

Object Oriented Approach

In the object-oriented approach, the focus is on capturing the structure and behavior of information systems into small modules that combines both data and process. The main aim of Object Oriented Design (OOD) is to improve the quality and productivity of system analysis and design by making it more usable.

In analysis phase, OO models are used to fill the gap between problem and solution. It performs well in situation where systems are undergoing continuous design, adaption, and maintenance. It identifies the objects in problem domain, classifying them in terms of data and behavior.

The OO model is beneficial in the following ways –

- It facilitates changes in the system at low cost.
- It promotes the reuse of components.
- It simplifies the problem of integrating components to configure large system.
- It simplifies the design of distributed systems.

Elements of Object-Oriented System

Let us go through the characteristics of OO System –

- **Objects** – An object is something that exists within problem domain and can be identified by data (attribute) or behavior. All tangible entities (student, patient) and some intangible entities (bank account) are modeled as object.
- **Attributes** – They describe information about the object.
- **Behavior** – It specifies what the object can do. It defines the operation performed on objects.
- **Class** – A class encapsulates the data and its behavior. Objects with similar meaning and purpose grouped together as class.
- **Methods** – Methods determine the behavior of a class. They are nothing more than an action that an object can perform.
- **Message** – A message is a function or procedure call from one object to another. They are information sent to objects to trigger methods. Essentially, a message is a function or procedure call from one object to another.

Features of Object-Oriented System

An object-oriented system comes with several great features which are discussed below.

Encapsulation

Encapsulation is a process of information hiding. It is simply the combination of process and data into a single entity. Data of an object is hidden from the rest of the system and available only through the services of the class. It allows improvement or modification of methods used by objects without affecting other parts of a system.

Abstraction

It is a process of taking or selecting necessary method and attributes to specify the object. It focuses on essential characteristics of an object relative to perspective of user.

Relationships

All the classes in the system are related with each other. The objects do not exist in isolation, they exist in relationship with other objects.

There are three types of object relationships –

- **Aggregation** – It indicates relationship between a whole and its parts.
- **Association** – In this, two classes are related or connected in some way such as one class works with another to perform a task or one class acts upon other class.
- **Generalization** – The child class is based on parent class. It indicates that two classes are similar but have some differences.

Inheritance

Inheritance is a great feature that allows to create sub-classes from an existing class by inheriting the attributes and/or operations of existing classes.

Polymorphism and Dynamic Binding

Polymorphism is the ability to take on many different forms. It applies to both objects and operations. A polymorphic object is one whose true type hides within a super or parent class.

In polymorphic operation, the operation may be carried out differently by different classes of objects. It allows us to manipulate objects of different classes by knowing only their common properties.

Structured Approach Vs. Object-Oriented Approach

The following table explains how the object-oriented approach differs from the traditional structured approach –

Structured Approach	Object Oriented Approach
It works with Top-down approach.	It works with Bottom-up approach.
Program is divided into number of submodules or functions.	Program is organized by having number of classes and objects.
Function call is used.	Message passing is used.
Software reuse is not possible.	Reusability is possible.
Structured design programming usually left until end phases.	Object oriented design programming done concurrently with other phases.
Structured Design is more suitable for offshoring.	It is suitable for in-house development.
It shows clear transition from design to implementation.	Not so clear transition from design to implementation.
It is suitable for real time system, embedded system and projects where objects are not the most useful level of abstraction.	It is suitable for most business applications, game development projects, which are expected to customize or extended.
DFD & E-R diagram model the data.	Class diagram, sequence diagram, state chart diagram, and use cases all contribute.
In this, projects can be managed easily due to clearly identifiable phases.	In this approach, projects can be difficult to manage due to uncertain transitions between phase.

Unified Modeling Language (UML)

UML is a visual language that lets you to model processes, software, and systems to express the design of system architecture. It is a standard language for designing and documenting a system in an object oriented manner that allow technical architects to communicate with developer.

