a start point to the finish point detailing the many decision paths that exist in the progression of events contained in the activity. Activity diagrams may be used to detail situations where parallel processing may occur in the execution of some activities.

Activity diagrams are useful for business modelling where they are used for detailing the processes involved in business activities and business and operational workflows of a system.

6.2 ELEMENTS OF ACTIVITY DIAGRAM AND THEIR NOTATION

6.2.1 Initial State

An initial state is an element that explicitly shows the beginning of a workflow on an activity diagram. It is the point at which reading of the activity diagram begins. Because an activity diagram shows a sequence of actions, it must indicate the starting point of the sequence using the initial state element. The initial state is drawn as a solid circle with an optional name or label as shown in Figure 6.1.

FIGURE 6.1 Begin admission process—initial state.

6.2.2 Final State

A final state is an element that explicitly shows the end of a workflow on an activity diagram. Unlike initial state, there can be multiple final states in an activity diagram to indicate termination of specific branches of the workflow. The final state is drawn as a filled circle inside a larger unfilled circle called *bull's-eye*, with an optional name or label as shown in Figure 6.2.

FIGURE 6.2 End of admission process—final state.

6.2.3 Action/Activity

Action states represent the non-interruptible actions of objects which cannot be further decomposed and takes insignificant execution time. The activity states can be further decomposed, their activity being represented by other activity diagrams. Also activity states are not atomic, so they may be interrupted and takes some time to complete.

There is no notational distinction between action and activity states, except that an activity state may have additional parts, such as entry and exit action. Action/Activity is drawn as a rounded rectangle with a name or description of action/activity as shown in Figure 6.3.

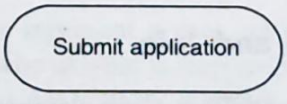

FIGURE 6.3 Activity in admission process.

6.2.4 Transitions

A transition element connects the various elements of the activity diagram. Typically the transition element represents the workflow between two or more actions/activities or other elements of the activity diagram. It is drawn as a solid line with an open arrowhead.

The transition element can have an optional label enclosed in square brackets called *guard condition* or simply *guard*. Guard conditions are used to define the conditional logic that controls the flow of control, and the specific criteria that must be met for a transition to occur as shown in Figure 6.4.

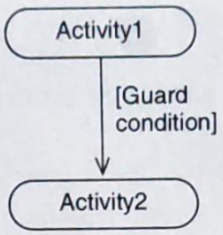

FIGURE 6.4 Transitions.

6.2.5 Decisions

A decision element typically has one incoming transition and two or more outgoing transitions based upon the outcome of guard conditions from the previous element.

A decision box is drawn as a diamond shape usually without a name or label (no need for name/label because the guard conditions usually imply the reason for the decision). Transitions that leave a decision model element often have guard conditions as shown in Figure 6.5.

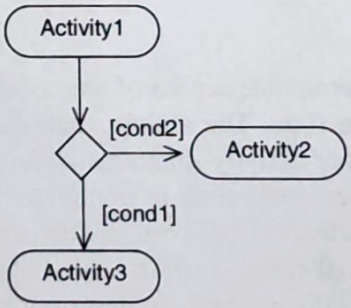

FIGURE 6.5 Decision.

6.2.6 Synchronization, Fork and Join

A synchronization element allows modelling of simultaneous workflows in an activity diagram, i.e. parallel activities. Synchronizations visually define forks and joins that represent a parallel workflow or execution and is drawn as a horizontal or vertical bar with no name or label as illustrated in Figure 6.6(a).

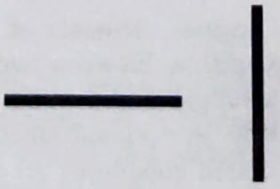

FIGURE 6.6(a) Synchronization.

A fork is a kind of synchronization element that facilitates the modelling of simultaneous workflows in an activity. A fork identifies where a single flow of control divides into two or

more separate, but simultaneous flows. It is drawn as a bar with one transition going into it and two or more transitions leaving it as in Figure 6.6(b).

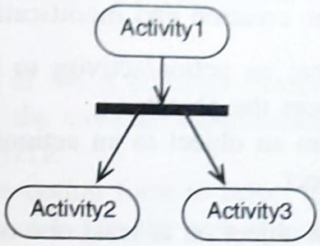

FIGURE 6.6(b) Fork.

A join is another kind of synchronization element that facilitates the modelling of simultaneous workflows in an activity. It identifies where two or more simultaneous flows of control unite into a single flow of control. A join is drawn as a bar with two or more transitions going into it and one transition leaving it as in Figure 6.6(c).

