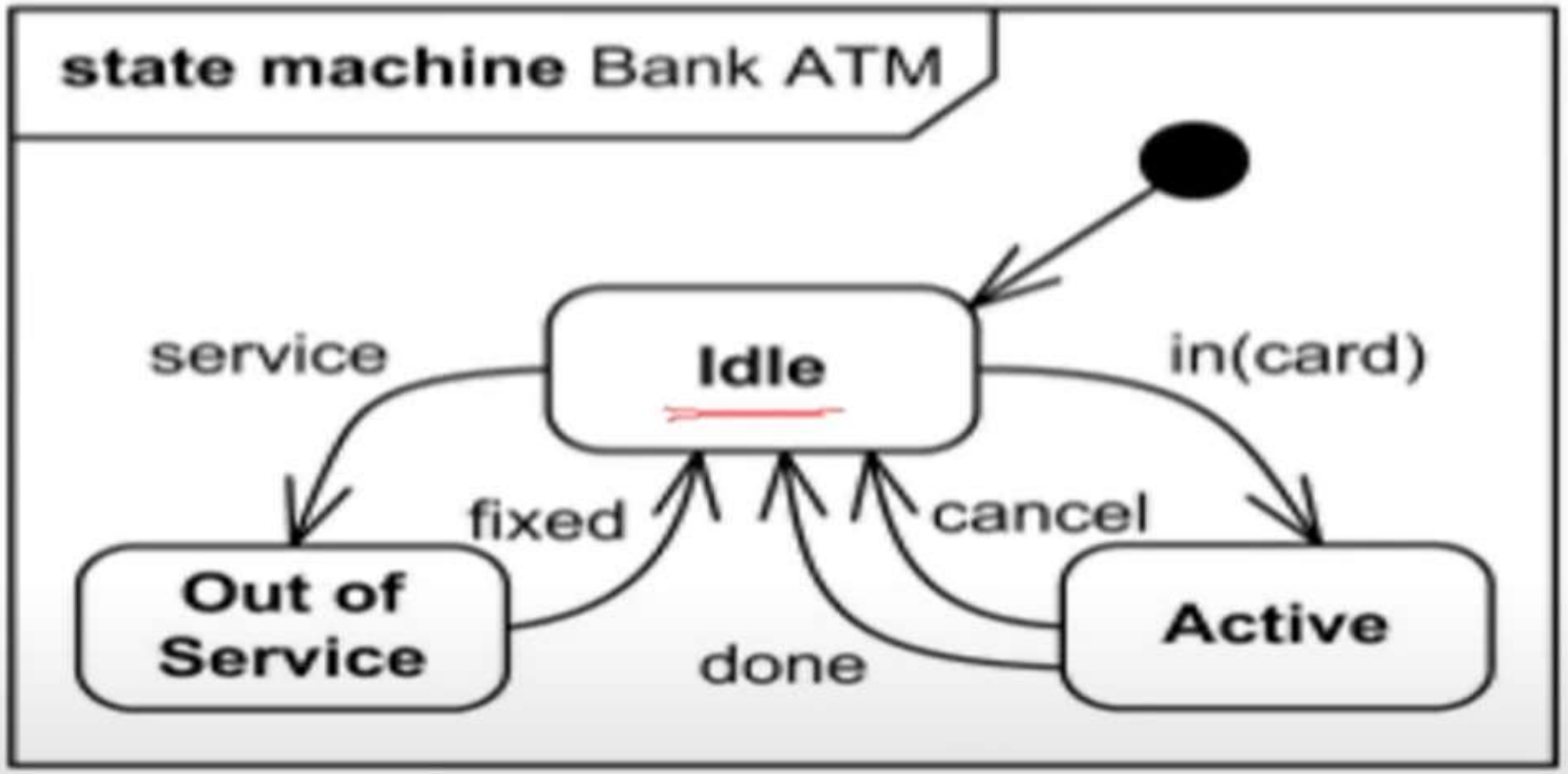
Object-Oriented Analysis and Design using JAVA

