UNIT 2 DYNAMIC MODELING

Structure **Page Nos.**

2.0	Introduction	20
2.1	Objectives	20
2.2	Events	20
2.3	State and State Diagram	21
2.4	Elements of a State Diagram	24
2.5	Advanced Concepts in Dynamic Modeling	25
2.6	Concurrency	26
2.7	A Dynamic Model	27
2.8	Summary	28
2.9	Solutions/Answers	29

2.0 INTRODUCTION

You must have observed that whenever something is done, some action is triggered. From daily life you can see that when you press a bell button, a ring tone is produced. That means some **event** has taken place. The dynamic model covers this aspect of the systems.

The dynamic model shows the **time-dependent** behavior of the system and the objects in it. Events can be defined as "something happened at a point of time". The dynamic model is important for interactive systems. The logical correctness of events depends on the sequences of interactions, or sequence of events.

You can understand a system by first looking at its static structure, the structure of its objects, and their relationships over time. Aspects of a system that are concerned with time and changes are the **dynamic models**. Control describes the sequences of operations that occur in response to external stimuli without consideration of what the **operations do**, **what they operate on**, or **how they are implemented**.

The major dynamic modeling concepts are **events**, which represent external stimuli, and **states**, which represent values of objects. The state diagram is a standard computer science concept (a graphical representation of finite state machines). Emphasis is on the use of events and states to specify control rather than as algebraic constructs. In this Unit, we will discuss the basic concepts of dynamic modeling which will cover events and states. We will also cover state diagram and concept of concurrency.

2.1 OBJECTIVES

After going through this unit, you should be able to:

- explain events, and transition;
- design state diagrams;
- explain the elements of state diagram;
- use advanced concepts in dynamic modeling;
- explain, concurrency; and
- represent dynamic model of systems.

2.2 EVENTS

You can understand an event as **some action oriented result,** such as **mouse click.** Whenever you will click on a mouse, the appropriate action takes place.

You may observe that an event has no **specific time period** duration. You can click on a mouse and keep it pressed as long as you want. So, as far as events are concerned, nothing is instantaneous. An event is simply **an occurrence**. An event is actually a one way transmission of information from one object to another, but some times it may be that an event that occurs on a single object and changes the state of that object.

Two events can take place at the same time, one after the other, or both the events independently of each other and occurring simultaneously. For example, like two trains can depart at the same time for two different places, or two can depart from the same place, but one after the other. It means that the two events can be **independent as well as dependent on each other**.

Two events which are unrelated and occur at the same time are known as concurrent events. They have no effect on each other. You will not find any **particular order** between the two events, because they can occur in any order. In a distributed system, you will notice **concurrent events** and **activities**.

An **object** sending an **event to another object** may expect **a reply**, but the reply will be a **separate event** of the second object. So, you may see conversations between two objects as a combination of two or more events.

Event Classes: Every event is a **unique occurrence**; event class is a name to **indicate common structure** and **behavior**. Some events are simple signals, but most event classes have attributes indicating the information they convey. For example, events like train departs which has the attributes train number, class, city, etc. It is not necessary that all the attributes of objects contribute to attributes of events.

Here, you must note that the **time** at which the **event occurs** is an **implicit attribute** of all events.

Some events convey information in the form of **data** from one object to another. Sometime it may be that some classes of events only **signal** that something has occurred, while other classes of events **convey** data values. The data values conveyed by an event are **its attributes**; it implies that the value of data objects involved in events.

Event class name (attributes)

Sometimes event refers to event instance, or event class.

Mouse button clicked (left click, location)

Digit dialed (digit)

Phone receiver lifted.

Events include error conditions as well as normal occurrences.

Scenario and Event traces

A scenario is seen as a ***sequence of events that occurs during one particular execution of a system***. As far as the scope of a scenario is concerned, it varies. It may include **all events** in the system, or only some events from some selected objects involved in the event. Example of a scenario can be the historical records of executing a system, or a thought experiment of executing a proposed system.

Now, let us discuss states, and state diagram.