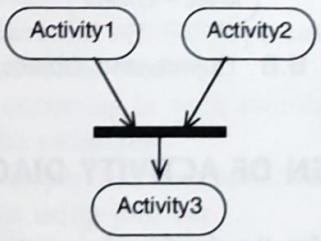

FIGURE 6.6(c) Join.

6.2.7 Swimlanes

A swimlane is an element that can represent a user, an organizational unit, or a role in an activity diagram. Swimlanes depict who or what is responsible for carrying out specific activities. They enable grouping of actions on the activity diagram performed by the same actor or by a single thread. An action transition can take place between two swimlanes. Swimlanes are drawn as vertical columns or horizontal rows with a name associated with each swimlane as shown in Figure 6.7.

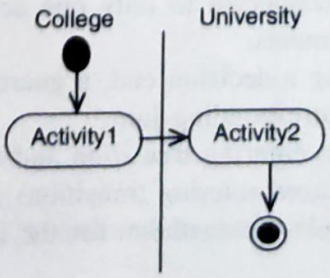

FIGURE 6.7 Swimlanes.

6.2.8 Objects and Object Flows

An object flow is a path along which objects or data can pass. An object is shown as a rectangle. Object flow refers to the creation and modification of objects by activities.

- An object flow arrow from an action/activity to an object means that the action/activity creates or influences the object.
- An object flow arrow from an object to an action/activity indicates that the action/activity state uses the object.

An object flow must have an object on at least one of its ends as shown in Figure 6.8.

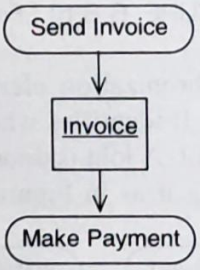

FIGURE 6.8 Object and Object Flows.

6.3 GUIDELINES FOR DESIGN OF ACTIVITY DIAGRAM

The following are the guidelines for the design of an activity diagram.

1. One activity diagram should be drawn for each use-case. No two use-cases flow should be mixed into a single activity diagram. An activity diagram can also be drawn for a system or an actor as a whole.
2. Always an activity diagram should reflect business flow rather than system flow.
3. Only one initial state element should be drawn in an activity diagram.
4. In case of swimlanes, the initial state should be placed in the first swimlane.
5. As a thumb rule, it should be avoided to have more than five swimlanes in a single activity diagram.
6. An initial state should be connected directly to "Action/Activity" element of the activity diagram and not to any other element.
7. The initial state must be connected to only one action/activity element and not to multiple action/activity elements.
8. On every transition, leaving a decision end, a guard condition must be specified.
9. Every fork must have a corresponding join.
10. A fork must have only one entering transition and two or more leaving transitions.
11. A join must have two or more entering transitions and only one leaving transition.
12. It is always better to provide a name/label for the initial and final states.

6.4 PROBLEM STATEMENT: MCA ADMISSION SYSTEM

MCA admission procedure as controlled by Directorate of Technical Education (DTE) is as follows:

1. DTE advertises the date of MCA entrance examination.
2. Student has to apply for the entrance examination.
3. Results are declared by DTE.
4. Student has to fill up the option form to select the college of his/her choice.
5. DTE displays the allotment list in the web site and intimation to all colleges.
6. Students should report the allotted colleges and complete the admission procedure.

6.4.1 Analysis of MCA Admission System

For drawing an activity diagram for the whole system we,

1. Find out swimlanes if any. To find swimlanes, see if we can span some activities over different organizational units/places.
2. Find out in which swimlane the admission process begins and where it ends. Those will become the initial and final states.
3. Then, identify activities occurring in each swimlane. Arrange activities in sequence flow spanning over all the swimlanes.
4. Identify conditional flow or parallel flow of activities. Parallel flow of activities must converge at a single point using join bar.
5. During the activities are performed, if any document is generated or used, take it as an object and show the object flow.

Swimlanes identified for admission process are shown in Figure 6.9.