B.Tech (CSE/IT) 5th SEM
2020-2021

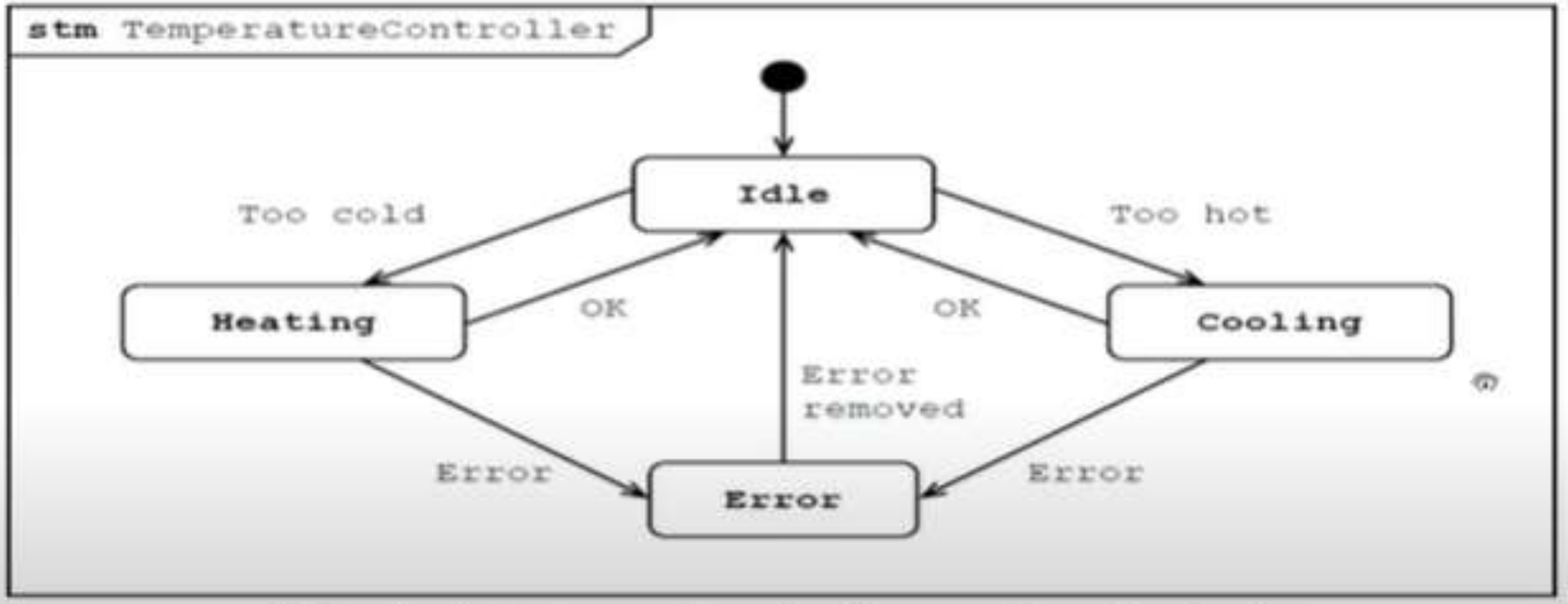
Lecture-25 State Machine diagram-2

Examples



High level behavioral state machine for bank ATM

Examples



Behavioral state machine for Temperature Controller

Introduction

- State machine diagram typically are used to describe state-dependent behavior for an object. An object responds differently to the same event depending on what state it is in.

For example:
Consider you have $100,000 in a bank account. The behavior of the withdraw function would be: balance := balance – withdraw Amount; provided that **the balance after the withdrawal is not less than $0;** this is true regardless of how many times you have withdrawn money from the bank. In such situations, the withdrawals do not affect the abstraction of the attribute values, and hence the gross behavior of the object remains unchanged.

However, if the **account balance would become negative after a withdrawal**, the behavior of the withdraw function would be quite different. This is because the state of the bank account is changed from positive to negative; in technical jargon, a transition from the positive state to the negative state is fired.

The abstraction of the attribute value is a property of the system, rather than a globally applicable rule. For example, if the bank changes the business rule to allow the bank balance to be overdrawn by 2000 dollars, the state of the bank account will be redefined with condition that the balance after withdrawal must not be less than $2000 in deficit.
Note That:

- A state machine diagram describes all events (and states and transitions for a single object)
- A sequence diagram describes the events for a single interaction across all objects involved

Reference: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/

Vertex

Vertex is named element which is an abstraction of a node in a state machine graph. In general, it can be the source or destination of any number of transitions.

Subclasses of vertex are:

- state
- pseudostate

State is a vertex which models a situation during which some (usually implicit) invariant condition holds.

Behavioral State

State in behavioral state machines models a situation during which some (usually implicit) invariant condition holds.

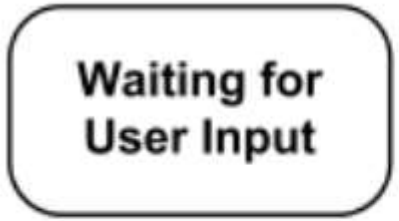
Inherited states are drawn with dashed lines or gray-toned lines.

The UML defines the following kinds of states:

- simple state,
- composite state,
- submachine state.

Simple State

A simple state is a state that does not have substates - it has no regions and it has no submachine states.



Simple state Waiting for Customer Input.

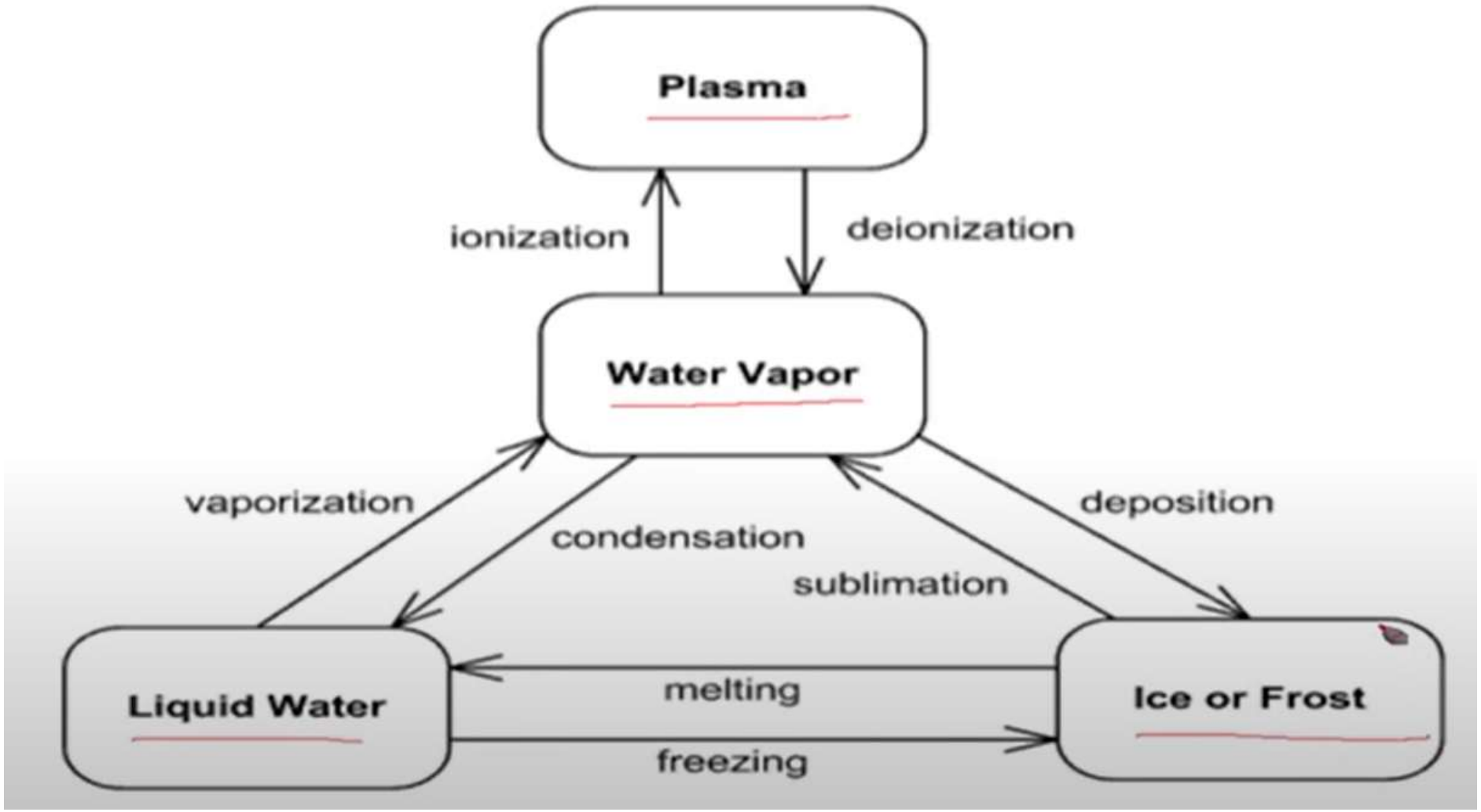
Simple state may have compartments.

