Name: Niraj Amrutkar

Roll No. 332002

PRN: 22010910

Batch: B1

Assignment 5

AIM: Implement the state model with a suitable object-oriented language.

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

Theory:

State diagram:

A state diagram is used to represent the condition of the system or part of the system at finite instances of time. It's a behavioral diagram and it represents thebehavior using finite state transitions. State diagrams are also referred to as State machines and State-chart Diagrams. These terms are often used interchangeably. So simply, a state diagram is used to model the dynamic behavior of a class in response to time and changing external stimuli. We can say that each and every class has a state but we don't model every class using State diagrams. We prefer to model the states with three or more states.

"The state machine view describes the dynamic behavior of objects over time by modeling the lifecycles of objects of each class. Each object is treated as an isolated entity that communicates with the rest of the world by detecting events and responding to them. Events represent the kinds of changes that objects can detect... Anything that can affect an object can be characterized as an event."

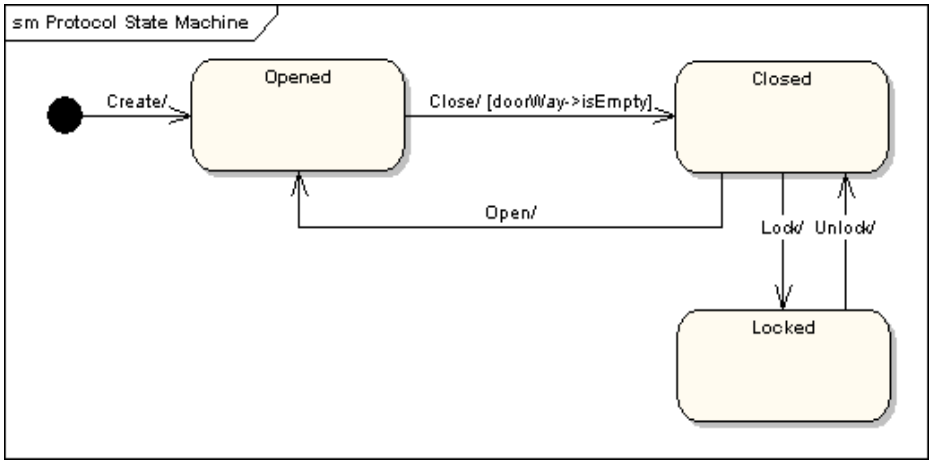
An object must be in some specific state at any given time during its lifecycle. An object transitions from one state to another as the result of some event that affects it. You may create a state diagram for any class, collaboration, operation, or use case in a UML model .
There can be only one start state in a state diagram, but there may be many intermediate and final states.

Uses of state chart diagram –

We use it to state the events responsible for change in state (we do not showwhat processes cause those events).

We use it to model the dynamic behavior of the system .

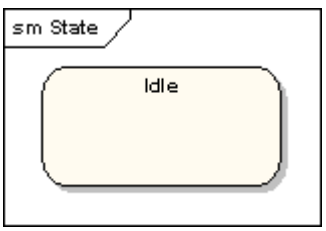
- A state machine diagram models the behavior of a single object, specifying the sequence of events that an object goes through during its lifetime in response to events. As an example, the following state machine diagram shows the states that a door goes through during its lifetime.



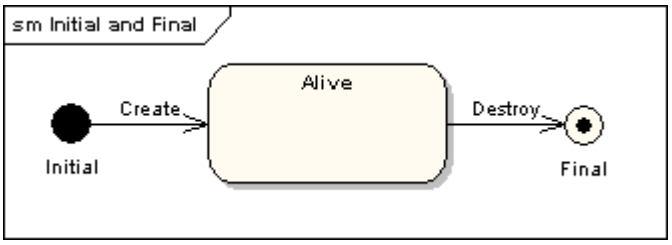
- The door can be in one of three states: "Opened", "Closed" or "Locked". It can respond to the events Open, Close, Lock and Unlock. Notice that not all events are valid in all states; for example, if a door is opened, you cannot lock it until you close it. Also notice that a state transition can have a guard condition attached: if the door is Opened, it can only respond to the Close event if the condition

- doorWay->isEmpty is fulfilled. The syntax and conventions used in state machine diagrams will be discussed in full in the following sections.