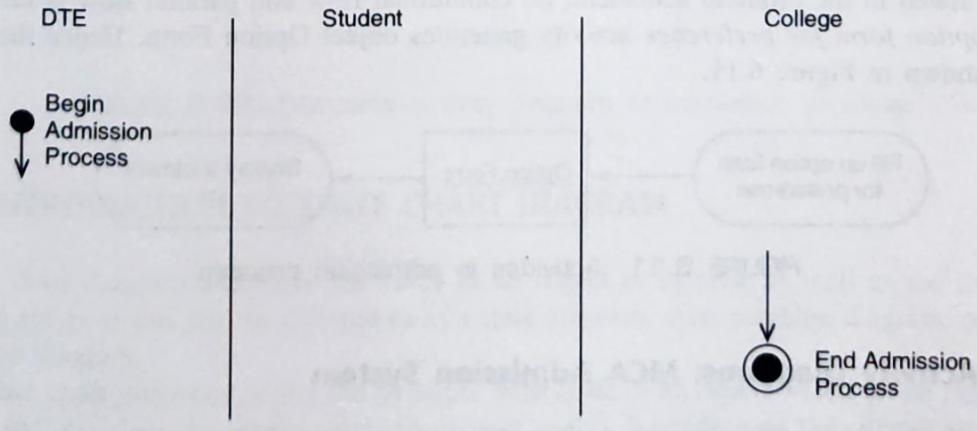

FIGURE 6.9 Swimlanes in admission process.

Activities identified in admission process are shown in Figure 6.10.

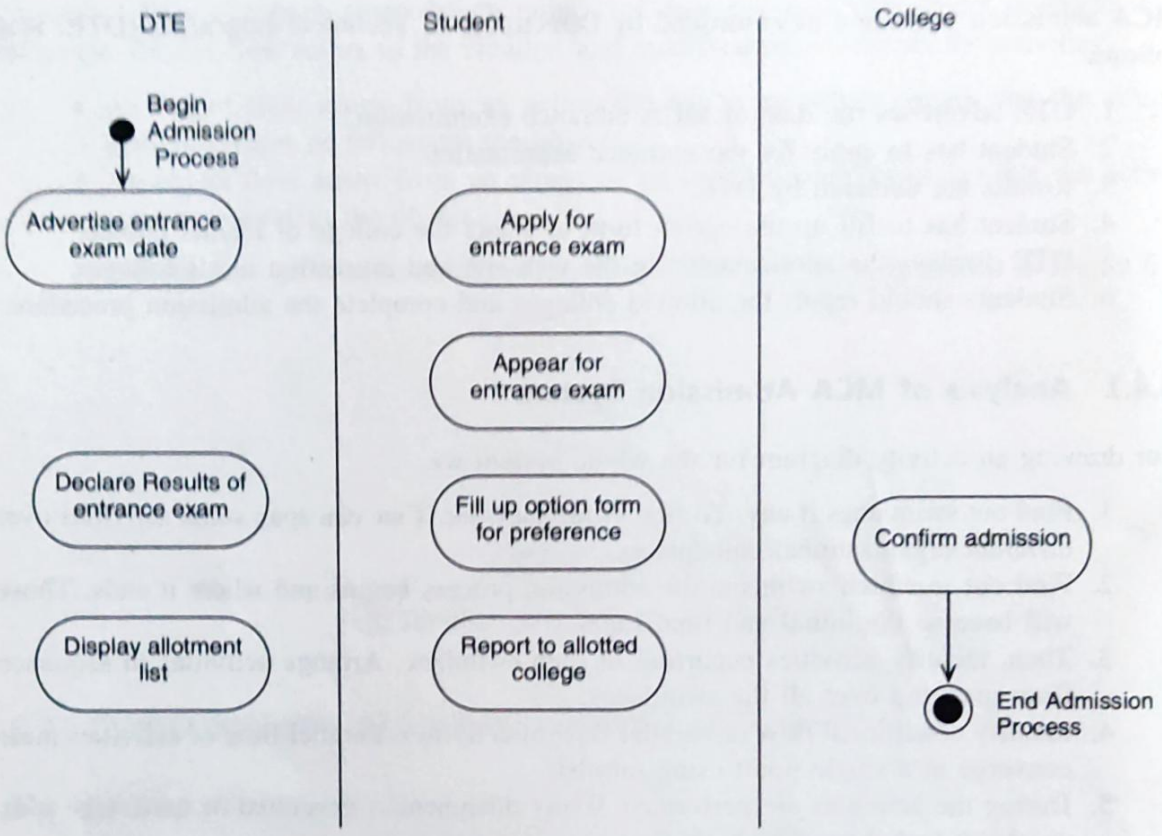

FIGURE 6.10 Activities in admission process.

As stated in the problem statement, no conditional flow and parallel flow is identified. *Fill up option form for preference* activity generates object Option Form. Hence the object flow is shown in Figure 6.11.

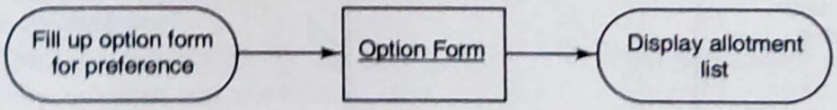

FIGURE 6.11 Activities in admission process.