The compartments of the state are:

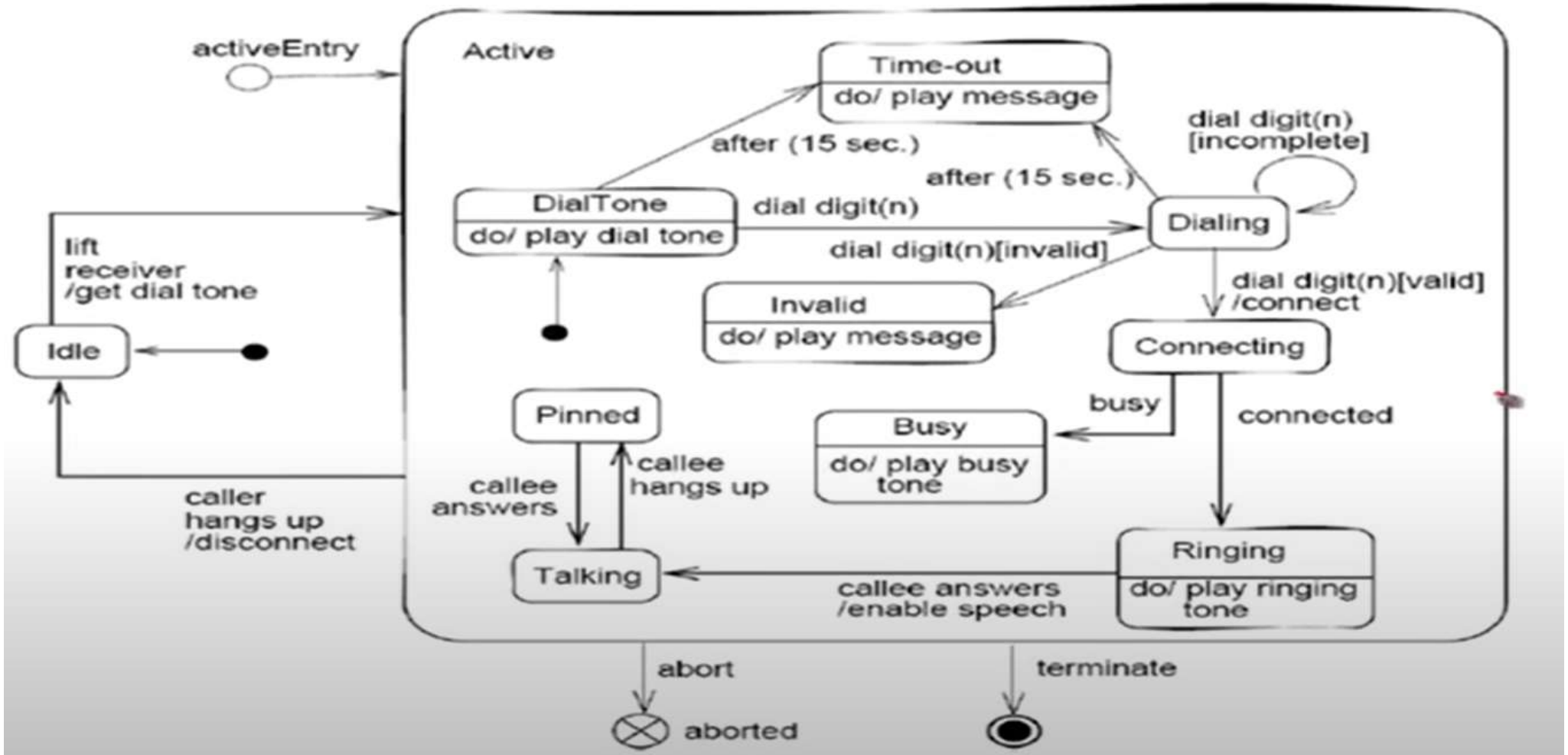
- name compartment
- internal activities compartment
- internal transitions compartment

Simple state ***Waiting for Customer Input*** *with name and internal activities compartments.*

Water Phase Management



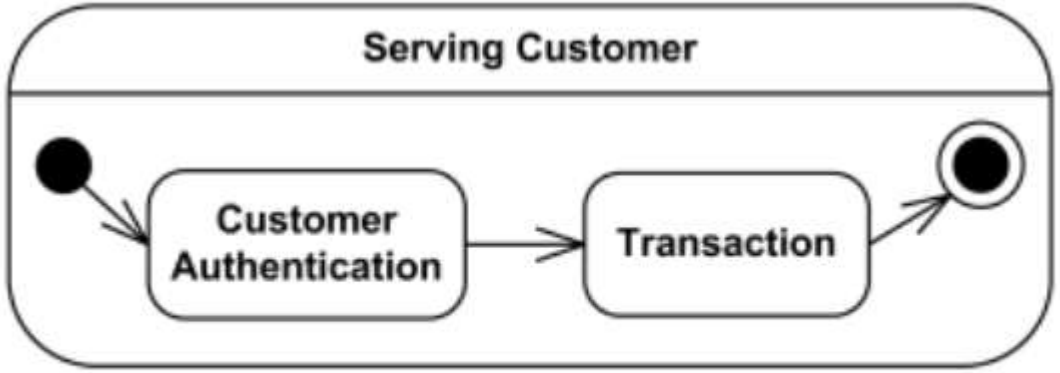
Behavioral State Machine



Behavioral state machine for Dialing a Phone

Composite State

- Generally, composite state is defined as state that has substates (nested states). Substates could be sequential (disjoint) or concurrent (orthogonal). UML 2.4 defines composite state as the state which contains one or more regions.
- A **region** contains states and transitions
- Simple composite state contains just one region.