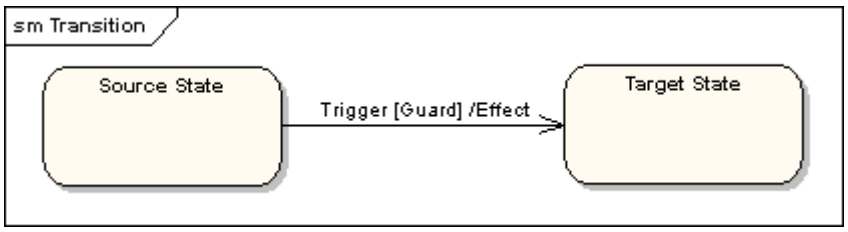
- ***States*** - A state is denoted by a round-cornered rectangle with the name of the state written inside it.

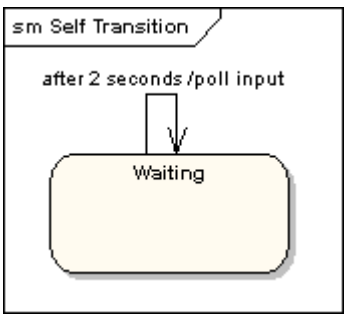


- ***Initial and Final States*** - The initial state is denoted by a filled black circle and may be labeled with a name. The final state is denoted by a circle with a dot inside and may also be labeled with a name.



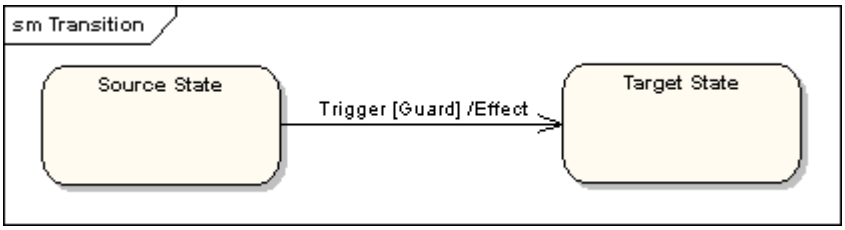
- ***Transitions*** - Transitions from one state to the next are denoted by lines with arrowheads. A transition may have a trigger, a guard and an effect, as below.
- ***Self-Transitions*** - A state can have a transition that returns to itself, as in the following diagram. This is most useful when an effect is associated with the transition.