6.4.2 Activity Diagrams: MCA Admission System

The complete activity diagram with the transitions is as given in Figure 6.12.

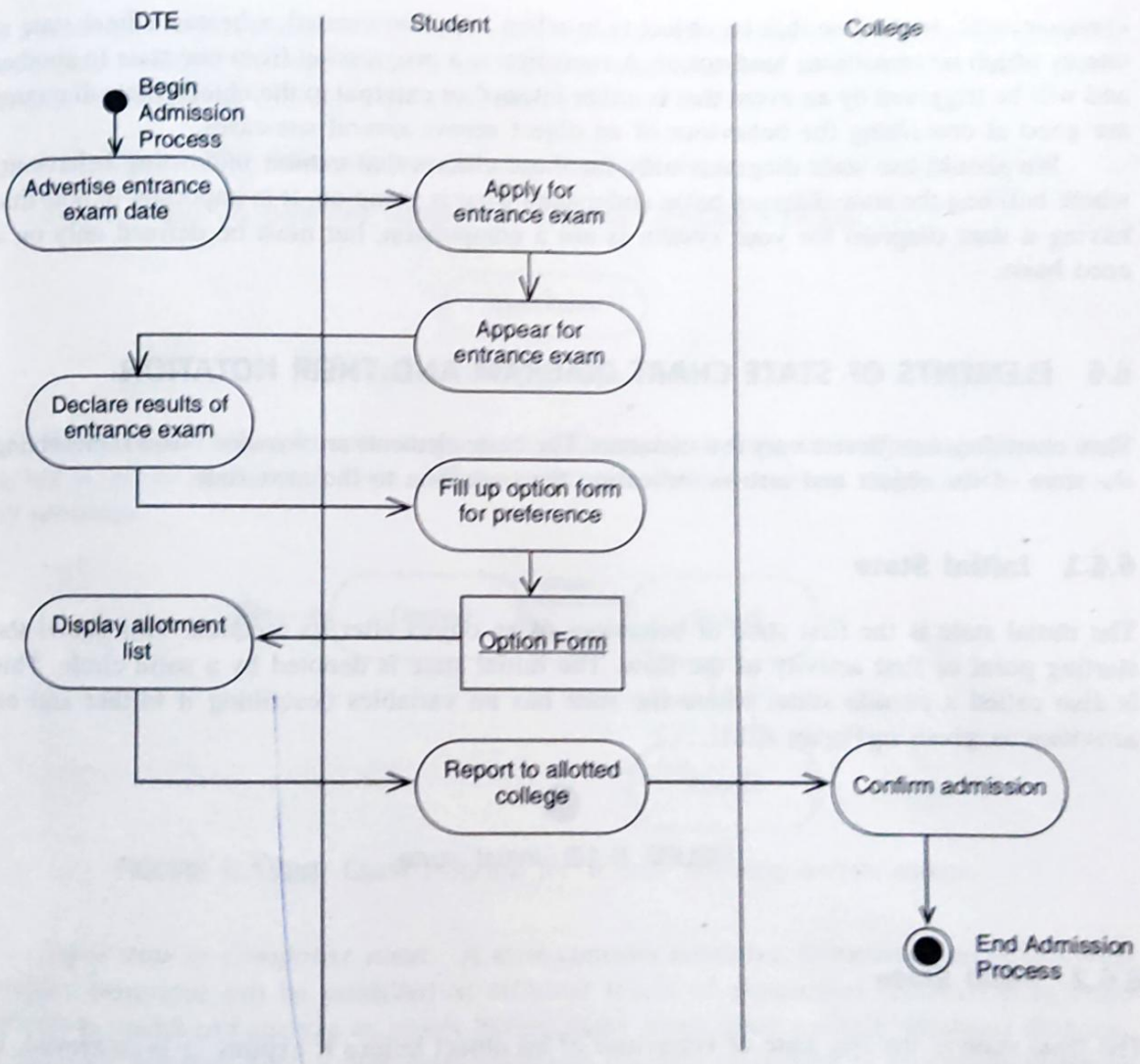

FIGURE 6.12 Complete activity diagram of admission process.

6.5 INTRODUCTION TO STATE CHART DIAGRAM

A state chart diagram describes the states of an object or system, as well as the transitions between states. It can also be referred to as a state diagram, state machine diagram, or a state-transition diagram.

State chart diagrams depict the dynamic behaviour of an object based on its response to events, showing how the object reacts to various events depending on the current state that it is in. A state chart diagram is drawn to explore the complex behaviour of a class, actor, subsystem, or component.