Simple composite state Serving Customer has two substates.

Composite State

Orthogonal composite state has more than one regions.

Any state enclosed within a region of a composite state is called a **substate** of that composite state.

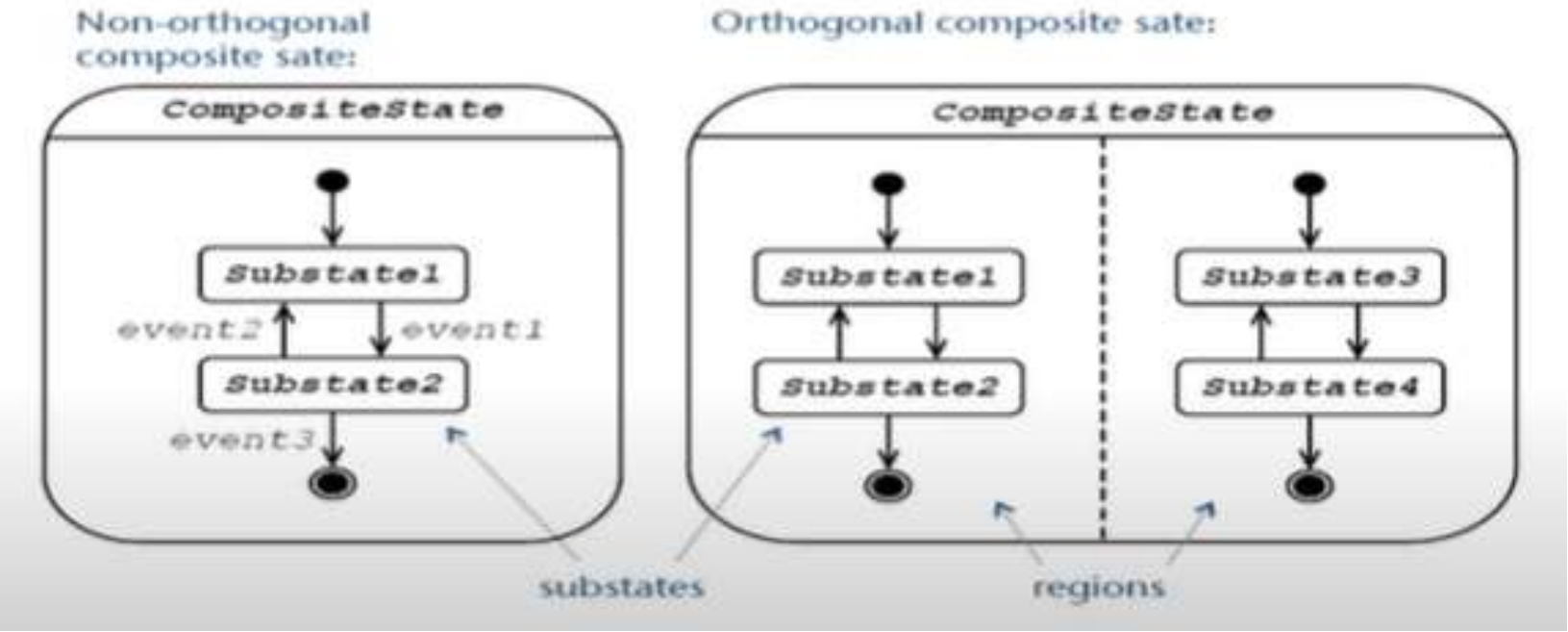
Composite state may have compartments. The compartments of the state are:

- name compartment
- internal activities compartment
- internal transitions compartment
- decomposition compartment

Decomposition compartment shows composition structure of the state as a a nested diagram with regions, states, and transitions.

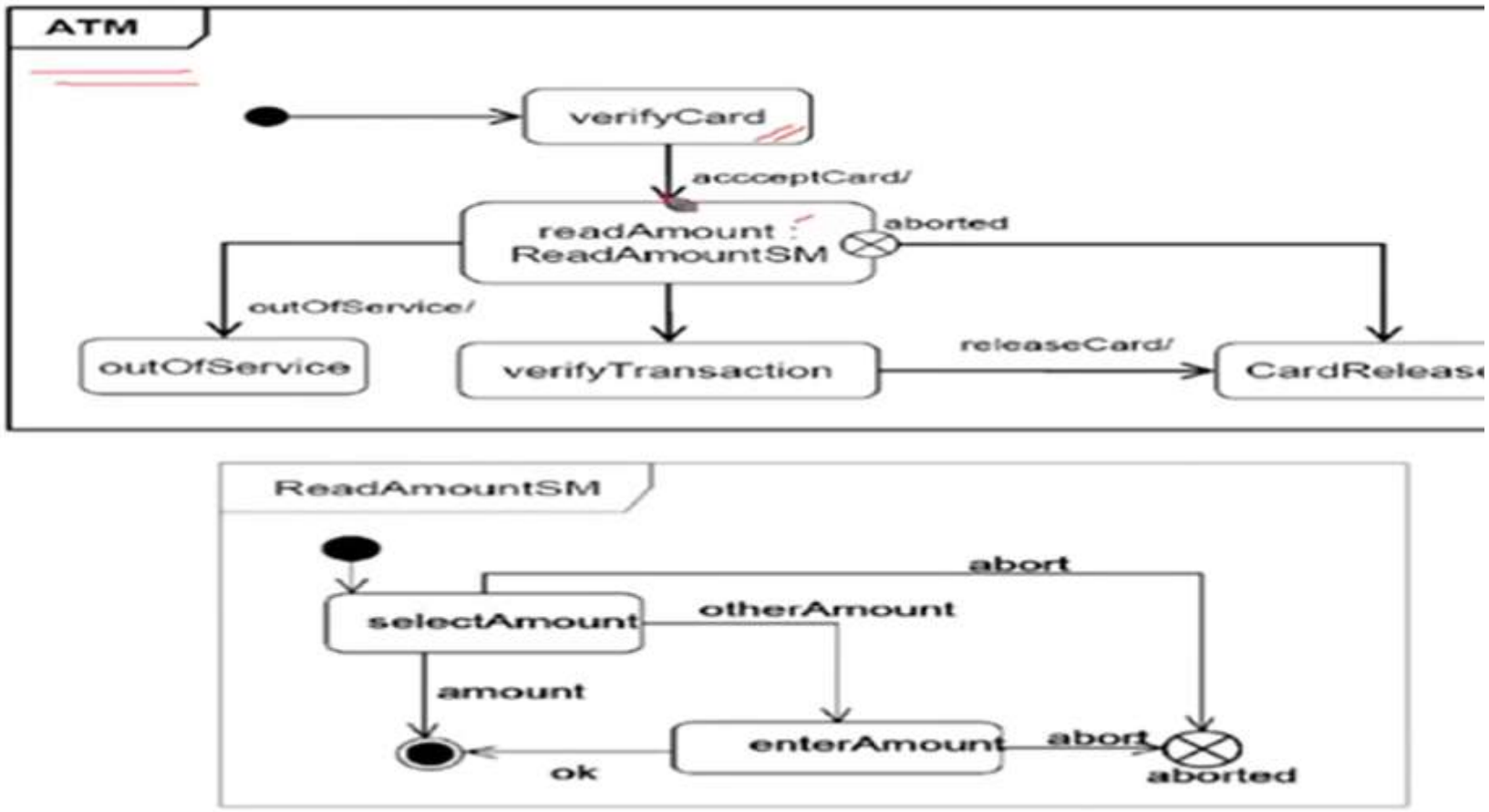
Composite state Serving Customer with decomposition hidden.