2.3 STATE AND STATE DIAGRAM

The state of an object is decided by the current values associated with the attributes of that object.

State

A state is **a condition** during the life of an **object,** or an interaction, during which, it satisfies some condition, performs some action, or waits for some event. An object remains in a state for a finite (non-instantaneous) time.

Actions are **atomic** and **non-interruptible**. A state may correspond to ongoing activity. Such activity is expressed as a **nested state machine**. Alternately, ongoing activity may be represented by a pair of actions, one that starts the activity on **entry to the state** and one that **terminates the activity on exit from the state**. So you can see that activities are the agents that are responsible for the change in state. Also, a state has its duration, and most of the time, a state is associated with some continuous activity**.**

You must see that a state should have **initial states** and **final states**. A transition to the enclosing state represents a transition to the initial state. A transition to a final state represents the completion of activity in the enclosing region. Completion of activity in all concurrent regions represents completion of activity by the enclosing state and triggers a "**completion of activity event**" on the enclosing state. Completion of the outermost state of an object corresponds to its **death**.

Notation

A state is shown as a **rectangle with rounded corners**. It may have one or more compartments. The compartments are **all optional**. They are as follows:

- Name compartment, holds the (optional) name of the state as a string. States without names are "**anonymous**" and are **all distinct**. It is undesirable to show the same named state twice in the same diagram.
- Initial state is shown by a solid circle.
- Final state is shown by a bull's eye.

Creating a State Diagram

Let us consider the scenario of travelling from station A to station B by the Bus Stand.

Following is the example of a **state diagram** of such scenario. It represents the normal flow. It does not show the **substates** for this scenario.

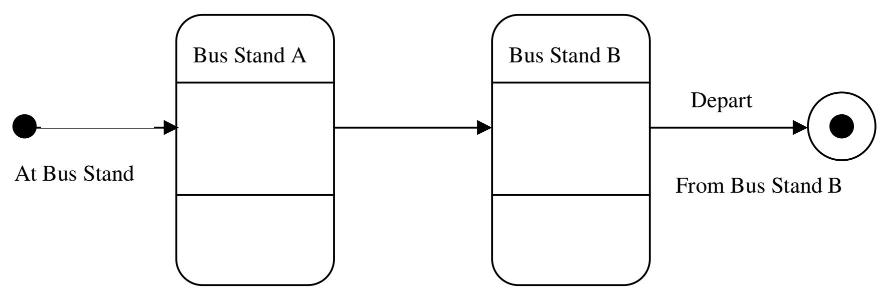


Figure 1: An example of flow in a state diagram

State chart diagrams

Objects have behaviors and state. The state of an object depends on its current activity, or condition. A **state chart diagram** shows the possible states of the object and the **transitions** that cause a change in state.

This diagram in *Figure 2* models the login part of an online banking system. Logging in consists of entering a valid social security number and personal id number, then submitting the information for validation.

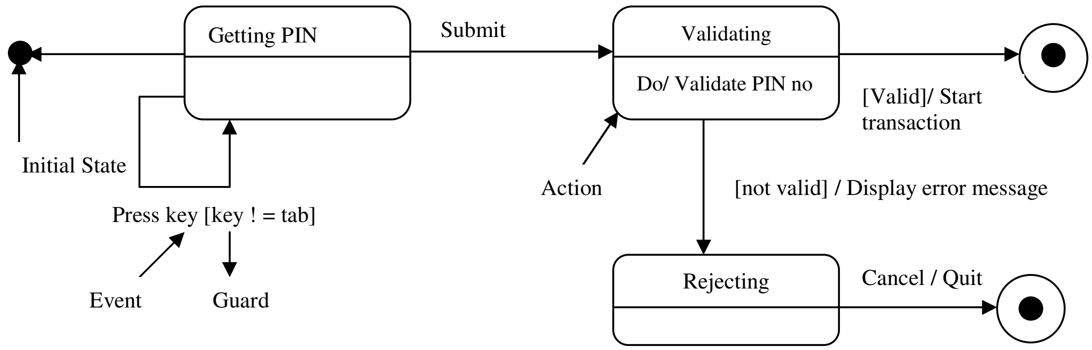


Figure 2: State chart diagram of login

Logging in can be factored into four non-overlapping states: **Getting PIN**, **Validating**, and **Rejecting**. From each state comes a complete set of **transitions** that determine the subsequent state.