A state represents a stage in the behaviour pattern of an object, and like UML activity diagrams it is possible to have initial states and final states. An initial state, also called a

creation state, is the one that an object is in when it is first created, whereas a final state is one in which no transitions lead out of. A transition is a progression from one state to another and will be triggered by an event that is either internal or external to the object. State diagrams are good at describing the behaviour of an object across several use-cases.

We should use state diagrams only for those classes that exhibit interesting behaviour, where building the state diagram helps understand what is going on. It is important to note that having a state diagram for your system is not a compulsion, but must be defined only on a need basis.

6.6 ELEMENTS OF STATE CHART DIAGRAM AND THEIR NOTATION

State chart diagrams have a very few elements. The basic elements are rounded boxes representing the state of the object and arrows indicating the transition to the next state.

6.6.1 Initial State

The initial state is the first state of behaviour of an object after its creation. This shows the starting point or first activity of the flow. The initial state is denoted by a solid circle. This is also called a *pseudo state*, where the state has no variables describing it further and no activities as given in Figure 6.13.

FIGURE 6.13 Initial state.

6.6.2 Final State

The final state is the last state of behaviour of an object before it expires or is destroyed. It indicates the end of the state diagram as shown by a bull's-eye symbol. A final state is another example of a pseudo state, because it does not have any variable or action described as shown in Figure 6.14.

FIGURE 6.14 Final state.

6.6.3 State

A state represents a visible mode of behaviour of an object that persists for a period of time. Activities can run within a state. Transformations occur between states rather than within a state. For representing the condition of an object at an instant of time we use state, denoted

by a rectangle with rounded corners and compartments. Each state on a state chart diagram can contain multiple internal actions. An action is best described as a task that takes place within a state such as *On entry*, *On exit*, *Do*.

Types of states: The following are different types of states:

Simple state: A state that contains no *substates* [Figure 6.15(a)].

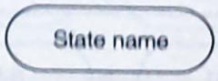

FIGURE 6.15(a) Simple state.

For example, a simple state diagram for a Door object with states *Opened*, *Closed* and *Locked* is shown in Figure 6.15(b). All the three states of the door are simple states without any substates.

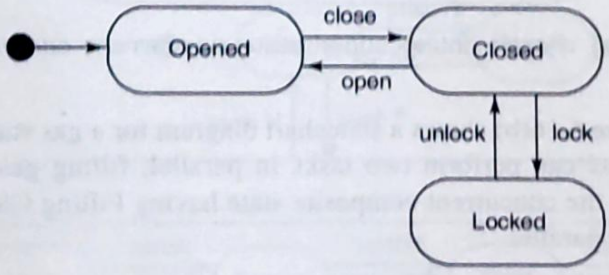

FIGURE 6.15(b) State diagram for a door showing simple states.

Super state or *Composite state:* A state contains *substates*. Sometimes, an object with complex behaviour can be modelled at different levels of abstraction (details). It is often difficult to model and analyze an object having many states using a single statechart diagram. In that case, we may draw a high level statechart diagram consisting of composite states or super states and other diagrams to further elaborate the internal states or substates inside individual composite states. For each super state or composite state, we can draw its nested states (internal states or substates) and their transitions between them.

States and transitions inside the super state are connected to the outside via the super state only. A super state is used when many transitions lead to a certain state. Transition arrows can be attached to the super state from the inside and the outside. By nesting a relatively autonomous part of a diagram in a super state, you can reduce the number of transitions and make the diagram easier to read.

A composite state or super state is composed of more than one sequential or concurrent substate and is called a *concurrent composite state* or a *sequential composite state* depending upon the kind of substate it has.

Concurrent composite state: The concurrent composite state is useful when a given object has sets of independent behaviours. However, there should not be too many concurrent

sets of behaviour occurring in a single object. If an object has several complicated concurrent statechart diagrams, the object should be split into separate objects. If a concurrent composite state is active, then one of the nested states from each concurrent state is also active. A state may be divided into regions containing substates that exist and execute concurrently by a dashed line [Figure 6.16(a)].

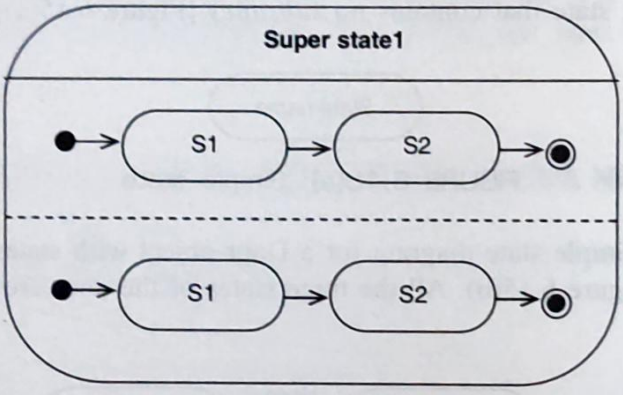

FIGURE 6.16(a) Representing super state concurrent composite state.

