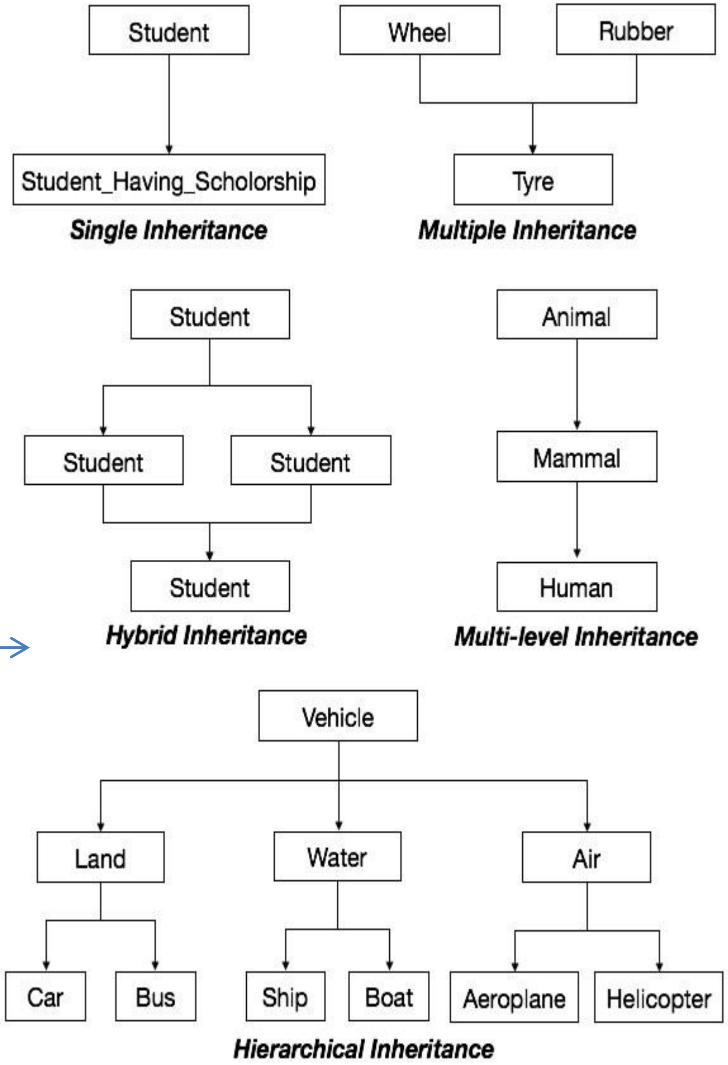
Object Oriented Analysis & Design

What is Object Model

- An object model helps describe or define a software/system in terms of objects and classes. It defines the interfaces or interactions between different models, inheritance, encapsulation and other object-oriented interfaces and features.
- 2 Basic Category
  - Document Object Model: A set of objects that provides a modelled representation of dynamic HTML (DHTML) and XHTML-based Web pages
  - Component Object Model: A proprietary Microsoft software architecture used to create software components

Object Model

- Objects & Class
- Encapsulation
  - Data Hiding
  - Massage passing
- Inheritance
  - Single Inheritance
  - Multiple Inheritance
  - Multilevel Inheritance
  - Hierarchical Inheritance
  - Hybrid Inheritance