Composite State



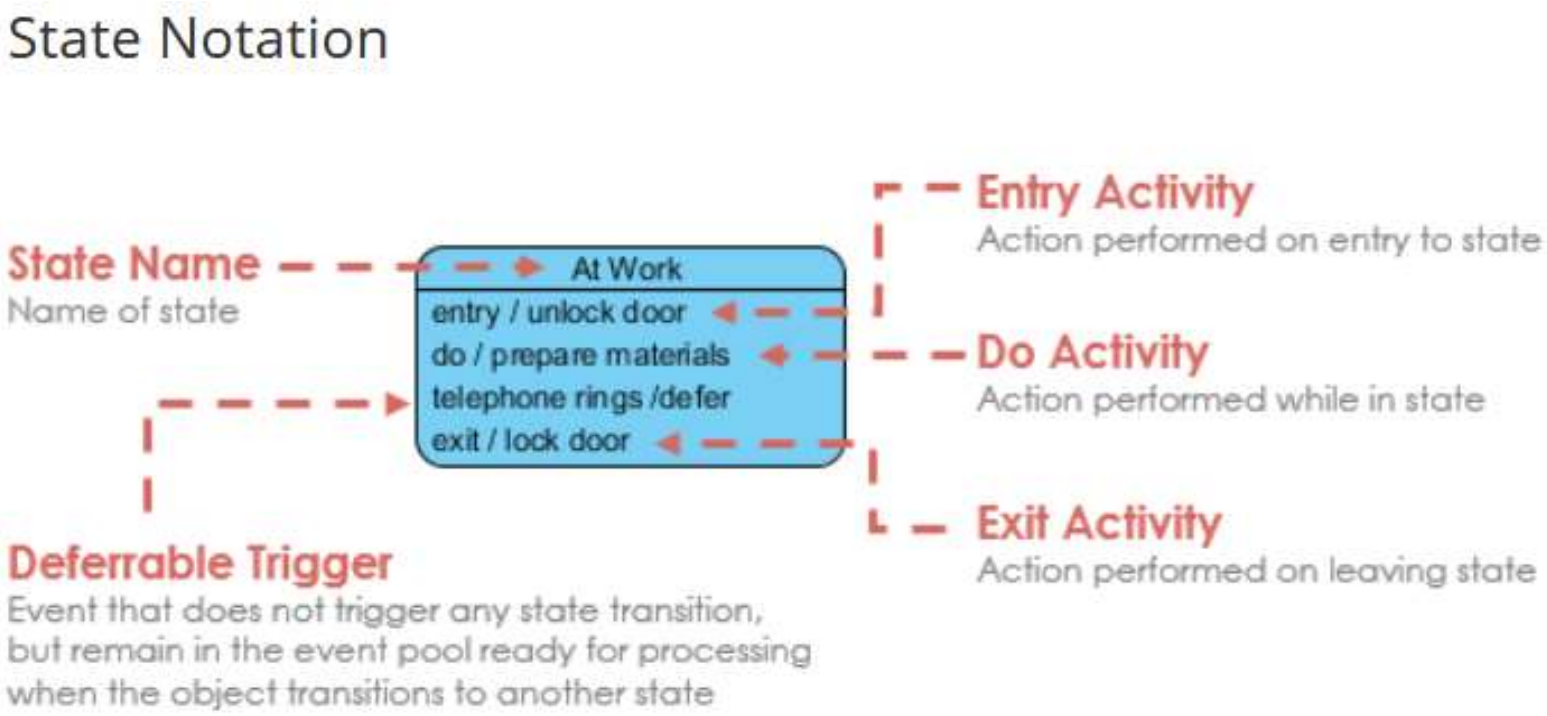
Submachine state

A submachine state specifies the insertion of the specification of a submachine state machine.



State

Rumbaugh defines that:
"A state is an abstraction of the attribute values and links of an object. Sets of values are grouped together into a state according to properties that affect the gross behavior of the object."



Reference: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/

There are several characteristics of states in general, regardless of their types:

- A state occupies an interval of time.
- A state is often associated with an abstraction of attribute values of an entity satisfying some condition(s).
- An entity changes its state not only as a direct consequence of the current input, but it is also dependent on some past history of its inputs.