For example, Figure 6.16(b) shows a statechart diagram for a gas station where on arrival for filling gas, attendants can perform two tasks in parallel, filling gas as well as washing windshield. InService is the concurrent composite state having Filling Gas Tank and Washing Windshield substates in parallel.

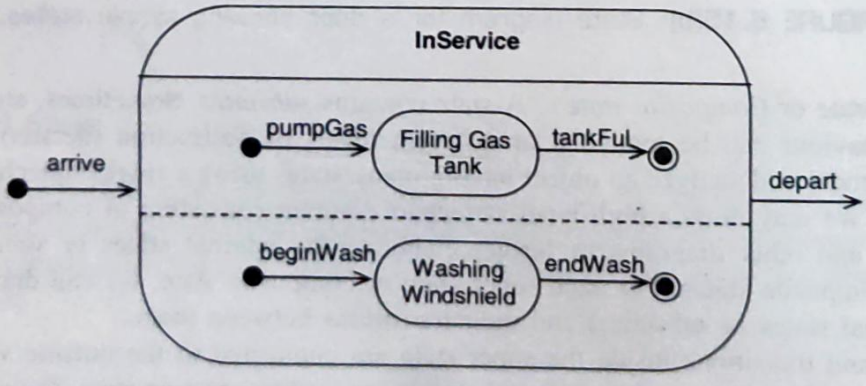

FIGURE 6.16(b) Example of concurrent composite state—InService.

Sequential composite state: If a sequential composite state is active, then exactly one of its substates is active [Figure 6.16(c)].

For example, Figure 6.16(d) shows the state chart diagram for car transmission. Transmission has three states: *Neutral, Reverse* and *Forward*. The forward state is a sequential composite state showing substates *First, Second, Third and Fourth*.

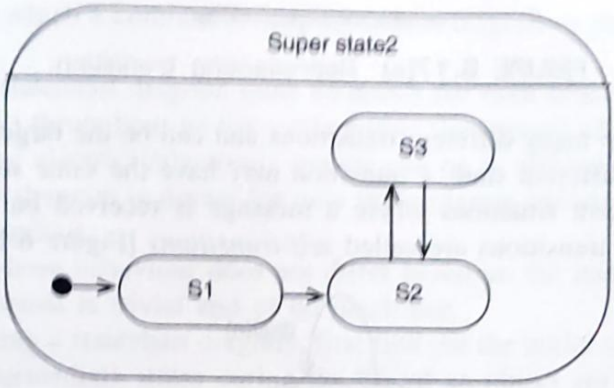

FIGURE 6.16(c) Representing superstate—sequential composite state.