States are rounded rectangles. Transitions are arrows from one state to another. Events or conditions that trigger transitions are written beside the arrows. Our diagram has self-transition, on **Getting PIN**.

The initial state (black circle) is a dummy to start the action. Final states are also dummy states that terminate the action.

The action that occurs as a result of an event or condition is expressed in the form action. While in its **Validating** state, the object does not wait for an outside event to trigger a transition. Instead, it performs an activity. The result of that activity determines its subsequent state.

Now, you can see that a statechart diagram shows the sequences of states that an object or an interaction goes through during its life in response to received stimuli, together with its responses and actions. Or, in other words, you can say that:

The state machine is a graph of states and transitions that describes the response of an object of a given class to the receipt of outside stimuli. A state machine is attached to a class or a method.

A statechart diagram represents a state machine. The states are represented by state symbols, and the transitions are represented by arrows connecting the state symbols. States may also contain sub diagrams by physical containment and tiling.

☞ Check Your Progress 1

1) What is a state chart diagram?

……………………………………………………………………………………

……………………………………………………………………………………

2) What is a UML state diagram?

……………………………………………………………………………………

……………………………………………………………………………………

3) Draw a state diagram for a mobile phone.

……………………………………………………………………………………

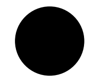
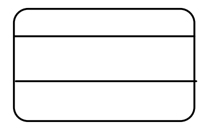
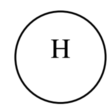
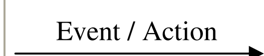
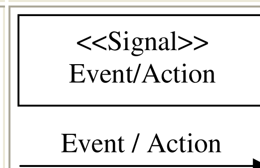
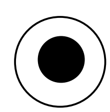
……………………………………………………………………………………

Now, let us discuss the basic components of a state diagram.

2.4 ELEMENTS OF A STATE DIAGRAM

We have seen different symbols used their meaning is a state diagram. Table1also explain about different State diagrams symbols.

Table 1: State Diagram Symbols

Elements and its Description	Symbol
Initial State: This shows the starting point or first activity of the flow. It is denoted by a solid circle. This is also called a **pseudo state**, where the state has no variables describing its further and no activities, to be done.	
State: Represents the state of an object at an instant of time. In a state diagram, there will be multiples of such symbols, one for each state of the object, denoted by a rectangle with rounded corners and compartments (such as a class with rounded corners to denote an object).	
Transition: An arrow indicating the object to transition from one state to the other. The actual trigger event and action causing the transition are written beside the arrow, separated by a slash. Transitions that occur because the state has completed an activity are called "triggerless" transitions.	
History States: A flow may require that the object go into a trance, or wait state, and on the occurrence of a certain event, go back to the state it was in when it went into a wait state — its last active state. This is shown in a State diagram with the help of a letter ***H enclosed within a circle***.	
Event and Action: A trigger that causes a transition to occur is called as an event or action. Every transition need not occur due to the occurrence of an event or action directly related to the state that transitioned from one state to another. As described above, an event/action is written above a transition that it causes.	
Signal: When an event causes a **message/trigger** to be sent to a state that causes the transition; then, that message sent by the event is called a **signal**.	
Final State: The end of the state diagram is shown by a **bull's eye** symbol, also called a final state. A final state is another example of a pseudo state because it does not have any variable or action described.	

Note: Changes in the system that occur, such as a background thread while the main process is running, are called "**substates**". Even though it affects the main state, a substate is not shown as a part of the main state. Hence, it is depicted as contained within the main state flow.

2.5 ADVANCED CONCEPTS IN DYNAMIC MODELING

Now let us look into advanced concepts in Dynamic Modeling. Entry and exit actions are part of every dynamic model. Let us see how they are performed.