State

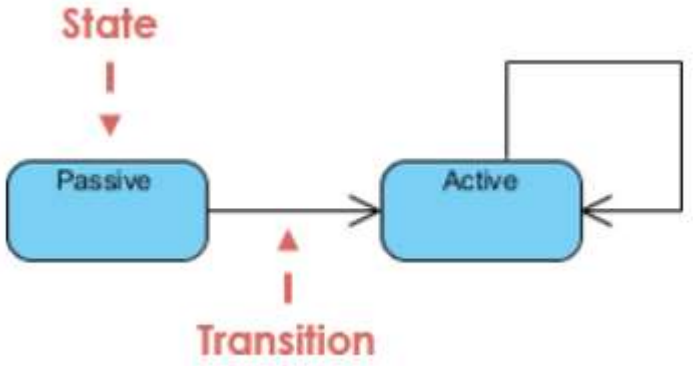
A state is a constraint or a situation in the life cycle of an object, in which a constraint holds, the object executes an activity or waits for an event. A state machine diagram is a graph consisting of:

- States (simple states or composite states)
- State transitions connecting the states

Reference: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/

Characteristics of State

- State represent the conditions of objects at certain points in time.
- Objects (or Systems) can be viewed as moving from state to state
- A point in the lifecycle of a model element that satisfies some condition, where some particular action is being performed or where some event is waited

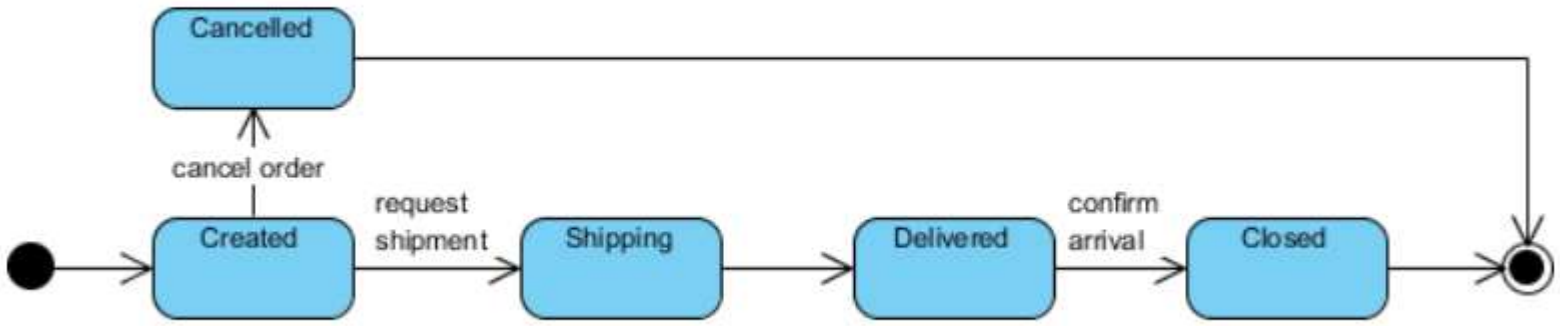


Reference: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/

Initial and Final States

- The **initial state** of a state machine diagram, known as an initial pseudo-state, is indicated with a solid circle. A transition from this state will show the first real state
- The **final state** of a state machine diagram is shown as concentric circles. An open loop state machine represents an object that may terminate before the system terminates, while a closed loop state machine diagram does not have a final state; if it is the case, then the object lives until the entire system terminates.

Example:



Reference: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/

Event

An event signature is described as Event-name (comma-separated-parameter-list). Events appear in the internal transition compartment of a state or on a transition between states. An event may be one of four types:

- Signal event - corresponding to the arrival of an asynchronous message or signal
- Call event - corresponding to the arrival of a procedural call to an operation
- Time event - a time event occurs after a specified time has elapsed
- Change event - a change event occurs whenever a specified condition is met

Characteristics of Events

- Represents incidents that cause objects to transition from one state to another.
- Internal or External Events trigger some activity that changes the state of the system and of some of its parts
- Events pass information, which is elaborated by Objects operations. Objects realize Events
- Design involves examining events in a state machine diagram and considering how those events will be supported by system objects

Reference: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/

Transition

Transition lines depict the movement from one state to another. Each transition line is labeled with the **event** that causes the transition.

- Viewing a system as a set of states and transitions between states is very useful for describing complex behaviors
- Understanding state transitions is part of system analysis and design
- A Transition is the movement from one state to another state
- Transitions between states occur as follows:
  - An element is in a source state
  - An event occurs
  - An action is performed
  - The element enters a target state
- Multiple transitions occur either when different events result in a state terminating or when there are guard conditions on the transitions
- A transition without an event and action is known as automatic transitions

Reference: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/

Action and activity

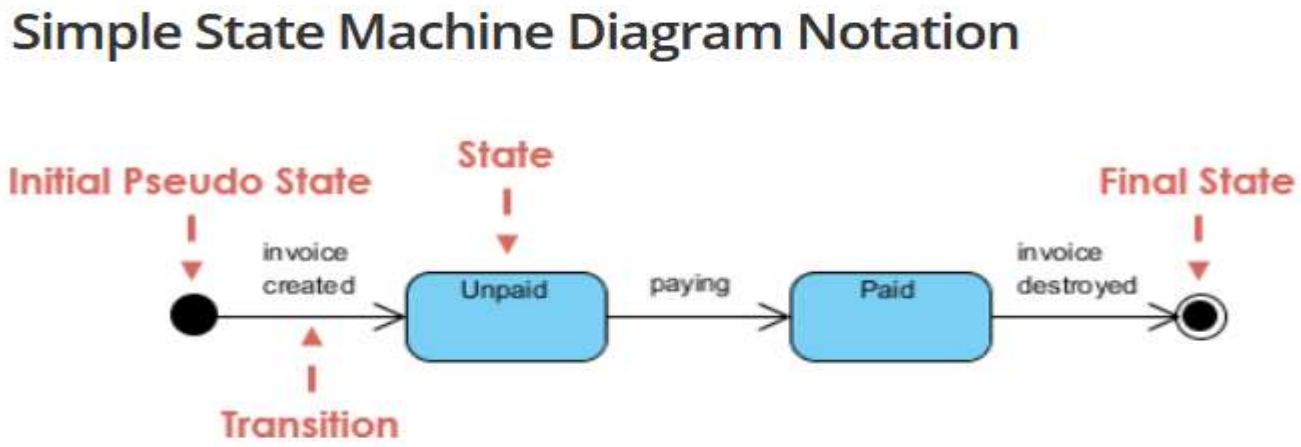
Actions: Action is an executable atomic computation, which includes operation calls, the creation or destruction of another object, or the sending of a signal to an object. An action is associated with transitions and during which an action is not interruptible - e.g., entry, exit