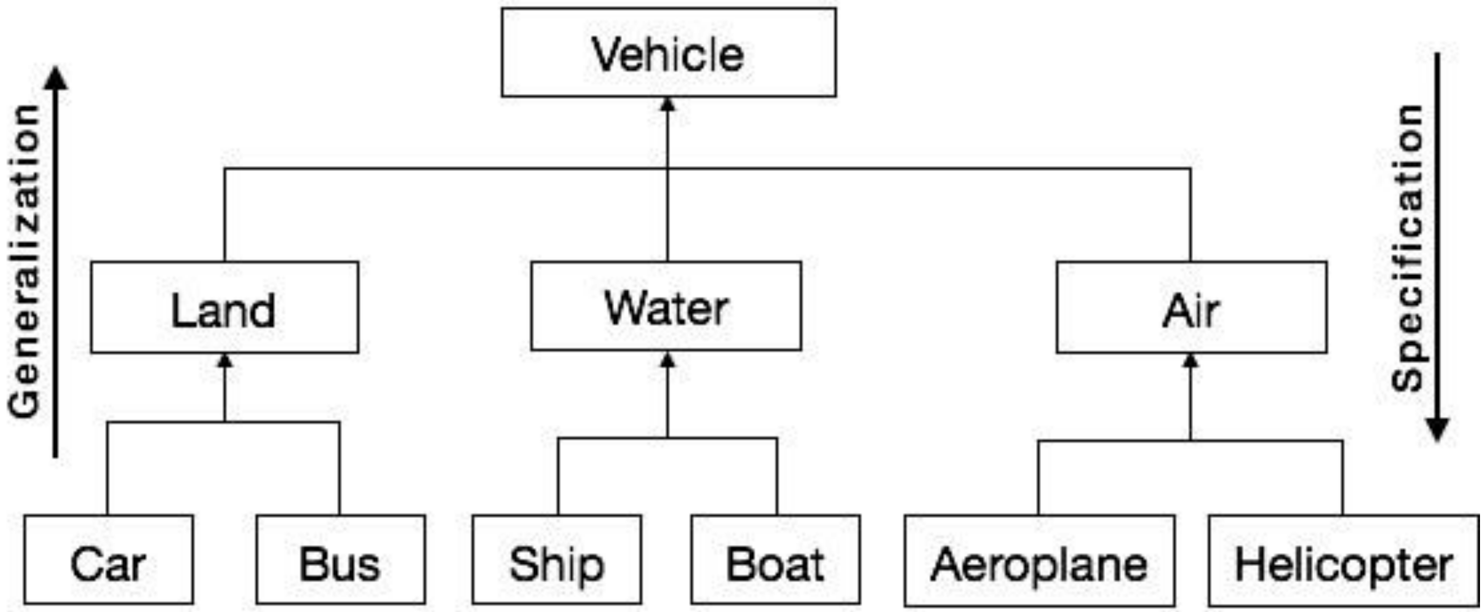


Contd.

- Abstraction
- Typing
- Polymorphism
- Generalization and Specialization
  - Generalization and specialization represent a hierarchy of relationships between classes, where subclasses inherit from super-classes.
- In Generalization, subclasses are combined to form a generalized super-class. It represents an "is – a – kind – of" relationship.
- e.g. "car ***is a kind of*** land vehicle"

Contd.

- Specialization is the reverse process of generalization.



Typing

- The object oriented languages are divided in to 2 category.
  - Strong type
  - Weak type
- In strongly typed languages, the compiler can detect when an object is being sent a message to which it does not respond. This can prevent run-time errors. earlier detection of errors speeds development
- better optimized code from compiler
- no run-time penalty for determining type but difficult to define collections of heterogenous objects
- E.g- c++
- In weak typing, messages may be sent to any class. The operation is checked only at the time of execution, as in the programming language Smalltalk.

Contd.

- Links and Association
  - Link Stands for a physical or conceptual connection between objects. A link depicts the relationship between two or more objects.
  - Association is a group of links having common structure and common behaviour. A link can be defined as an instance of an association.
- Degree of an Association
  - *Unary relationship* connects objects of the same class.
  - *Binary relationship* connects objects of two classes.
  - *Ternary relationship* connects objects of three or more classes.
- Cardinality Ratios of Associations
  - One–to–One
  - One–to–Many
  - Many–to–Many

Contd.

- Aggregation
  - It is a relationship among classes by which a class can be made up of any combination of objects of other classes. It is also referred as a “part–of” or “has–a” relationship, with the ability to navigate from the whole to its parts.
  - e.g - a car has–a motor
  - Physical containment − a computer is composed of monitor, CPU, mouse, keyboard, etc.
  - Conceptual containment − In share market, shareholder has–a share.

Modularity

- Modularity is the degree to which a system's components are made up of relatively independent components or parts which can be combined.