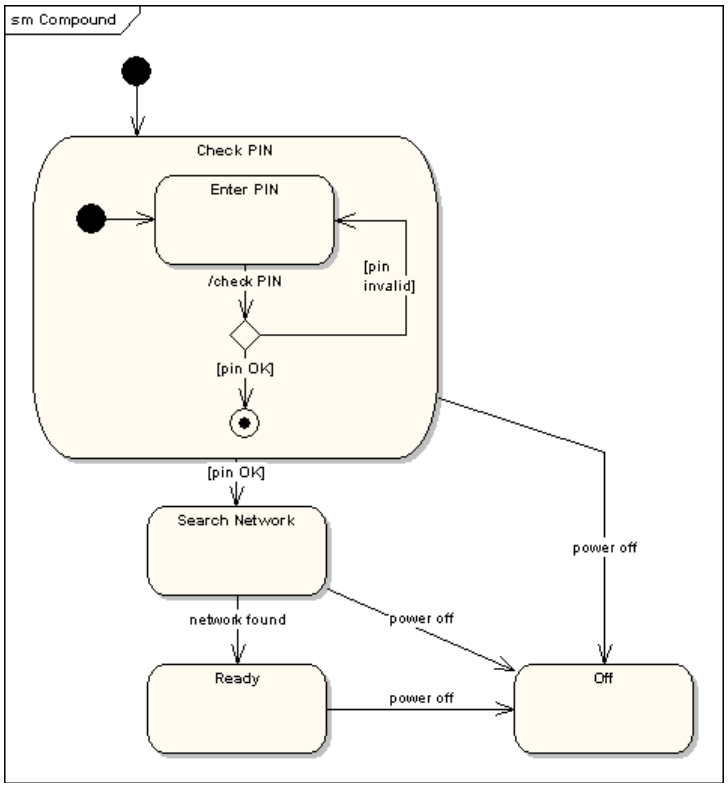


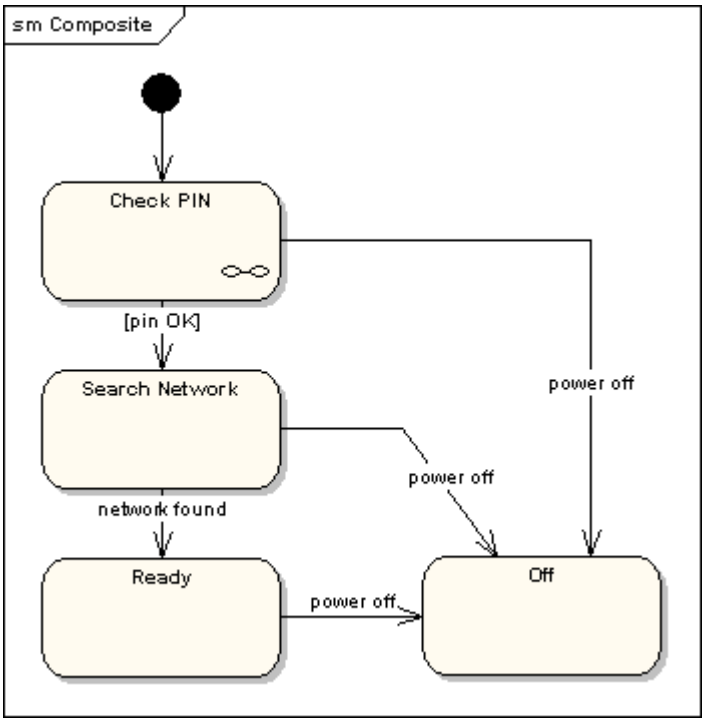
- "***Trigger***" is the cause of the transition, which could be a signal, an event, a change in some condition, or the passage of time. "***Guard***" is a condition which must be true in order for the trigger to cause the transition. "Effect" is an action which will be invoked directly on the object that owns the state machine as a result of the transition.

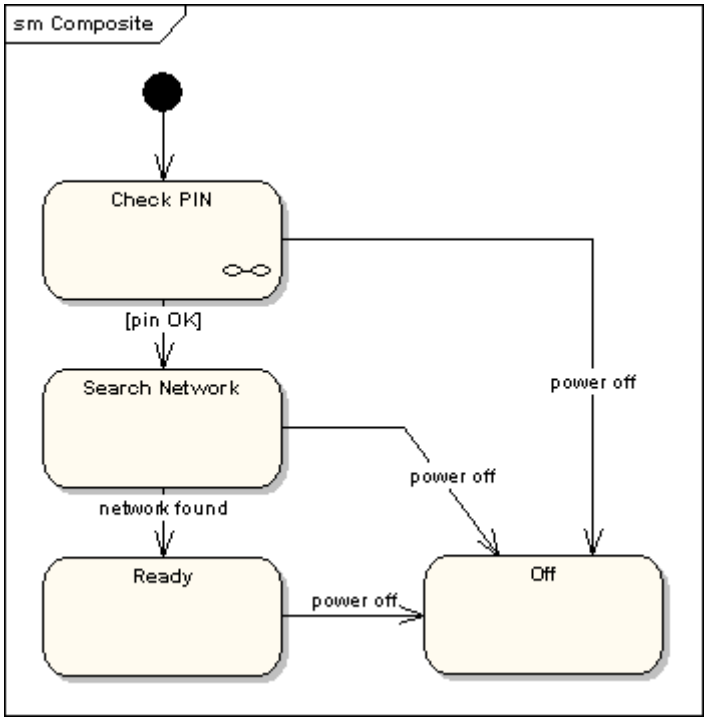
- ***State Actions*** - In the transition example above, an effect was associated with the transition. If the target state had many transitions arriving at it, and each transition had the same effect associated with it, it would be better to associate the effect with the target state rather than the transitions. This can be done by defining an entry action for the state. The diagram below shows a state with an entry action and an exit action.