Entry and Exit Actions

The following special actions have the same form, but represent reserved words that cannot be used for event names:

'Entry' '/' ***action-expression***: An atomic action performed on entry to the state.
'Exit' '/' ***action-expression:*** An atomic action performed on exit from the state.

Action expressions may use attributes and links of the owning object and parameters of incoming transitions (if they appear on all incoming transitions).

The following keyword represents the invocation of a nested state machine:
'**do**' '/' ***machine-name (argument-list).***

The *machine-name* must be the name of a state machine that has an initial and final state. If the nested machine has parameters, then the argument list must match correctly. When this state is entered, after any entry action, then execution of the nested state machine begins with its initial state.

Example

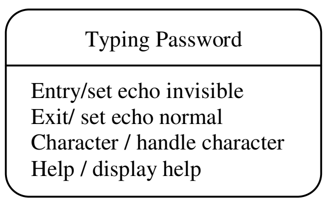


Figure 3: Entry-exit Action

The internal transition compartment holds a list of internal actions or activities performed in response to events received while the object is in the state, without changing state.

The format to represent this is:

event-name argument-list '['guard-condition']' '/' action-expression

Each event name 'or **pseudo-event name'** may appear at most once in a single state.

You can see what happens when 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 (in other words, in the form of a **Boolean expression**).

☞ Check Your Progress 2

1) What is a guard condition? Explain it with an example.

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

2) What are two special events?

……………………………………………………………………………………

……………………………………………………………………………………..

……………………………………………………………………………………

3) What is a self-transition?

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

……………………………………………………………………………………

Now, let us discuss the concept of concurrent object.

2.6 CONCURRENCY

You are already familiar with the term concurrent lines, which goes without affecting other operations. Similarly, when in a system **objects** can change state independently, they are termed **concurrent objects**.

In a dynamic model, some systems are described as a set of concurrent objects, each with its own state and state diagram.

An expansion of a state into concurrent substates is shown by **tiling the graphic region** of the state using dashed lines to divide it into **subregions**. Each subregion is a **concurrent substate**. Each subregion may have an optional name, and must contain a nested state diagram with disjoined states.

Composite States

Now you can say that **a state can be decomposed using *and*-relationships into concurrent substates or using or-relationships into mutually exclusive disjoint substates**. A given state may only be refined in one of these two ways. Its substates may be refined in the same way or the other way.

A newly-created object starts in its initial state. The event that creates the object may be used to trigger a transition from the initial state symbol. An object that transitions to its outermost final state ceases to exist.

An **expansion of a state** shows its **fine structure**. In addition to the (optional) name and internal transition compartments, the state may have an additional compartment that contains a region holding a nested diagram. For convenience and appearance, the text compartments may be shrunk horizontally within the graphic region.

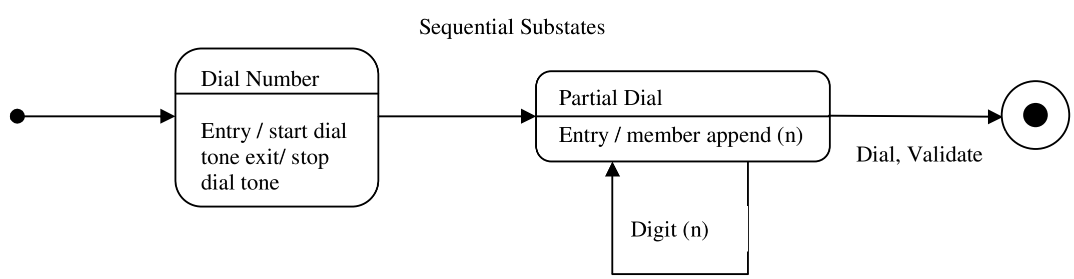


Figure 4: States Sequence

In *Figure 4*, you can see that **dial a number process** state is further divided into its sequential substrates such as, when it is **entering number state** then the state can be named as "**Partial Dial**" in which the user is still entering the number the action is "**append in digits**" then next state will **validate the number and so on**.