Meta Data & Meta Class

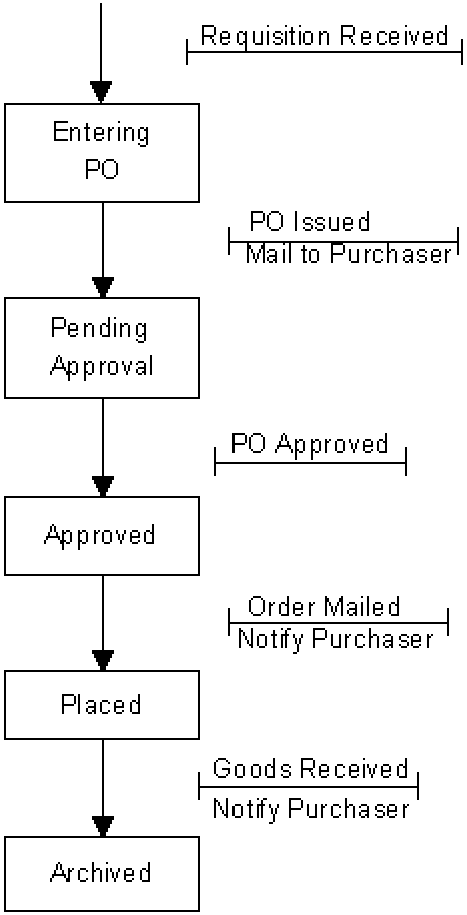
- A metaclass defines the behaviour of certain classes and their instances.
- if class A's parent class is class B, then A's metaclass's parent class is B's metaclass)
- Meta Data
- Metaclass objects do not behave differently (you cannot add class methods for a metaclass, so metaclass objects all have the same methods), they are all instances of the same class—the metaclass of the root class
- Ref: objective C

Benefits of Object Model

- It helps in faster development of software.
- It is easy to maintain.
- It supports relatively hassle-free upgrades.
- It enables reuse of objects, designs, and functions.
- It reduces development risks, particularly in integration of complex systems. etc.

Dynamic Model

- A dynamic model represents the behaviour of an object over time.
- It is used where the object's behaviour is best described as a set of states that occur in a defined sequence.
- The components of the dynamic model are:
  - **States**
  - **State transitions**
  - **Events**
  - **Actions**.
  - **Activities**



Functional Modelling

- The process of functional modelling can be visualized in the following steps −
  - Identify all the inputs and outputs
  - Construct data flow diagrams showing functional dependencies
  - State the purpose of each function
  - Identify constraints
  - Specify optimization criteria

Structured Analysis vs. Object Oriented Analysis

- The Structured Analysis/Structured Design (SASD) approach is the traditional approach of software development based upon the waterfall model. The phases of development of a system using SASD are −
  - Feasibility Study
  - Requirement Analysis and Specification
  - System Design
  - Implementation
  - Post-implementation Review

Advantages/Disadvantages of Object Oriented Analysis

Advantages	Disadvantages
Focuses on data rather than the procedures as in Structured Analysis.	Functionality is restricted within objects. This may pose a problem for systems which are intrinsically procedural or computational in nature.
The principles of encapsulation and data hiding help the developer to develop systems that cannot be tampered by other parts of the system.	It cannot identify which objects would generate an optimal system design.
The principles of encapsulation and data hiding help the developer to develop systems that cannot be tampered by other parts of the system.	The object-oriented models do not easily show the communications between the objects in the system.
It allows effective management of software complexity by the virtue of modularity.	All the interfaces between the objects cannot be represented in a single diagram.
It can be upgraded from small to large systems at a greater ease than in systems following structured analysis.	

Advantages/Disadvantages of Structured Analysis

Advantages	Disadvantages
As it follows a top-down approach in contrast to bottom-up approach of object-oriented analysis, it can be more easily comprehended than OOA.	In traditional structured analysis models, one phase should be completed before the next phase. This poses a problem in design, particularly if errors crop up or requirements change.
It is based upon functionality. The overall purpose is identified and then functional decomposition is done for developing the software. The emphasis not only gives a better understanding of the system but also generates more complete systems.	The initial cost of constructing the system is high, since the whole system needs to be designed at once leaving very little option to add functionality later.
The specifications in it are written in simple English language, and hence can be more easily analyzed by non-technical personnel.	It does not support reusability of code. So, the time and cost of development is inherently high.

