| Regularity | Regularity | Viva-voce | Timely Submission | Total | Dated Sign of Subject Teacher |
|---|---|---|---|---|---|
| 2 | 4 | 2 | 2 | 10 | |
| | | | | | |

**Date of performances:..... Expected Date of Completion:.....**

**Actual Date of Completion:.....**

---

## Assignment No: 7

---

**Title of the Assignment:** Prepare a state model from the given problem description, draw a state diagram using UML2 notations and Implement the state model with a suitable object oriented language.

---

**Objective of the Assignment:** Identify States and events for your system.

1. Study state transitions and identify Guard conditions.
2. Draw State chart diagram with advanced UML 2 notations.
3. Implement the state model with a suitable object-oriented language.

---

**Prerequisite:** Basic object oriented concepts.

---

**Theory:** According to view of a system there are three models namely,

1. Class Model
2. State Model
3. Interaction Model

**1. State Model:**

State Model describes those aspects of objects concerned with Time and sequencing of operations, Events that marks changes, States that define the context for events and The organization of events and states. This model captures control of a system, State diagram express the state model.

**2. State Diagram:**

State diagrams are used to describe the behavior of a system. These diagrams describe all of the possible states of an object as events occur. Each diagram usually represents objects of a single class and tracks the different states of its objects through the system. Using a state machine we can model the behavior of a society of objects that work together or behavior of an individual object.

**2.1 State machine:** A state machine is a behavior that specifies the sequences of states an object goes through during its life time in response to events, together with its responses to those events.

**a. State:** A state is a condition of an object in which it performs some activity or waits for an event. Denoted by a rectangle with rounded corners and compartments (such as a class with rounded corners to denote an Object).

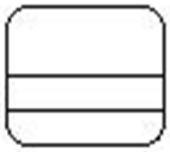

**Fig 1 Graphical Notation for State**

A state has following things associated with it

| | |
|---|---|
| **Name** | A textual string which distinguishes the state from other states; a state may also be anonymous, meaning that it has no name. |
| **Entry/exit actions** | Actions executed on entering and exiting the state. |
| **Internal transitions** | Transitions that are handled without causing a change in state. |
| **Substates** | The nested structure of a state, involving disjoint (sequentially active) or concurrent (concurrently active) sub states. |
| **Deferred events** | A list of events that are not handled in that state but are postponed and queued for handling by the object in another state. |

**b. Entry and Exit Actions**

Entry and exit actions allow the same action to be dispatched every time the state is entered or left, respectively. Entry and exit actions enable this to be done cleanly, without having to explicitly put the actions on every incoming or outgoing transition explicitly. Entry and exit actions may not have arguments or guard conditions. The entry actions at the top-level of a state machine for a model element may have parameters representing the arguments that the machine receives when the element is created.

**c. Internal Transitions**

Internal transitions allow events to be handled within the state without leaving the state, thereby avoiding triggering entry or exit actions. Internal transitions may have events with parameters and guard conditions, and essentially represent interrupt-handlers.

**d. Deferred Events**

Deferred events are those whose handling is postponed until a state in which the event is not deferred becomes active. When this state becomes active, the event occurrence is triggered and may cause transitions as if it had just occurred. The implementation of deferred events requires the presence of an internal queue of events. If an event occurs but is listed as deferred, it is queued. Events are taken off this queue as soon as the object enters a state that does not defer these events.

**e. Composite state-** A state that consists of sub states (concurrent sub states or sequential sub states) is called as composite states. A composite state may contain either concurrent (orthogonal) or sequential (disjoint) substrates. A composite state is either a simple composite state (with just one region) or an orthogonal state (with more than one region)

**f. Region:** A region is an orthogonal part of either a composite state or a state machine. It contains states and transitions.

**g. Sequential sub state:** A sequential substate is a substate in which an object can reside to the exclusion of all other sub states at that same level within the given composite state. Given two or more sequential sub states at the same level, an object can be in only one of them. It is also called as disjoint state.

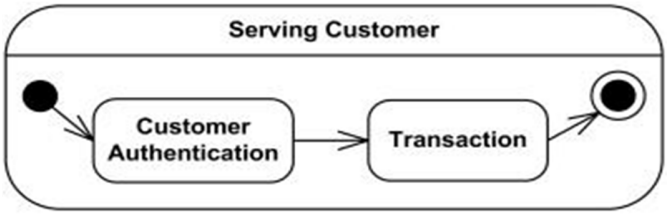


**Fig2: sequential Sub states**

**h. Concurrent sub state:** A concurrent substate is a substate in which an object can reside simultaneously with other sub states at that same level within the given composite state. Given two or more composite sub states at the same level, an object may be in one or more of them. In this situation, two or more sets of sub states can represent parallel flows of control. When the object enters a composite state with concurrent sub states, the object also enters the initial states of each set of sub states. Just like on an activity diagram, you can resynchronize these parallel flows. On a state diagram, however, you show this resynchronization by using a final state for each parallel set of concurrent sub states.