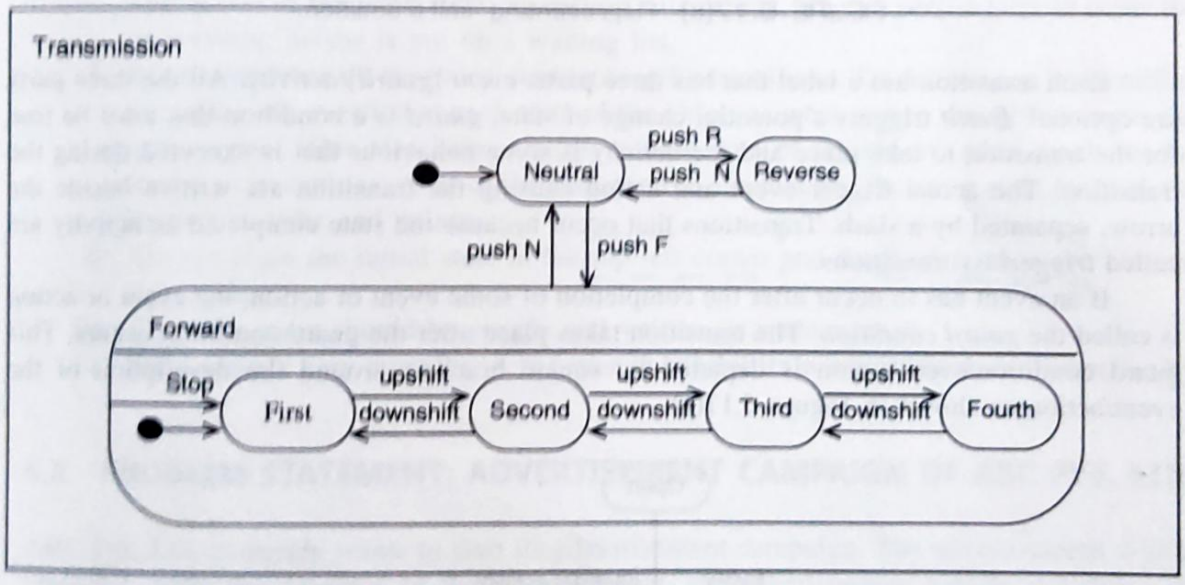

FIGURE 6.16(d) Example of sequential composite state transmission.

Substate: A state that is nested inside another state is called a substate. Substates allow state diagrams to show different levels of abstraction. Substates may be sequential or concurrent. Substates may be nested to any level. In Figure 6.16(d) *First, Second, Third, Fourth, Neutral* and *Reverse* are representing substates.

6.6.4 Transitions

A state transition is a relationship between two states that indicates when an object transit from one state to other state once certain conditions are met.

A transition to the same state is recursive transition which goes into the same state. An arrow (optional) indicates the transition of an object from one state to the other as given in Figure 6.17(a).

FIGURE 6.17(a) Representing transition.

A state can source many different transitions and can be the target of many transitions. Instead of going to a different state, a transition may have the same source and target state. Such transitions represent situations where a message is received but does not result in a change of state. These transitions are called *self-transitions* [Figure 6.17(b)].

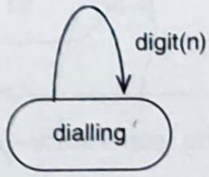

FIGURE 6.17(b) Representing self-transition.

Each transition has a label that has three parts: *event* [*guard*]/*activity*. All the three parts are optional. *Event* triggers a potential change of state, *guard* is a condition that must be true for the transition to take place and the *activity* is some behaviour that is executed during the transition. The actual trigger event and action causing the transition are written beside the arrow, separated by a slash. Transitions that occur because the state completed an activity are called *triggerless* transitions.

If an event has to occur after the completion of some event or action, the event or action is called the *guard condition*. The transition takes place after the guard condition occurs. This guard condition/event/action is depicted by square brackets around the description of the event/action as shown in Figure 6.17(c).

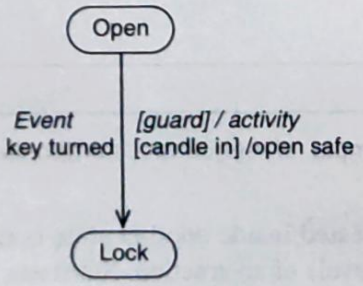

FIGURE 6.17(c) Transition with a guard condition.

6.7 GUIDELINES FOR DESIGN OF STATECHART DIAGRAM

The guidelines for the design of a state chart diagram are as follows:

1. A statechart diagram may be either one shot life cycle type or continuous loop type.
2. Objects for which one shot life-cycle statechart diagram is drawn have fixed life—from START state till END state, e.g. a chess game which has the START state presenting a new game opens till the END state presenting a win/loss of game.

3. Objects for which a continuous loop statechart diagram is drawn, do not have an end state.
4. A separate statechart diagram must be drawn for each object, showing various states of that object throughout its life cycle within the context of the system, e.g. a switch object in any electrical/electronic system can be in either ON or OFF state.
5. A statechart diagram is drawn for only those classes showing interesting or complex internal behaviour.
6. The class whose behaviour does not differ based on the state, the statechart diagram for those classes is trivial and of no much use.
7. While drawing a statechart diagram, first find out the initial state and final state. After this, the intermediate states during the life of an object are identified. The states of an object can be identified by looking at the boundary values of its attributes, e.g. in case of Railway reservation, when all the seats are reserved, reservation becomes full. *Full* is one of the valid states as various rules apply, when a person tries to make the reservation, he/she is put on a waiting list.
8. After identifying all the states, start looking for transitions. Transitions can be identified by asking the question at each state of the object as what makes that object to get out of that state. This also helps identify a new state of the object. Also check for the recursive transitions. A recursive transition is one that has the same state for both the starting and the end points.
9. Always place the initial state in the top-left corner and the final state in the bottom-right corner.
10. The state name should be simple and written in present tense.
11. If the state is very complex, it is better to exhibit its substates to resolve complexity.