- It is also possible to define actions that occur on events, or actions that always occur. It is possible to define any number of actions of each type.
- ***Compound States*** - A state machine diagram may include sub-machine diagrams, as in the example below.



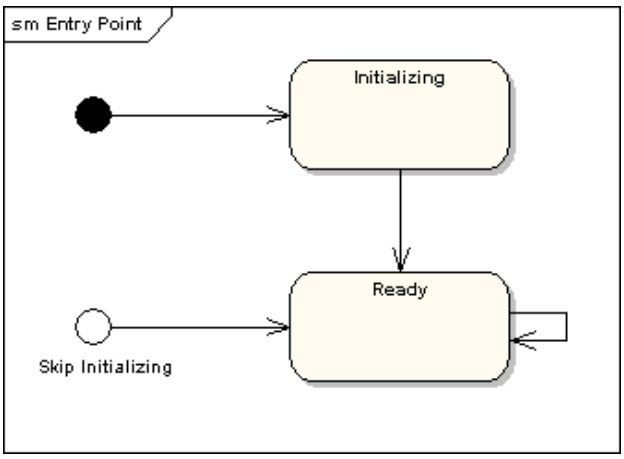


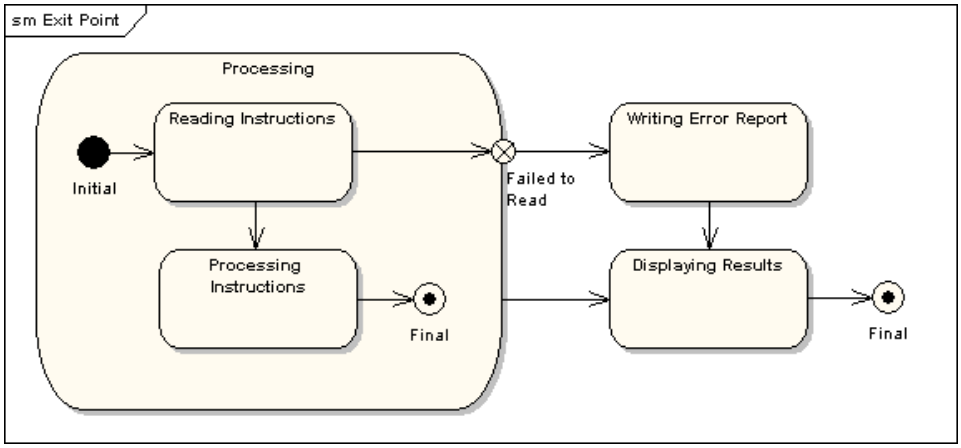


- **The ∞ symbol indicates that details of the Check PIN sub-machine are shown in a separate diagram.**

  - ***Entry Point*** - Sometimes you won't want to enter a sub-machine at the normal initial state. For example, in the following sub-machine it would be normal to begin in the "Initializing" state, but if for some

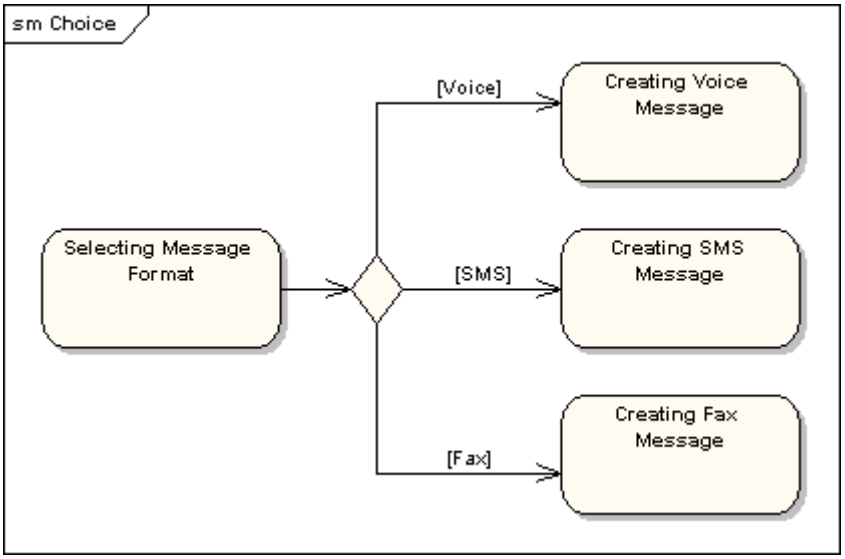
reason it wasn't necessary to perform the initialization, it would be possible to begin in the "Ready" state by transitioning to the named entry point.



