

## State Transition Diagram

Topic # 13

Chapter 29 – UML State  
Machine Diagrams and  
Modeling – Craig Larman

### i. Static

- a. Use case diagram
- b. Class diagram

### ii. Dynamic

- a. Activity diagram
- b. Sequence diagram
- c. Object diagram
- d. State diagram**
- e. Collaboration diagram

### iii. Implementation

- a. Component diagram
- b. Deployment diagram

## State Chart Diagrams

- 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.
- A state chart diagram is normally used to model how the state of an object changes in its lifetime.
- **When and when not?**
  - State chart diagrams are good at describing how the behavior of an object changes across several use case executions.
  - However, if we are interested in modeling some behavior that involves several objects collaborating with each other, state chart diagram is not appropriate.

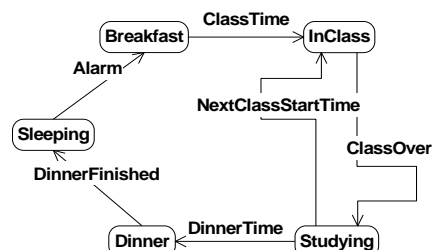
## State Chart Diagrams

- **Object state** – a condition of the object at one point in its lifetime.
- **State transition event** – an occurrence that triggers a change in an object's state through the updating of one or more of its attribute values.
- **State chart diagram** – a UML diagram that depicts
  - the combination of states that an object can assume during its lifetime,
  - the events that trigger transitions between states,
  - the rules governing the from and to states an object may transition.





## Why Use State chart Diagrams?

- State charts 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.
  - Usually applied to objects but can be applied to any element that has behavior
    - Actors, use cases, methods, subsystems, systems
- Statecharts are typically used in conjunction with interaction diagrams (usually sequence diagrams)
  - A statechart 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

## State Diagram- Investigating



## State Chart Diagram Symbols

A STATE	
AN INITIAL STATE	
A FINAL STATE	
AN EVENT	anEvent
A TRANSITION	

## States

- State is a condition or situation during the life of an object within which it performs some activity, or waits for some events .
- State may also label activities, e.g., do/check item
- Examples of object states are:
  - The invoice (object) is paid (state).
  - The car (object) is standing still (state).
  - The engine (object) is running (state).
  - Jim (object) is playing the role of a salesman (state).

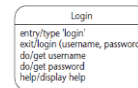
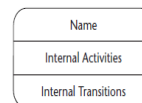


## State Compartments

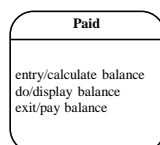
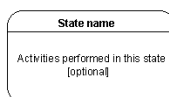
- A state may contain three kinds of compartments.
- The first compartment shows the **name of the state**, for example, idle, paid, and moving.
- The second compartment is the optional **activity compartment**, which lists behavior in response to events. You can define your own event, such as selecting a Help button, as well as the activity to respond to that event.
- Three standard events names are reserved in UML: entry, exit, and do.
  - The **entry event** can be used to specify actions on the entry of a state; for example, assigning an attribute or sending a message.
  - The **exit event** can be used to specify actions on exit from a state.
  - The **do event** can be used to specify an action performed while in the state; for example, sending a message, waiting, or calculating.

## State Compartments

- The third compartment is the optional internal transition compartment.
- This compartment contains a list of internal transitions.
- A transition can be listed more than once if it has different guard conditions.



## State Actions



## Initial/Final States

### Initial State

A filled circle followed by an arrow represents the object's initial state.



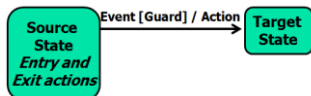
### Final State

An arrow pointing to a filled circle nested inside another circle represents the object's final state.



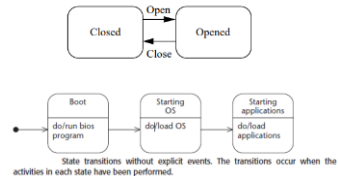
## Transition

- Show what the dependencies are between the state of an object and its reactions to messages or other events
- A solid arrow represents the path between different states of an object.
- Label the transition with the event that triggered it and the action that results from it.
- A state can have a transition that points back to itself.



## Examples

- If a Bank Account was Closed and it saw an Open event, it would end up in the Opened state - If the account was Opened and it saw a Close event it would end up in the Closed state.