Dynamic Modelling

- The dynamic model represents the time–dependent aspects of a system.
  - State, which is the situation at a particular condition during the lifetime of an object.
  - Transition, a change in the state
  - Event, an occurrence that triggers transitions
  - Action, an uninterrupted and atomic computation that occurs due to some event, and
  - Concurrency of transitions.

States and State Transitions

- It is a situation occurring for a finite time period in the lifetime of an object, in which it fulfils certain conditions, performs certain activities, or waits for certain events to occur. In state transition diagrams, a state is represented by rounded rectangles.
- Parts of a state
  - Name − A string differentiates one state from another. A state may not have any name.
  - Entry/Exit Actions − It denotes the activities performed on entering and on exiting the state.
  - Internal Transitions − The changes within a state that do not cause a change in the state.
  - Sub–states − States within states.

Contd.

- Initial and Final States
  - The initial and the final states are pseudo-states, and may not have the parts of a regular state except name. In state transition diagrams, the initial state is represented by a filled black circle. The final state is represented by a filled black circle encircled within another unfilled black circle.
- Transition

The transition gives the relationship between the first state and the new state. A transition is graphically represented by a solid directed arc from the source state to the destination state.

The five parts of a transition are −

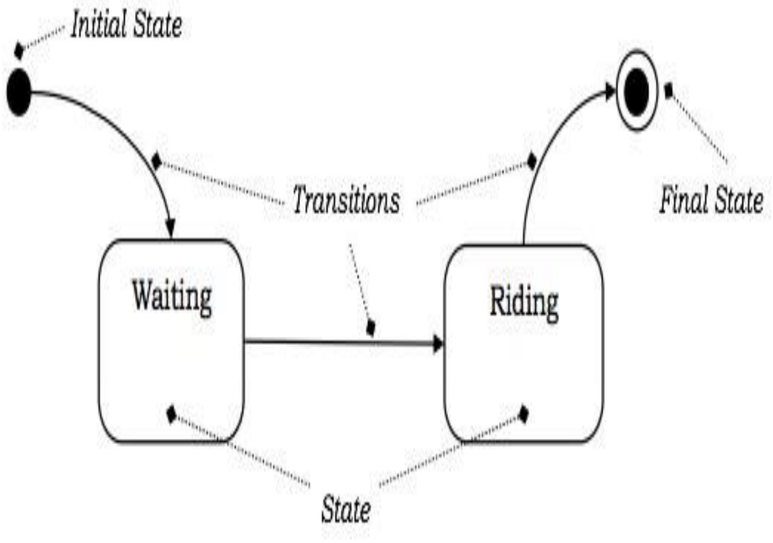
Source State − The state affected by the transition.

Event Trigger − The occurrence due to which an object in the source state undergoes a transition if the guard condition is satisfied.

Guard Condition − A Boolean expression which if True, causes a transition on receiving the event trigger.

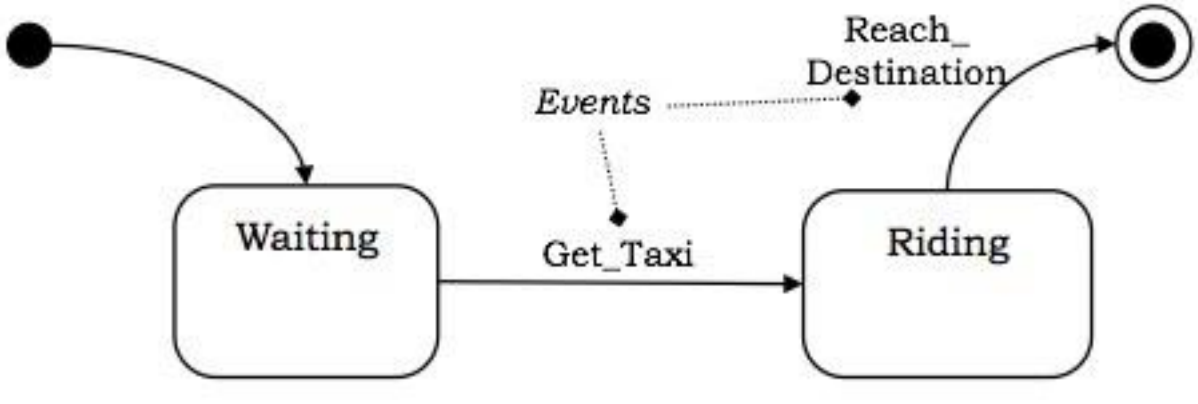
Action − An un-interruptible and atomic computation that occurs on the source object due to some event.