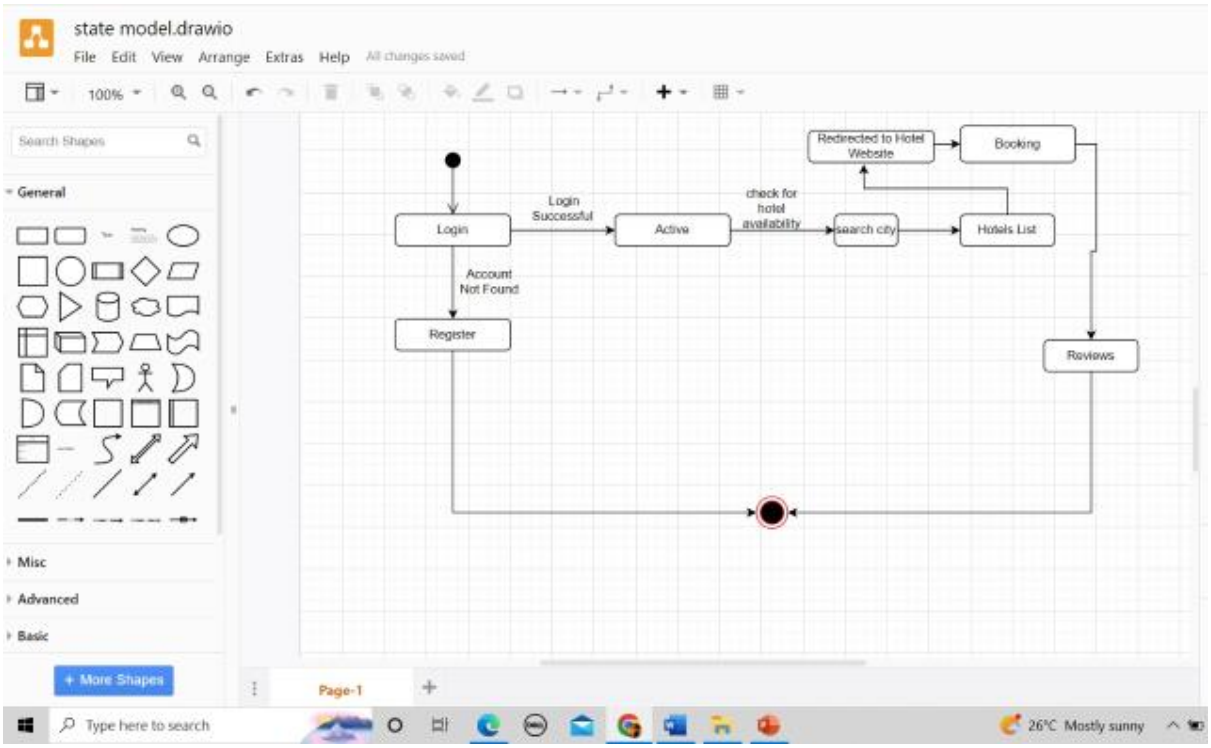


- ***Exit Point*** - In a similar manner to entry points, it is possible to have named alternative exit points. The following diagram gives an example where the state executed after the main processing state depends on which route is used to transition out of the state.

 Choice Pseudo-State - A choice pseudo-state is shown as a diamond with one transition arriving and two or more transitions leaving. The following diagram shows that whichever state is arrived at, after the choice pseudo-state, is dependent on the message format selected during execution of the previous state.



Diagram



Conclusion: In this assignment, we learned about state transitions and implemented them by creating state transition diagrams for the actors involved.