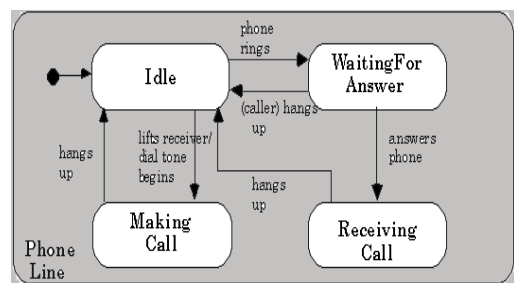


## Events

- An object transitions (changes) from one state to another state when something happens, which is called an event; for example, someone pays an invoice, starts driving the car,
- An event is an instant in time that may be significant to the behavior of the objects in a class - Events can have associated arguments
- Events are written simply as text strings
  - Open
  - Deposit(Amount)
  - Withdraw(Amount)
  - Close



## State Chart Diagram – Phone Line



## Types of Events

- UML has four types of events, also referenced as triggers because the event triggers behavior that depends on how you build the system.
- A **condition** becoming true. This is shown as a guard-condition on a state transition.
- **Receipt of an explicit signal** from another object. The signal is itself an object. This type of event is called a message.
- **Receipt of a call** on an operation by another object (or by the object itself). This type of event is also called a message.
- **Passage of a designated period of time.**

## Transition label examples

- Transition name only:  
DeleteOrder()
- No guard condition:  
Order item (ItemID,qty)/reset register
- Guard condition only:  
[Credit no good]
- Multiple actions:  
CreateOrder(Order Information)/Build order information; Associate with Customer

## Guard Condition

- Guard-condition is a Boolean expression placed on a state transition. If the guard-condition is combined with an event-signature, the event must occur, *and the guard-condition must be true for the transition to fire.*
- If *only a guardcondition* is attached to a state transition, the transition fires when the condition becomes true. Examples of state transitions with a guard-condition are as follows:
  - $[t = 15\text{sec}]$
  - $[\text{number of invoices} > n]$
  - $\text{withdrawal}(\text{amount}) [\text{balance} \geq \text{amount}]$

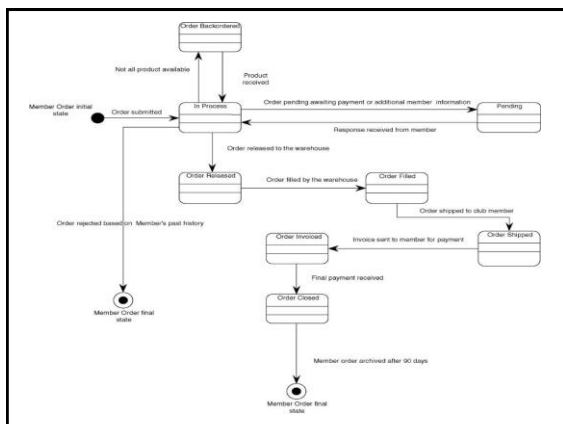
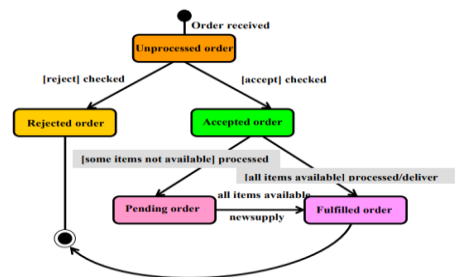
## Internal transitions

- An event causes only some internal actions to execute but does not lead to a change of state (state transition). In this case, all actions executed comprise the **internal transition**.
- For example, when one types on a keyboard, it responds by generating different character codes. However, unless the Caps Lock key is pressed, the state of the keyboard does not change (no state transition occurs).

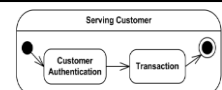
## Example

- Here is just an example of how an online ordering system might look like :
- On the event of an order being received, we transit from our initial state to Unprocessed order state.
- The unprocessed order is then checked.
- If the order is rejected, we transit to the Rejected Order state.
- If the order is accepted and we have the items available we transit to the fulfilled order state.
- However if the items are not available we transit to the Pending Order state until all the items become available.
- After the order is fulfilled, we transit to the final state.

## State chart diagram for an order object

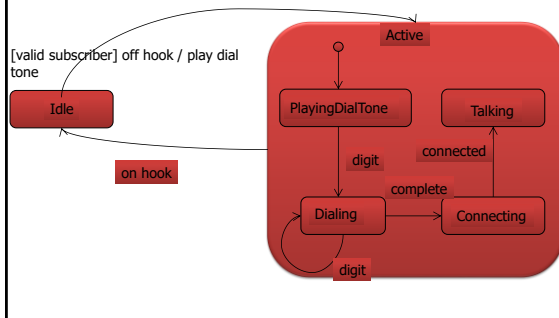


## Composite State

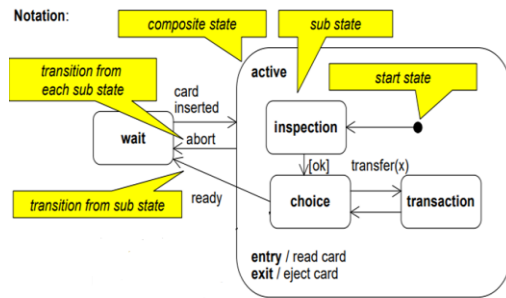


- A state can be refined hierarchically by composite states.
- Generally, **composite state** is defined as **state** that has substates (nested **states**). Substates could be sequential (disjoint) or concurrent (orthogonal).
- A state allows nesting to contain substates. A substate inherits the transitions of its superstate (the enclosing state).
  - Within the *Active* state, and no matter what substate the object is in, if the *on hook* event occurs, a transition to the *idle* state occurs.