Target State − The destination state after completion of transition.



Events

- Events are some occurrences that can trigger state transition of an object or a group of objects.
- Events are generally associated with some actions.
- E.g. mouse click, key press, an interrupt, stack overflow, etc.
- Events that trigger transitions are written alongside the arc of transition in state diagrams.



Contd.

- External and Internal Events
  - External events are those events that pass from a user of the system to the objects within the system. e.g- mouseclick etc.
  - Internal events are those that pass from one object to another object within a system. e.g- stack overflow etc.
- Deferred Events
  - Events which are not handled immediately by the object in current state. The scheduled to be handled by the object in some different state at different time.
- Event Classes
  - A group of events with common structure or behaviour.

Actions

- Activity
  - Activity is an operation upon the states of an object that requires some time period. Activities are shown in activity diagrams that portray the flow from one activity to another.
- Action
  - An action is an atomic operation that executes as a result of certain events.
- Entry and Exit Actions
  - Entry action is the action that is executed on entering a state. The action that is executed while leaving a state, irrespective of the transition that led out of it, is called an exit action.
- Scenario
  - Scenario is a description of a specified sequence of actions.

Diagrams for Dynamic Modelling

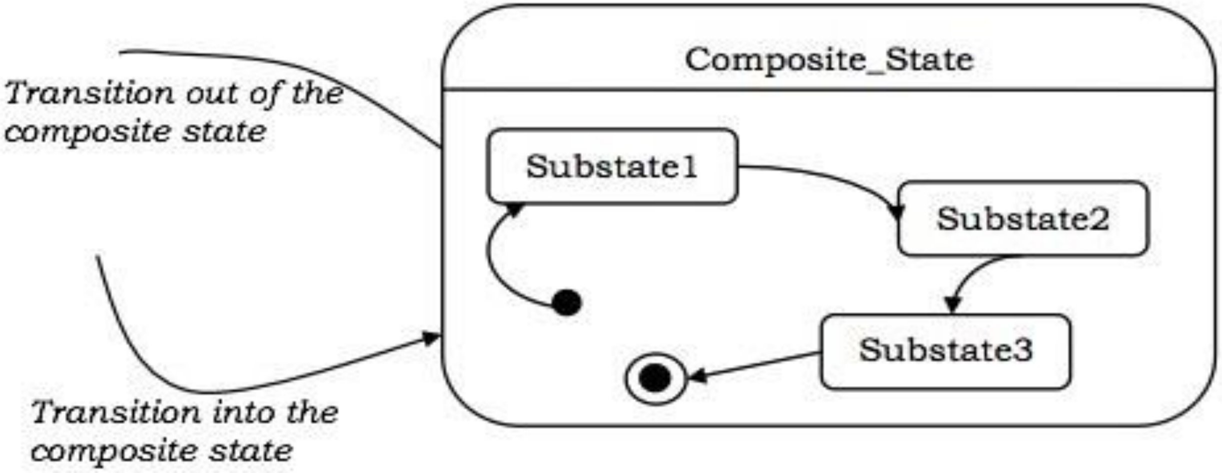
- Interaction Diagrams
  - Interaction diagrams describe the dynamic behaviour among different objects. an interaction models the behaviour of a group of interrelated objects.
    - Sequence Diagram − It represents the temporal ordering of messages in a tabular manner.
    - Collaboration Diagram − It represents the structural organization of objects that send and receive messages through vertices and arcs.
- State Transition Diagram
  - State transition diagrams or state machines describe the dynamic behaviour of a single object.

Concurrency of Events

- System Concurrency
  - The overall system is modelled as the aggregation of state machines, where each state machine executes concurrently with others.
- Concurrency within an Object
  - An object may have states that are composed of sub-states, and concurrent events may occur in each of the sub-states.
- Simple and Composite States
  - A simple state has no sub-structure.
  - A state that has simpler states nested inside it is called a composite state.

