Use Case Diagram –

It is a diagrammatic way of representing the functionalities of software in a horizontal way rather than representing individual factors. Use case diagrams are used to gather the requirements of a system including external and internal influences. These requirements are usually design requirements. So when a system is analyzed to gather its functionalities, use case are prepared and actors are identified.

So, in brief purpose of use case diagram can be as follows –

- Used to gather the requirements of a system/software.
- Used to get outside view of system.
- Identify external and internal factors influencing the system.
- Show the interactions among the requirements and actors.

Components used in Use Case Diagram –

1. Actors-

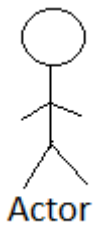
The actors are the one who uses or interacts with the system or software functionalities. Actors are usually designated using nouns. There are broadly two types of actors namely,

1. Primary Actors

2. Secondary Actors

The **Primary Actors** are those actors who uses the functionalities of primary functions of a software or system. For example- For a Library Management Software, the students can be considered as primary actors.

While, the **Secondary Actors** are the one who make some changes in the system, or assign the values of some factors of the functionalities of system. For example- For a Library Management Software, the staff or library members can be considered as secondary actors, because they have the access to make changes in functionalities, like book quota limit, book issue duration, etc.



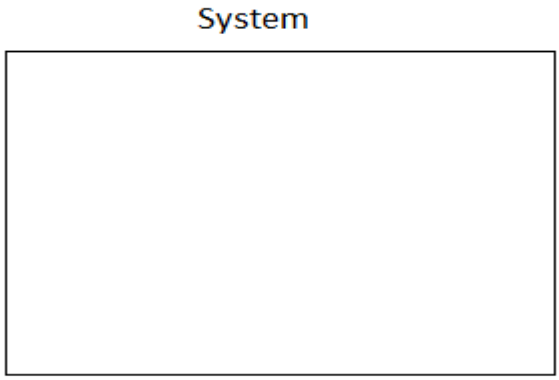
2. System-

A boundary inside which there are all functions and functionalities of a system.

For a system,

No actors are inside the boundary of system, and;

No functionalities are outside boundary.



3. **Use Case Functions-**

All modules represent use cases; i.e., they show all the functions of the software. All use case functions are represented using verbs.

4. Connections-

4.1- Association- Used to show connections between actors and use cases, or between use cases and use cases.

4.2- Generalization- Used to represent the generalized behavior of actors and use cases. There cannot be generalization between actors and use cases.

4.3- Include- These are the links that signify those use cases that are compulsorily necessary to perform certain actions. That is, <include> link shows dependency. For example, for "reservation", check for "seat availability" is necessary. So we use an "include" link here.

4.4- Extends- This are the links that denote those functionalities which are not necessarily important to perform certain action. For example, for a "login" functionality, "Forgot password" is not a necessary factor; so we can represent it using "extends" link.

Importance Of Use Case-

1. To identify functions and how actors interact with them.

2. For presenting a high level view of system.

3. To identify external and internal factors.

Activity Diagram-

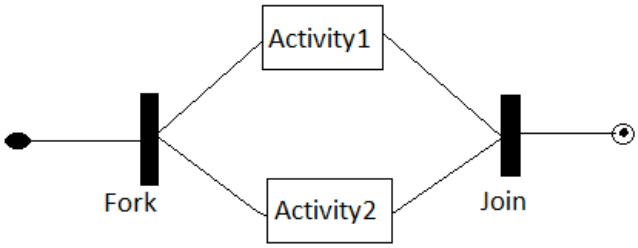
Activity Diagram denotes flow of activity in a system. It is basically a flow chart to represent the flow from one activity to another activity. The activity can be described as an operation of the system. Activity diagram deals with all types of flow control by using different elements which include…

Components used in Activity Diagram –

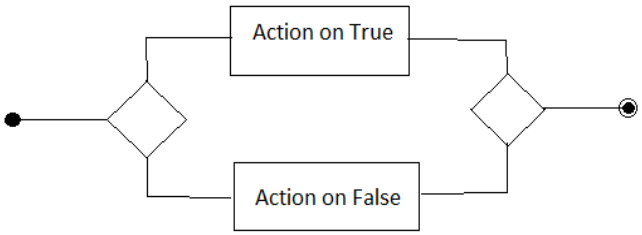
1. Name of Activity- It denotes the name of the various activities as performed by the software.

2. Start/End nodes- The start node signify the starting of an activity, and similarly the stop node denote the end of the activity.

3. Fork and Join nodes- When we want to perform two or more activities concurrently, we use fork & join nodes. For example, for online payment, payment options selection, i.e., via COD or Credit Card can be considered as two concurrent activities and thus can be represented by "fork & join" node.



4. Decision and Merge node- When there is a need of decision making out of two activities, we use decision and merge node to represent the activities; in which one branch correspond to True and other to False.
For a decision and merge node, the decision does not necessarily require merge.
For example, if for "login", correct password will correspond to the "logged in" node; while incorrect password will correspond to the "login" node again.



Data Flow Diagram-(DFD)

Data flow diagram is also known as Bubble chart, Process model, Information flow model. DFD provides the input-process-output view of a system. It shows the flow of data. It combines the information domain and functional domain. DFD should not have control elements or any information about the processing time.
Data Flow Diagram is prepared in various levels. In zero level DFD the software overview is presented showing relation between system and actor. The level-1 DFD represents all the functional requirements of software. The further level DFDs, represents the exploration of the functional requirements.

Components used in Data Flow Diagram-(DFD)

1. Process- These represent the functional requirements or the decomposition of the functional requirements. It denotes the transportation of input data to produce output data.

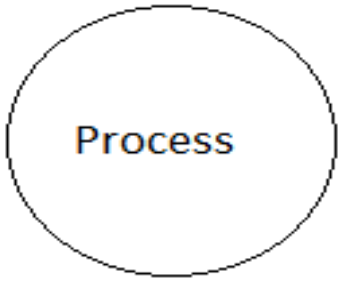
2. Data Flow- It is denoted by arrows which connect two processes. The data that flows across the processes is designated over the data flow arrow.

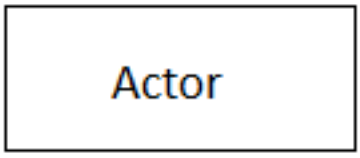
3. Data Store- It is central database representing data storage of the necessary details in an organized manner. Data store can be database, file, etc. which can store details such as login name, password, etc.

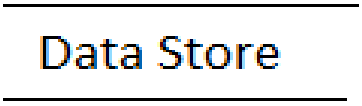
4. Actor- An actor is one which uses or interacts with system.

5. Data Updation, Storage- Used for storing data, and is represented by incoming arrow to data store.

6. Data Retrieval- Used for fetching data, and is represented by outgoing arrow from data store.







Rules –

Data cannot flow from-

1. External entity to External entity.
2. Actor to data store.
3. Data store to actor.
4. Data store to data store.

Data can flow from-

1. Process to process.
2. Actor to process.
3. Process to actor.
4. Process to data store and vice versa.