A state diagram for an assembly is a collection of **state diagrams**, one for each **component**. Aggregation means **concurrency**. Aggregation is the "**and-relationship**", you will see, it is the combined states of all component diagrams. For example, the state of a Car as an aggregation of component states: the Ignition, Transmission, Accelerator, and Brake. Each component state also has states. The state of the car includes one substate from each component. Each component undergoes transititions in parallel with all others.

Semantics

An event is a noteworthy occurrence. For practical purposes in **state diagrams**, it is an **occurrence** that may **trigger a state transition**. Events may be of several kinds (not necessarily mutually exclusive): The event occurs whenever the value of the expression changes from false to true. **Note** that this is different from a guard condition: A guard condition is evaluated *once* whenever its **event fires**; if it is false then the transition does not occur and the event is lost. Guarded transitions for one object can depend on another object being in a given state.

2.7 A DYNAMIC MODEL

Now you are familiar with **events and their occurring time.** The *dynamic model* describes those aspects of the system concerned with the **sequencing of operations and time - events** that cause state changes, sequences of events, states that define the context for events and the organization of events and states. The dynamic model captures control information without regard **for what the operations act on** or how they are implemented.

The dynamic model is represented graphically by **state diagrams**. A state corresponds to the interval between two events received by an object, and describes the “value” of the object for that **time period**. **A state is an abstraction of an object’s attribute values and links**, where sets of values are grouped together into a state according to properties that affect the general behavior of the object. Each state diagram shows the state and event sequences permitted in a system for one object class. State diagrams also refer to other models: actions correspond to functions in the functional model; events correspond to operations on objects in the object model.

The state diagram should adhere to OMT’s notation and exploit the capabilities of OMT, such as **transition guards**, **actions** and **activities**, **nesting** (state and event generalization), and **concurrency**.

Here is the transition diagram for a digital watch.

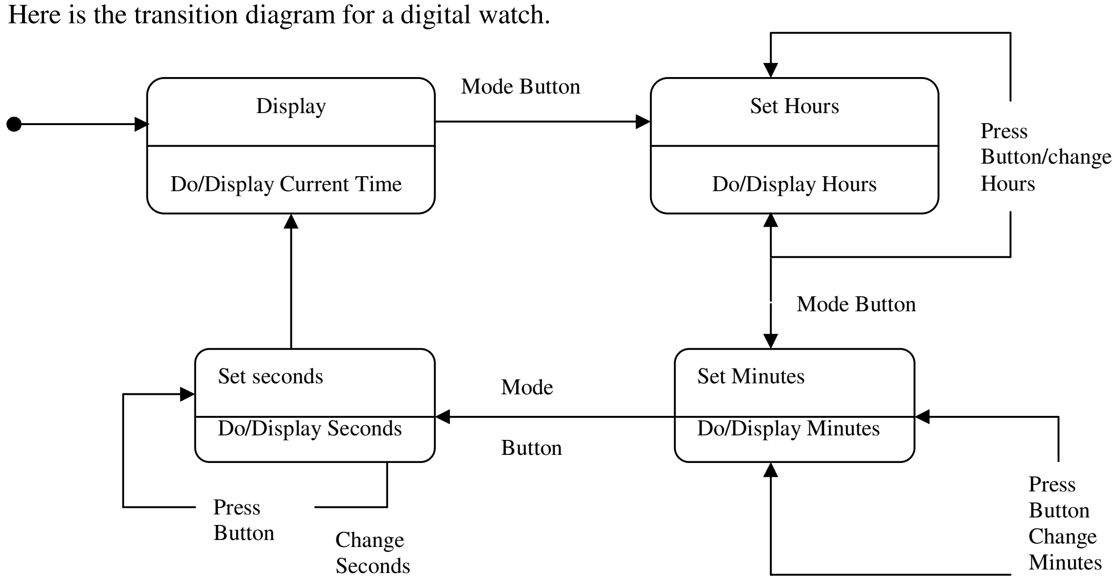


Figure 5: State Transition diagram for digital watch

In *Figure 5,* you can see that the state diagram of a digital watch is given. Where user wants to set Hours set Minutes and followed by setting seconds.

☞ Check Your Progress 3

1) Give a Concurrent substates diagram for classroom and exam held.