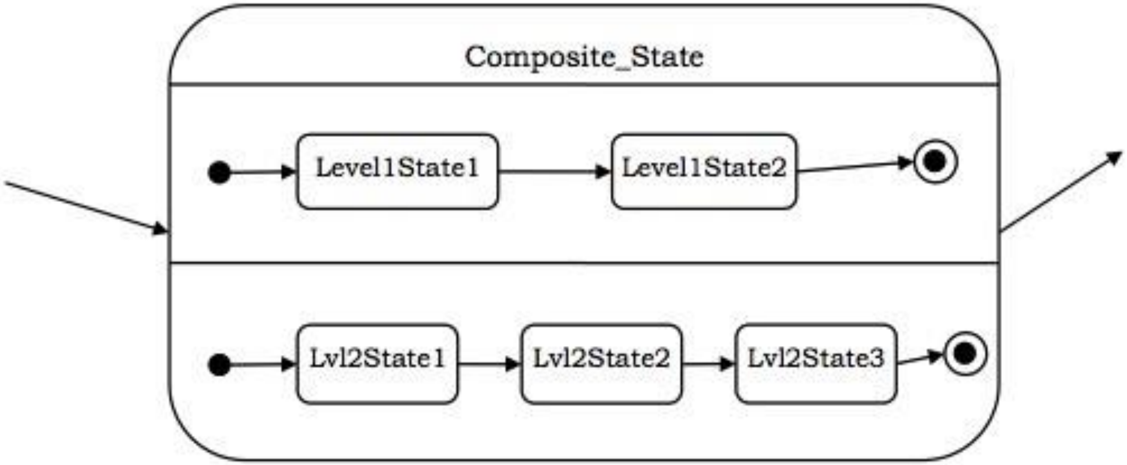
Contd.

- Sequential Sub-states
  - In sequential sub-states, the control of execution passes from one sub-state to another sub-state one after another in a sequential manner.


Composite_State
Transition out of the composite state
Substate1
Substate2
Substate3
Transition into the composite state

Contd.

- Concurrent Sub-states
  - the sub-states execute in parallel, or in other words, each state has concurrently executing state machines within it. Each of the state machines has its own initial and final states.



Data Flow Diagrams

Advantages	Disadvantages
DFDs depict the boundaries of a system and hence are helpful in portraying the relationship between the external objects and the processes within the system.	DFDs take a long time to create, which may not be feasible for practical purposes.
They help the users to have a knowledge about the system.	DFDs do not provide any information about the time-dependent behavior, i.e., they do not specify when the transformations are done.
The graphical representation serves as a blueprint for the programmers to develop a system.	They do not throw any light on the frequency of computations or the reasons for computations.
DFDs provide detailed information about the system processes.	The preparation of DFDs is a complex process that needs considerable expertise. Also, it is difficult for a non-technical person to understand.
They are used as a part of the system documentation.	The method of preparation is subjective and leaves ample scope to be imprecise.

Relationship between Object, Dynamic, and Functional Models

- Object modelling develops the static structure of the software system in terms of objects. Thus it shows the “doers” of a system.
- Dynamic Modelling develops the temporal behaviour of the objects in response to external events. It shows the sequences of operations performed on the objects.
- Functional model gives an overview of what the system should do.

Functional Model and Object Model

- The four main parts of a Functional Model in terms of object model are −
  - Process − Processes imply the methods of the objects that need to be implemented.
  - Actors − Actors are the objects in the object model.
  - Data Stores − These are either objects in the object model or attributes of objects.
  - Data Flows − Data flows to or from actors represent operations on or by objects. Data flows to or from data stores represent queries or updates.

Functional Model and Dynamic Model

- The dynamic model states when the operations are to be performed
- Functional model states how they are performed and which arguments are needed.
- As actors are active objects, the dynamic model has to specify when it acts.
- The data stores are passive objects and they only respond to updates and queries; therefore the dynamic model need not specify when they act.

Object Model and Dynamic Model

- The dynamic model shows the status of the objects and the operations performed on the occurrences of events and the subsequent changes in states.
- The state of the object as a result of the changes is shown in the object model.