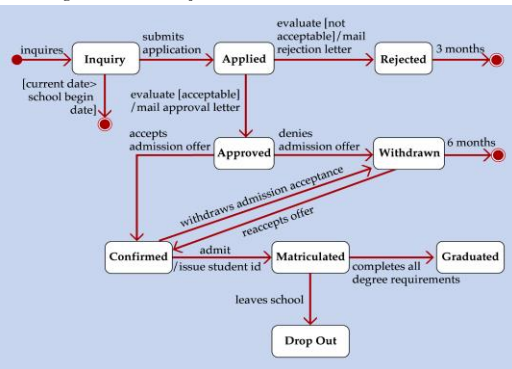
## Composite States/Nested State



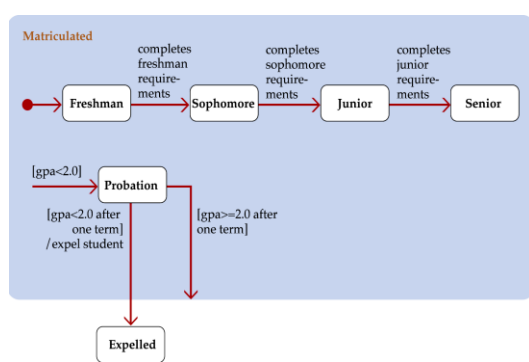
## Composite State



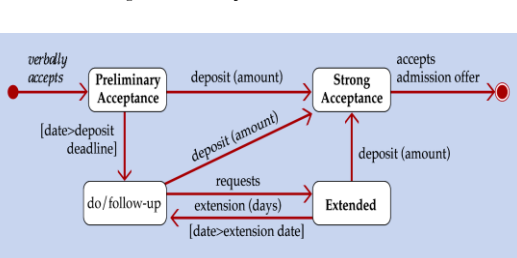
State Diagram for Student Object



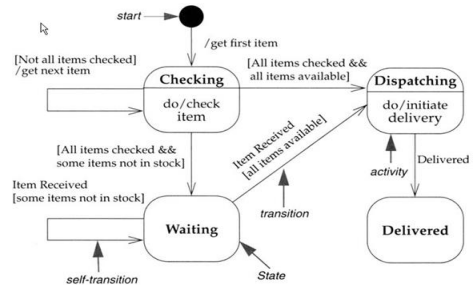
Nested State Diagram for Matriculated State of Student Object



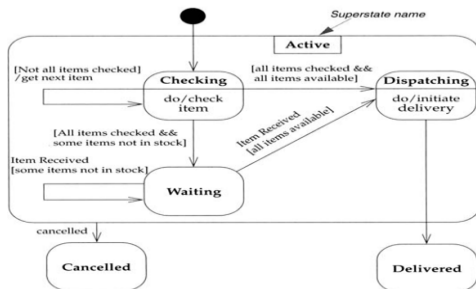
\*\*Nested State Diagram for the Accepts Admission Offer Event



## Order Management State Chart

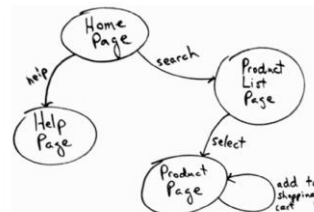


## Order Management State Chart

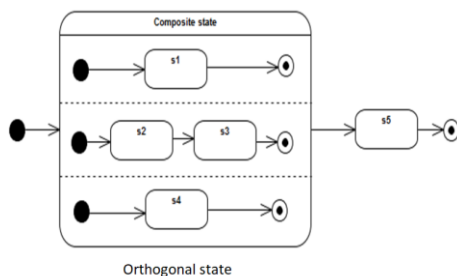


## UI Navigation Modeling with state machine diagram

Applying a state machine to Web page navigation modeling.



