Object-oriented system can be modeled using modeling diagrams which collectively form 'Unified modeling language'. There are 2 types of diagrams in UML - Structural diagrams which represent the components of the system. It includes class diagram, object diagram and package diagram etc. and Behavioral diagrams that represents interaction between objects/components of the system.

**Use case diagram**: It is a behavioral diagram which can be used to represent the functionality provided by the system to the Actors which are the external users of the system.

To illustrate a use case on a use case diagram, we draw an oval in the middle of the diagram and put the name of the use case in the center of the oval. To show an actor (indicating a system user) on a use-case diagram, we draw a stick figure to the left or right of the diagram.

The different components of the use case diagram are:

System boundary: A system boundary defines the scope and limits of the system. It is shown as a rectangle that spans all use cases of the system.

Actors: An actor interacts with a use case of the system. For example, in a banking system, the customer is one of the actors along with bankers.

Use Case: The use case should list the discrete business functionality specified in the problem statement.

Include: Include relationship represents an invocation of one use case by another use case. From a coding perspective, it is like one function being called by another function.

Extend: It's about creating a use case which extends the base use case like search product/item can inherit base search class.

**Class diagram**: It denotes static structures of a system. A class diagram is defined by a rectangular block which is divided into 3 sections- class name, attributes and methods. Different type of relationship can exist between classes:

Association: It simply means that there is a communication between 2 classes.

Aggregation: It is a type of association which is used to model a “whole to its parts” relationship. In a basic aggregation relationship, the lifecycle of a PART class is independent of the WHOLE class’s lifecycle. It implies a relationship where the child can exist independently of the parent. Aircraft can exist without Airline.

Composition: In this type of association, contained class lifecycle is dependent on container class. e.g. WeeklySchedule class is dependent on Flight class. If Flight is destroyed, WeeklySchedule is destroyed automatically.



**Sequence Diagrams**: Sequence diagrams describe interactions among classes in terms of an exchange of messages over time and are used to explore the logic of complex operations, functions or procedures. In this diagram, the sequence of interactions between the objects is represented in a step-by-step manner.

**Activity Diagram**: Activity Diagram represents the flow of control in a given functionality.