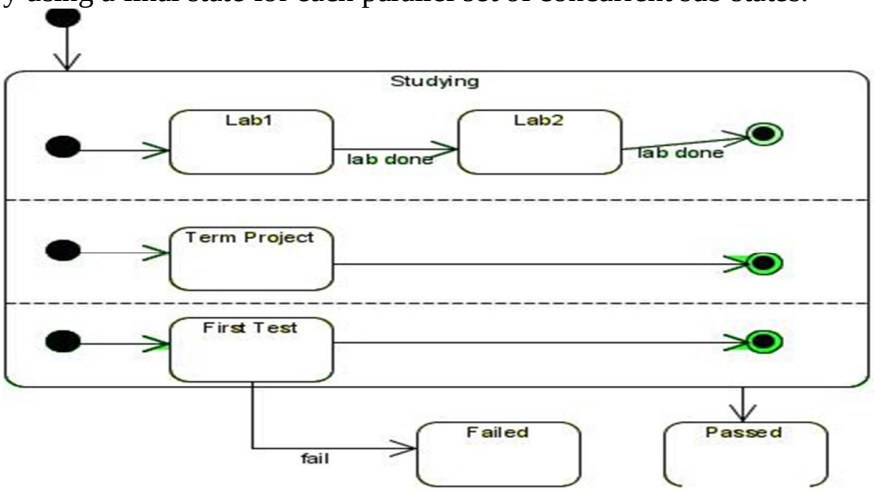


**Fig3 Concurrent sub states separated by region**

**i. 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.

**j. Transition**

A transition is a directed relationship between a source vertex and a target vertex. It may be part of a compound transition, which takes the state machine from one state configuration to another, representing the complete response of the state machine to an occurrence of an event of a particular type

A transition has five parts:

1. Source state: The state affected by the transition; if an object is in the source state an outgoing transition may fire when the object receives the trigger event of the transition and if the guard condition, if any, is satisfied.
2. Event trigger: The event whose reception by the object in the source state makes the transition eligible to fire.

3. Guard condition: A Boolean expression that is evaluated when the transition is triggered by the reception of the event trigger.
4. Action: An executable atomic computation that may directly act on the object that owns the state machine.
5. Target state: The state that is active after the completion of the transition.

### k. Event Triggers

In the context of the state machine, an event is an occurrence of a stimulus that can trigger a state transition. Events may include signal events, call events, the passing of time, or a change in state. A signal or call may have parameters whose values are available to the transition, including expressions for the guard conditions and action. It is also possible to have a trigger less transition, represented by a transition with no event trigger. These transitions, also called completion transitions, is triggered implicitly when its source state has completed its activity.

### l. Guard Conditions

A guard condition is evaluated after the trigger event for the transition occurs. It is possible to have multiple transitions from the same source state and with the same event trigger, as long as the guard conditions don't overlap. A guard condition is evaluated just once for the transition at the time the event occurs. The boolean expression may reference the state of the object.

### m. Actions

An action is an executable atomic computation, meaning that it cannot be interrupted by an event and therefore runs to completion. This is in contrast to an activity, which may be interrupted by other events. Actions may include operation calls (to the owner of the state machine as well as other visible objects), the creation or destruction of another object, or the sending of a signal to another object. In the case of sending a signal, the signal name is prefixed with the keyword 'send'.

## 3. UML State Machine Diagram Example for Bank ATM

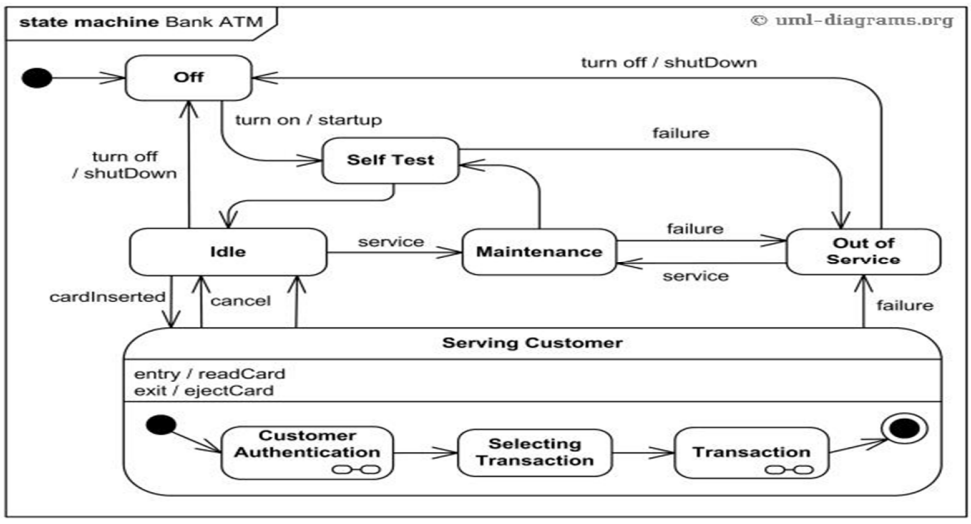


Fig 4 state Machine diagram for ATM

This is an example of UML behavioral state machine diagram showing Bank Automated Teller Machine (ATM) top level state machine.

ATM is initially turned off. After the power is turned on, ATM performs startup action and enters Self Test state. If the test fails, ATM goes into Out of Service state, otherwise there is trigger less transition to the Idle state. In this state ATM waits for customer interaction.
The ATM state changes from Idle to Serving Customer when the customer inserts banking or credit card in the ATM's card reader. On entering the Serving Customer state, the entry action read Card is performed. Note, that transition from Serving Customer state back to the Idle state could be triggered by cancel event as the customer could cancel transaction at any time.
Serving Customer state is a composite state with sequential sub states Customer Authentication, Selecting Transaction and Transaction. Customer Authentication and Transaction are composite states by themselves which is shown with hidden decomposition indicator icon. Serving Customer state has trigger less transition back to the Idle state after transaction is finished. The state also has exit action eject Card which releases customer's card on leaving the state, no matter what caused the transition out of the state.

**4. Oral Questions**

1. State Diagram is which kind of diagram?
2. Define State, State machine, Transition and Event.
3. Give difference between Activity diagram and State Diagram.
4. What do you mean by Tireless Transition?

**Assignment Question**

Prepare and Implement State Diagram for Washing Machine, ATM Machine and Coffee wandering machine

**Conclusion:**

So we can conclude that a state diagram, sometimes known as a state machine *diagram*, is a type of behavioral diagram in the Unified Modeling Language (UML) that shows transitions between various objects.