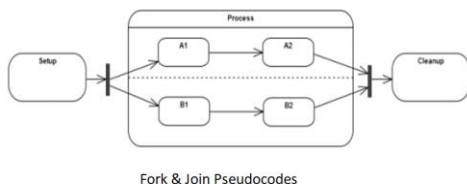
## Orthogonal State



## Concurrency

- Concurrency can be shown explicitly using fork and join pseudostates.
- A fork is represented by a bar with one or more outgoing arrows terminating on orthogonal regions (i.e. states in different regions);
- A join merges one or more transitions.
- Concurrent substates specify two or more state machines that execute in parallel in the context of the enclosing object
- Execution of these concurrent substates continues in parallel. These substates wait for each other to finish to joins back into one flow

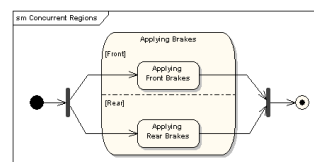
## Orthogonal States – Concurrency



Fork & Join Pseudocodes

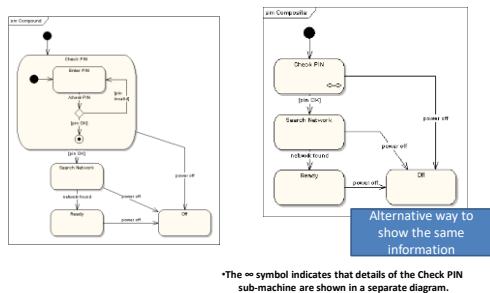
## State Machine Diagrams-Concurrent

- **Concurrent Regions** - A state may be divided into regions containing sub-states that exist and execute concurrently. The example below shows that within the state "Applying Brakes", the front and rear brakes will be operating simultaneously and independently.



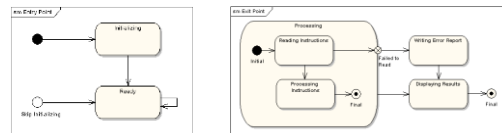
## State Machine Diagrams

- Compound States** - A state machine diagram may include sub-machine diagrams, as in the example below.



## State Machine Diagrams

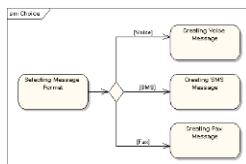
- Entry Point** - Sometimes you won't want to enter a sub-machine at the normal initial state. For example, in the following sub-machine it would be normal to begin in the "Initializing" state, but if for some reason it wasn't necessary to perform the initialization, it would be possible to begin in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

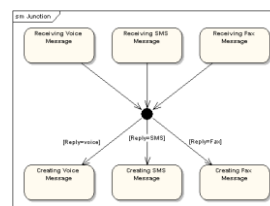
## State Machine Diagrams

- 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.



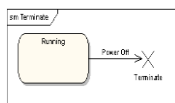
## State Machine Diagrams

- Junction Pseudo-State** - Junction pseudo-states are used to chain together multiple transitions. A single junction can have one or more incoming, and one or more outgoing, transitions; a guard can be applied to each transition. Junctions are semantic-free.



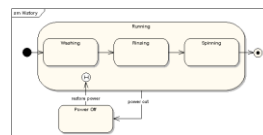
## State Machine Diagrams

- Terminate Pseudo-State** - Entering a terminate pseudo-state indicates that the lifeline of the state machine has ended. A terminate pseudo-state is notated as a cross.



## State Machine Diagrams

- History States** - A history state is used to remember the previous state of a state machine when it was interrupted. The following diagram illustrates the use of history states. The example is a state machine belonging to a washing machine.



## Internal transition

- **Internal Transition: transition from one state to itself.** Object handles event without changing its state.
- **Mutation Event: The initiator of a transition from one state to another, or for an internal transition, where the state remains the same.**

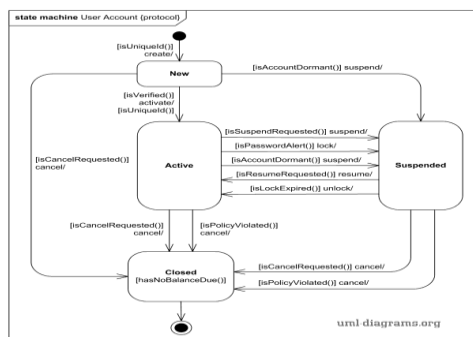
State  
state event

state Mutation  
event

## State Model Pitfalls

- Too detailed
- Badly drawn/organized
  - need for state model tool
- Not really a state model, but a flow chart
  - states are actions and transitions are conditions on existing values, like a control flow diagram
- Transitions should be made on the occurrence of events

## User Account Management



## READING

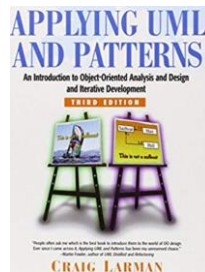
Chapter 29 – Applying UML and Pattern by Craig Larman 3<sup>rd</sup> Edition

Chapter 29. UML State Machine Diagrams and Modeling

No, no, you're not thinking, you're just being logical.  
Wesley Bohr

### Objectives

- Introduce UML state machine diagram notation, with examples, and various modeling applications.



## END OF TOPIC 13

-COMING UP!!!!!!

-Timing Diagrams  
-Implementation Diagrams  
-RUP  
-4+1 Model view