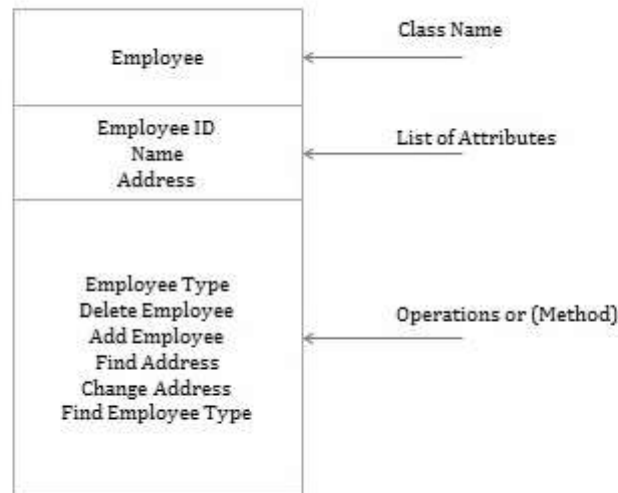
It is defined as set of specifications created and distributed by Object Management Group. UML is extensible and scalable.

The objective of UML is to provide a common vocabulary of object-oriented terms and diagramming techniques that is rich enough to model any systems development project from analysis through implementation.

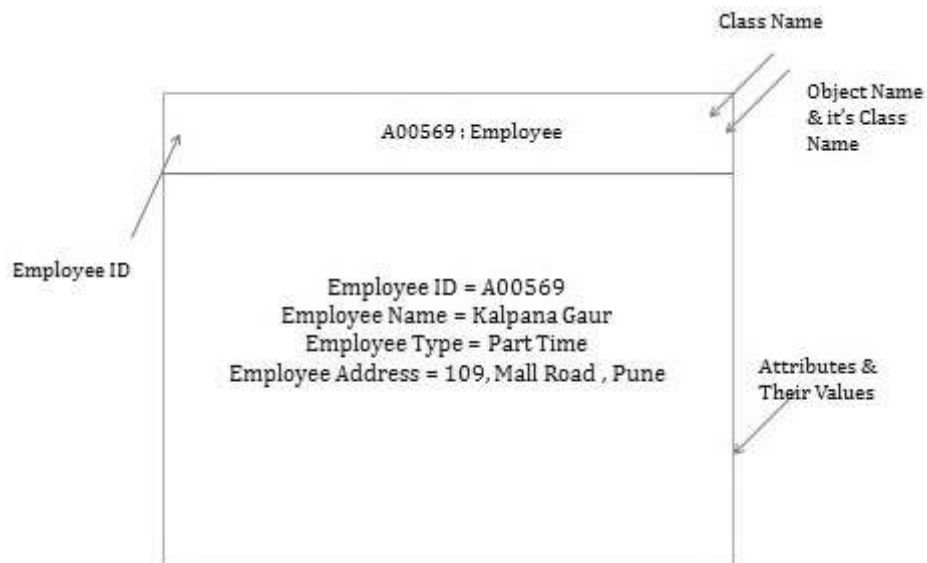
UML is made up of –

- **Diagrams** – It is a pictorial representations of process, system, or some part of it.
- **Notations** – It consists of elements that work together in a diagram such as connectors, symbols, notes, etc.

Example of UML Notation for class



Instance diagram-UML notation



Operations Performed on Objects

The following operations are performed on the objects –

- **Constructor/Destructor** – Creating new instances of a class and deleting existing instances of a class. For example, adding a new employee.

- **Query** – Accessing state without changing value, has no side effects. For example, finding address of a particular employee.
- **Update** – Changes value of one or more attributes & affect state of object For example, changing the address of an employee.

Uses of UML

UML is quite useful for the following purposes –

- Modeling the business process
- Describing the system architecture
- Showing the application structure
- Capturing the system behavior
- Modeling the data structure
- Building the detailed specifications of the system
- Sketching the ideas
- Generating the program code

Static Models

Static models show the structural characteristics of a system, describe its system structure, and emphasize on the parts that make up the system.

- They are used to define class names, attributes, methods, signature, and packages.
- UML diagrams that represent static model include class diagram, object diagram, and use case diagram.