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

2) Describe Dynamic Model.

………………………………………………………………………………

………………………………………………………………………………

3) Give a sample of a Dynamic Model.

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

4) Show, with an example, that relatively high-level transitions result when the system or program is stimulated by outside events.

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

………………………………………………………………………………

2.8 SUMMARY

The dynamic model is the model which represents the control information: the sequence of events, states and operations that occur within a system of objects. It has scenarios to occur with meaningful constraints.

An event is triggered by an instantaneous action. One kind of action is sending an event to another object. **External event** also known as a system event is caused by something outside our system boundary. **Internal event** is caused by something inside our system boundary.

States may be "nested." A nested state usually indicates a functional decomposition within the "super" state. The term *macro-state* is often used for the super state. The macro-state may be composed of multiple *micro-states*.

The basic notion is that the system is always in **one state**, and never in more than **one state** (at any given level). The system remains in that state until a transition is triggered by an **external event**. Transitions take no time to occur. There is no time in which the system is not in one of the defined states.

State diagrams must be created for **state-dependent objects** with complex behavior like **Use Cases, Stateful session, Systems, Windows, Controllers, Transactions**, devices, and role mutators.

Actions are associated with transitions and are considered to be processes that occur quickly, and are not interrupted. The syntax for a transition label has three parts, all of which are optional: **Event [Guard] / Action**.

2.9 SOLUTIONS/ANSWERS

Check Your Progress 1

1) State diagrams (State Chart Diagram) describe all the possible states that a particular object can get into, and how the object's state changes as result of events that reach the object. It states all possible states and transitions.

2) The UML state diagram illustrates the events and states of an object and the behavior of an object in reaction to an event.

3)

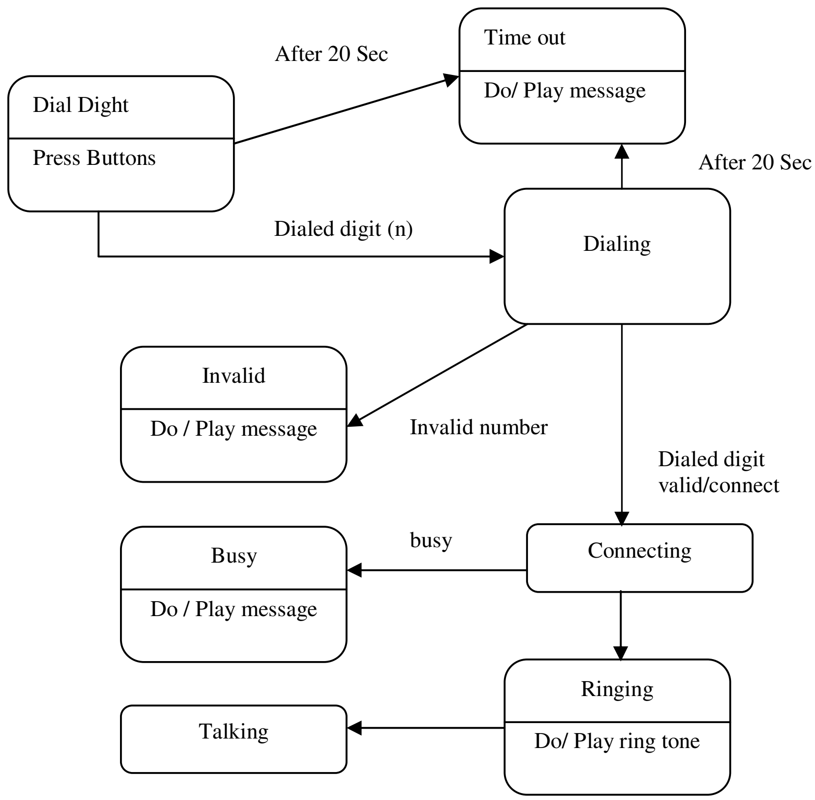


Figure 6: State Diagram for a Mobile

Check Your Progress 2

1) The *guard-condition* is a Boolean expression written in terms of parameters of the triggering event and attributes and links of the object that owns the state machine. The guard condition may also involve tests of concurrent states of the current machine (or explicitly designated states of some reachable object); for example, "**in** State1" or "**not in** State2". State names may be fully qualified by the nested states that contain them, yielding path names of the form "State1: State2::State3"; this may be used in case the same state name occurs in different composite state regions of the overall machine.