6.8 PROBLEM STATEMENT: ADVERTISEMENT CAMPAIGN OF ABC PVT. LTD.

ABC Pvt. Ltd. company wants to start its advertisement campaign. The advertisement will be prepared. After the approval of the advertisement, the advertisement will be scheduled for publication. The advertisement will be published after scheduling is done.

6.8.1 Analysis of Advertisement Campaign of ABC Pvt. Ltd.

In this problem statement, the object for which a statechart diagram should be drawn is:

Advertisement campaign

Since the *Advertisement campaign* has fixed states from start till end, the statechart diagram will be of type one shot life cycle statechart diagram. Hence there will be one initial state and one or more final state.

Initial state: Figure 6.18(a) shows the initial state of the advertisement compain process.

FIGURE 6.18(a) Begin advertisement campaign process.

Final state: Figure 6.18(b) shows the final state of the compain process.

FIGURE 6.18(b) End advertisement campaign process.

Intermediate states: Figure 6.19(a) shows the intermediate states of the process.

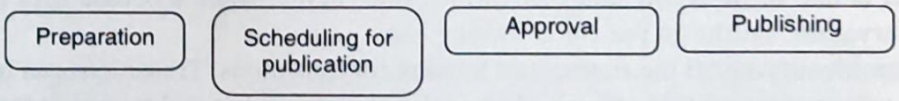

FIGURE 6.19(a) Intermediate states—advertisement campaign.

Figure 6.19(b) shows all the transitions of the advertisement compaign process.

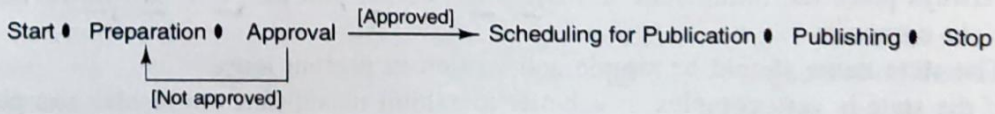

FIGURE 6.19(b) Transition—advertisement campaign: scheduling for publication and publishing.

- All the transitions in the above process are triggerless transitions which occur on completion of activity during the previous state.
- The guard condition to enter from *Approval* state into *Scheduling for Publication* state is Advertisement is approved/Not approved. No trigger is required to transit from the state.

6.8.2 State Chart Diagram for Advertisement Campaign of ABC Pvt. Ltd.

The complete state transition diagram for the *advertisement campaign* object is shown in Figure 6.20.

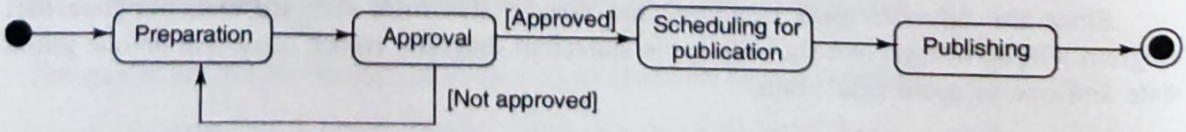

FIGURE 6.20 State chart diagram—advertisement campaign.

EXERCISES

1. What does a state-chart diagram represent?
2. What is the purpose of an activity diagram?
3. What are forks and joins in an activity diagram?
4. What is the difference between Action and Activity states?
5. Explain the super state in a statechart diagram with an example.
6. Propose a statechart diagram for a library book states (i.e., available, borrowed, to-be-returned, etc.).
7. Draw a statechart diagram capturing the state-transitions in a vending machine (i.e. waiting, inserting, choose-item, get-item, get-changes, etc.).
8. Draw a statechart diagram capturing the state-transitions in an ATM machine.
9. Draw a statechart diagram for a DVD player.
10. Develop an activity diagram based on the following narrative.

    The purpose of the Open Access Insurance System is to provide automotive insurance to car owners. Initially, prospective customers fill out an insurance application, which provides information about the customer and his or her vehicles. This information is sent to an agent, who sends it to various insurance companies to get quotes for insurance. When the responses return, the agent then determines the best policy for the type and level of coverage desired and gives the customer a copy of the insurance policy proposal and quote.

11. Create an activity diagram based on the following narrative.

    The purchasing department handles purchase requests from other departments in the company. People in the company who initiate the original purchase request are the *customers* of the purchasing department. A case worker within the purchasing department receives that request and monitors it until it is ordered and received. Case workers process the requests for purchasing products under $1,500, write a purchase order, and then send it to the approved vendor. Purchase requests over $1,500 must first be sent out for a bid from the vendor that supplies the product. When the bids return, the case worker selects one bid. Then, the case worker writes a purchase order and sends it to the approved vendor.