Activity: Activity is associated with states, which is a non-atomic or ongoing computation. Activity may run to completion or continue indefinitely. An Activity will be terminated by an event that causes a transition from the state in which the activity is defined

Characteristics of Action and Activities

- States can trigger actions
- States can have a second compartment that contains actions or activities performed while an entity is in a given state
- An action is an atomic execution and therefore completes without interruption
- Five triggers for actions: On Entry, Do, On Event, On Exit, and Include
- An activity captures complex behavior that may run for a long duration - An activity may be interrupted by events, in which case it does not complete occur when an object arrives in a state.

Reference: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/

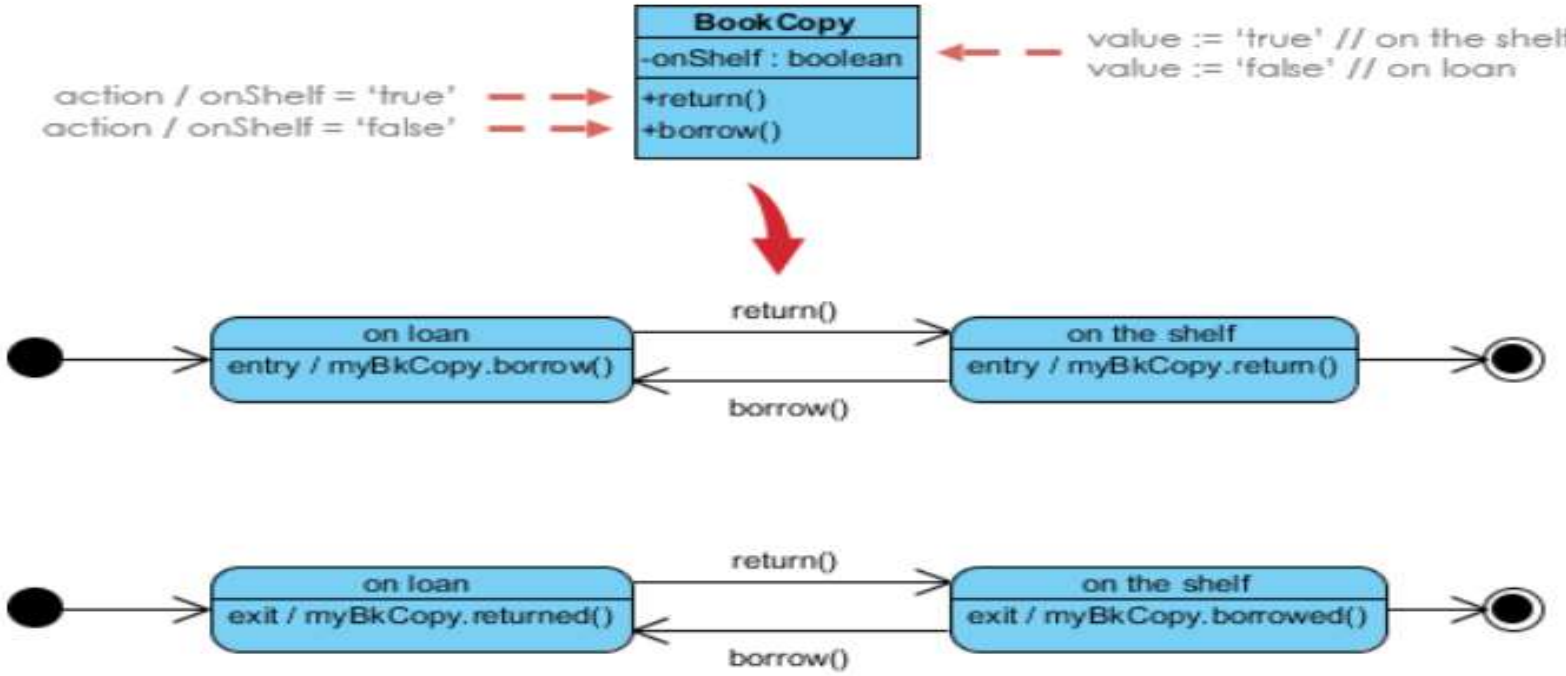


Entry and Exit Actions

Entry and Exit actions specified in the state. It must be true for every entry / exit occurrence. If not, then you must use actions on the individual transition arcs

- **Entry Action** executed on entry into state with the **notation: Entry / action**
- **Exit Action** executed on exit from state with the **notation: Exit / action**