2) There are two special events "**entry**" and "**exit**". Any action that is marked as linked to the entry event is executed whenever the given state is entered via transition. The action associated with the **exit event** is executed whenever the state is left via transition.

3) If there is a transition that goes back to the same state, it is called "**self-transition**." With a trigger action the **exit action would be executed first**, then the transition's action and finally **the entry action**. If the state has an associated activity as well, that activity is executed after the **entry action**.

Check Your Progress 3

1)

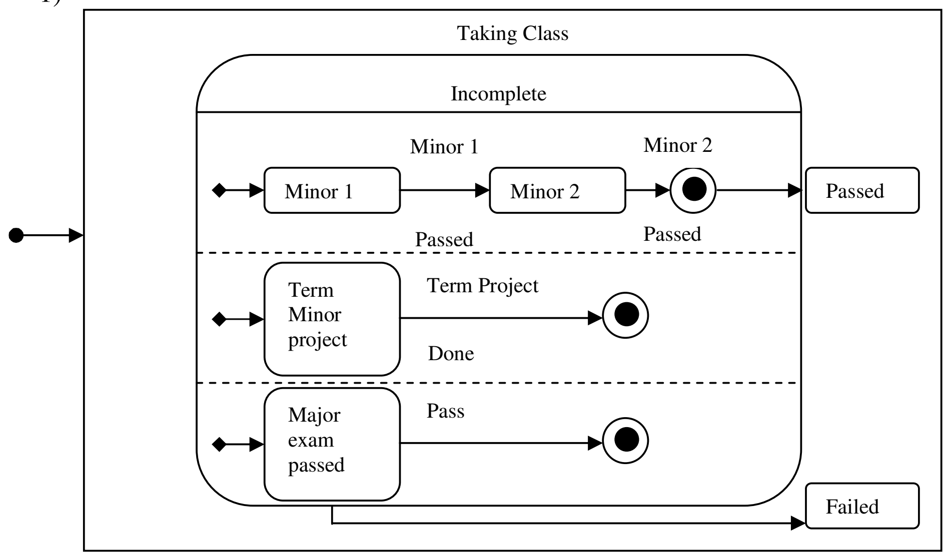


Figure 7: Concurrent Substate diagram

In *Figure 7* of concurrent substrates have been taken. After passing Minor 1 test you can give Minor 2 test. Term minor project of that semester minor should be done before Major exam of that semester.

2) The dynamic model specifies allowable sequences of changes to objects from the object model. It includes event trace diagrams describing scenarios. An event is an external stimulus from one object to another, which occurs at a particular point in time. An event is a one-way transmission of information from one object to another. A scenario is a sequence of events that occurs during one particular execution of a system. Each basic execution of the system should be represented as a scenario.

3) **Dynamic model for car:**

Accelerator and Brake	Applies Accelerator or Brake
Accelerator	Applies Accelerator
Brake	Applies Brake
off	Put off the car
on	Put on the car
off	
on	
press acc	
rel acc	
press brake	
rel brake	

4) In this diagram, you can observe if that initial state is state X if event Z occurs, then state X is terminated, and state Y is entered.

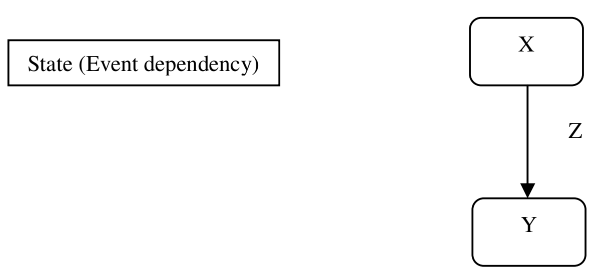


Figure 8: State event dependency