Dynamic Models

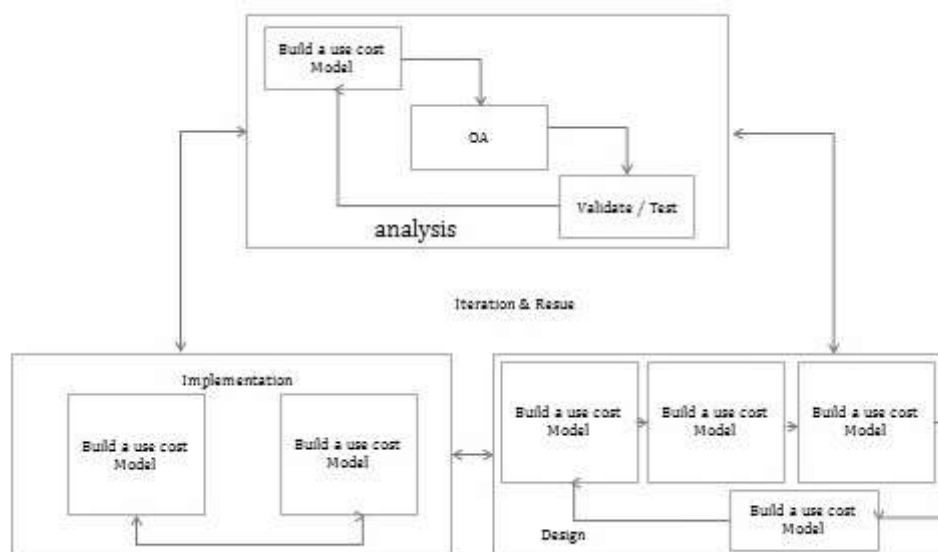
Dynamic models show the behavioral characteristics of a system, i.e., how the system behaves in response to external events.

- Dynamic models identify the object needed and how they work together through methods and messages.
- They are used to design the logic and behavior of system.
- UML diagrams represent dynamic model include sequence diagram, communication diagram, state diagram, activity diagram.

Object Oriented System Development Life Cycle

It consists of three macro processes –

- Object Oriented Analysis (OOA)
- Object oriented design (OOD)
- Object oriented Implementation (OOI)



Object Oriented Systems Development Activities

Object-oriented systems development includes the following stages –

- Object-oriented analysis
- Object-oriented design
- Prototyping
- Implementation
- Incremental testing

Object-Oriented Analysis

This phase concerns with determining the system requirements and to understand the system requirements build a **use-case model**. A use-case is a scenario to describe the interaction between user and computer system. This model represents the user needs or user view of system.

It also includes identifying the classes and their relationships to the other classes in the problem domain, that make up an application.

Object-Oriented Design

The objective of this phase is to design and refine the classes, attributes, methods, and structures that are identified during the analysis phase, user interface, and data access. This phase also

identifies and defines the additional classes or objects that support implementation of the requirement.

Prototyping

Prototyping enables to fully understand how easy or difficult it will be to implement some of the features of the system.

It can also give users a chance to comment on the usability and usefulness of the design. It can further define a use-case and make use-case modeling much easier.

Implementation

It uses either Component-Based Development (CBD) or Rapid Application Development (RAD).

Component-based development (CBD)

CODD is an industrialized approach to the software development process using various range of technologies like CASE tools. Application development moves from custom development to assembly of pre-built, pre-tested, reusable software components that operate with each other. A CBD developer can assemble components to construct a complete software system.

Rapid Application Development (RAD)

RAD is a set of tools and techniques that can be used to build an application faster than typically possible with traditional methods. It does not replace SDLC but complements it, since it focuses more on process description and can be combined perfectly with the object oriented approach.

Its task is to build the application quickly and incrementally implement the user requirements design through tools such as visual basic, power builder, etc.

Incremental Testing

Software development and all of its activities including testing are an iterative process. Therefore, it can be a costly affair if we wait to test a product only after its complete development. Here incremental testing comes into picture wherein the product is tested during various stages of its development.