Reference: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/

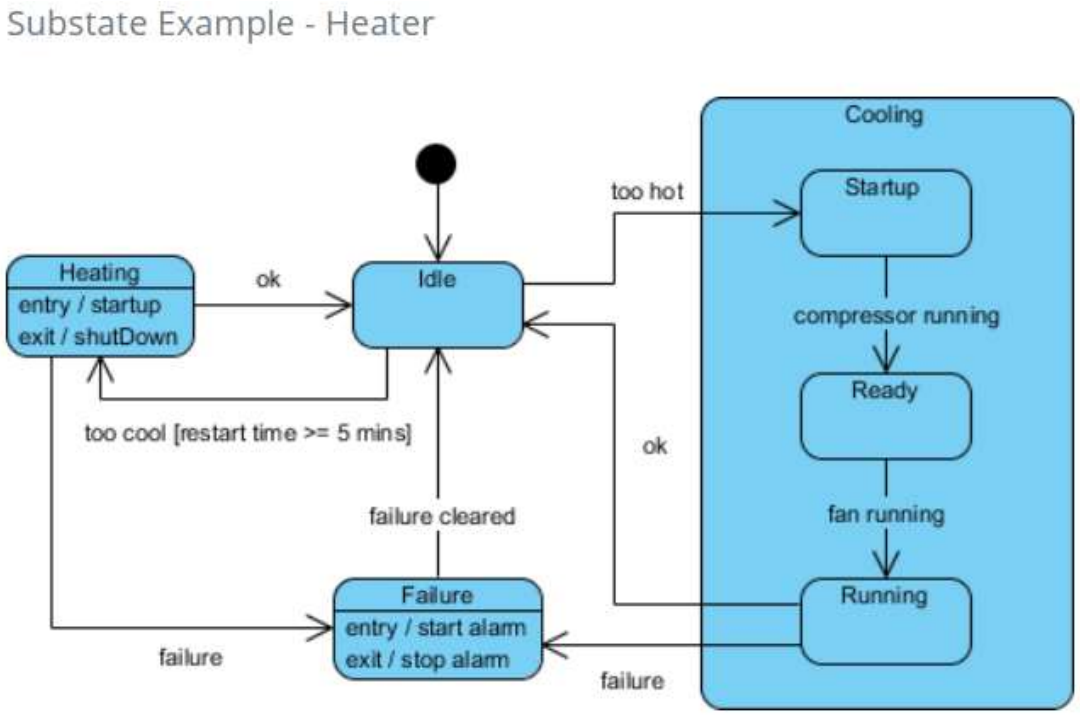


- This state machine diagram shows the state of an object myBkCopy from a BookCopy class
- Entry action : any action that is marked as linked to the entry action is executed whenever the given state is entered via a transition
- Exit action : any action that is marked as linked to the exit action is executed whenever the state is left via a transition

Reference: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/

Substate

A simple state is one which has no substructure. A state which has substates (nested states) is called a composite state. Substates may be nested to any level. A nested state machine may have at most one initial state and one final state. Substates are used to simplify complex flat state machines by showing that some states are only possible within a particular context (the enclosing state).



State Machine Diagrams are often used for deriving testing cases, here is a list of possible test ideas:

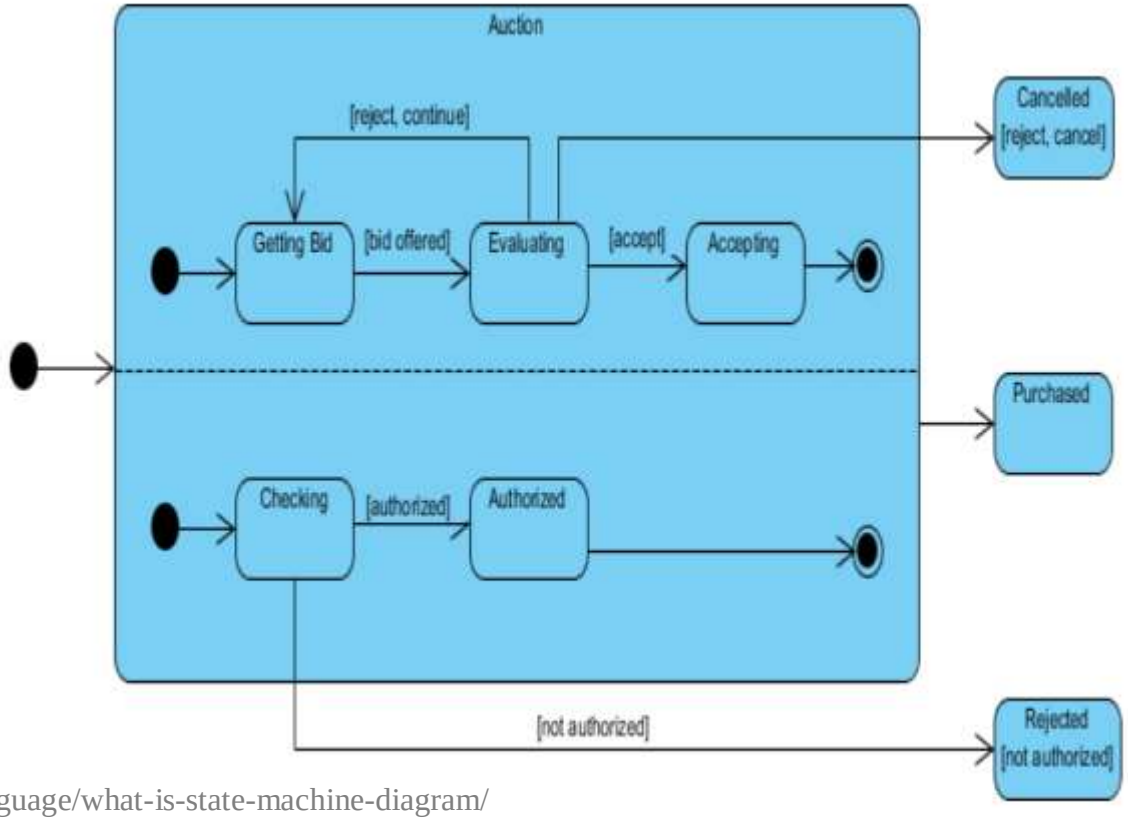
- Idle state receives Too Hot event
- Idle state receives Too Cool event
- Cooling/Startup state receives Compressor Running event
- Cooling/Ready state receives Fan Running event
- Cooling/Running state receives OK event
- Cooling/Running state receives Failure event
- Failure state receives Failure Cleared event
- Heating state receives OK event
- Heating state receives Failure event

Reference: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/

Concurrent state

As mentioned above, states in state machine diagrams can be nested. Related states can be grouped together into a single composite state. Nesting states inside others is necessary when an activity involves concurrent sub-activities. The following state machine diagram models an auction with two concurrent substates: processing the bid and authorizing the payment limit.

Concurrent State Machine Diagram Example - Auction Proces
In this example, the state machine first entering the Auction requires a fork at the start into two separate start threads. Each substate has an exit state to mark the end of the thread. Unless there is an abnormal exit (Canceled or Rejected), the exit from the composite state occurs when both substates have exited.



Reference: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram/

Key references

Unified Modeling Language (UML): Complete Guide & Examples
-By James Rumbaugh, Ivar Jacobson, Grady Booch

https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-state-machine-diagram

Thank You