Use Case Diagram

Introduction:

- A use case diagram describes how a system interacts with outside actors.
- It is a graphical representation of the interaction among the elements and system.
- Each use case representation a piece of functionality that a system provides to its user.
- Use case identifies the functionality of a system.
- Use case diagram allows for the specification of higher level user goals that the system must carry out.
- These goals are not necessarily to tasks or actions, but can be more general required functionality of the system.
- You can apply use case to capture the intended behavior of the system you are developing, without having to specify how that behavior is implemented.
- A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case.
- A use case diagram contains four components.
    i. The boundary, which defines the system of interest in relation to the world around it.
    ii. The actors, usually individuals involved with the system defined according to their roles.
    iii. The use cases, which the specific roles are played by the actors within and around the system.
    iv. The relationships between and among the actors and the use cases.

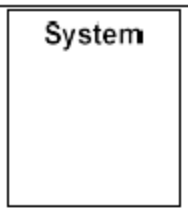
Purpose:

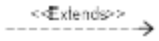
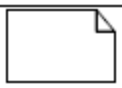
- The main purpose of the use case diagram is to capture the dynamic aspect of a system.
- Use case diagram shows, what software is suppose to do from user point of view.
- It describes the behavior of system from user's point.
- It provides functional description of system and its major processes.
- Use case diagram defines the scope of the system you are building.

When to Use: Use Cases Diagrams

- Use cases are used in almost every project.
- They are helpful in exposing requirements and planning the project.
- During the initial stage of a project most use cases should be defined.

Use Case Diagram Notations:

No.	Name	Notation	Description
1	System boundary		The scope of a system can be represented by a system boundary. The use cases of the system are placed inside the system boundary, while the actors who interact with the system are put outside the system. The use cases in the system make up the total requirements of the system.
2	Use case		A use case represents a user goal that can be achieved by accessing the system or software application.
3	Actor		Actors are the entities that interact with a system. Although in most cases, actors are used to represent the users of system, actors can actually be anything that needs to exchange information with the system. So an actor may be people, computer hardware, other systems, etc. Note that actor represent a role that a user can play, but not a specific user.
4	Association		Actor and use case can be associated to indicate that the actor participates in that use case. Therefore, an association corresponds to a sequence of actions between the actor and use case in achieving the use case.
5	Generalization		A generalization relationship is used to represent inheritance relationship between model elements of same type.
6	Include		An include relationship specifies how the behavior for the inclusion use case is inserted into the behavior defined for the base use case.

7	Extends		An extend relationship specifies how the behavior of the extension use case can be inserted into the behavior defined for the base use case.
8	Constraint		Show condition exists between actors an activity.
9	Package		Package is defined as collection of classes. Classes are unified together using a package.
10	Interface		Interface is used to connect package and use-case. Head is linked with package and tail linked with use-case.
11	Note		Note is generally used to write comment in use-case diagram.
12	Anchor		Anchor is used to connect a note the use case in use case diagram

Class Diagram

Introduction

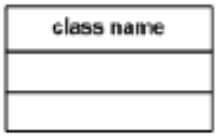
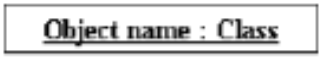
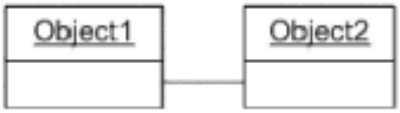
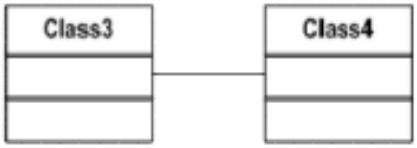
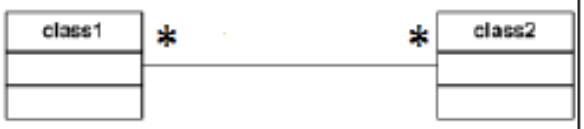
- The class diagram is a static diagram.
- A class model captures the static structure of a system by characterizing the objects in the system, the relationship between the objects, and the attributes and operations for each class of objects.
- The class diagram can be mapped directly with object oriented languages.
- The class model is the most important amonethe three models.
- Class diagram provide a graphical notation for modeling classes and their relationship.
- They are concise, easy to understand, and work well in practice.
- Class diagrams are the backbone of almost every object-oriented method including UML.
- They describe the static structure of a system.

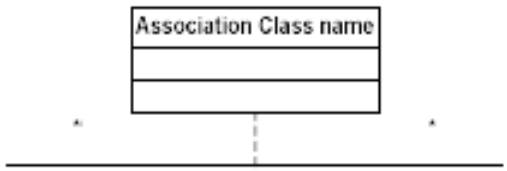
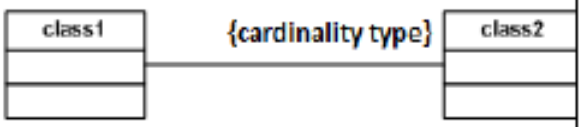
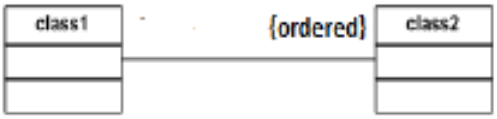
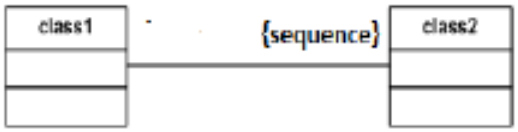
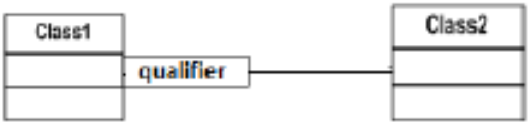
Purpose

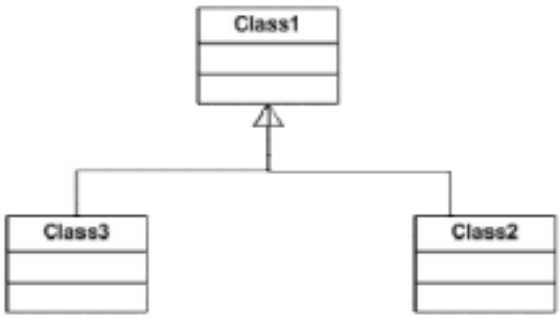
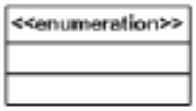
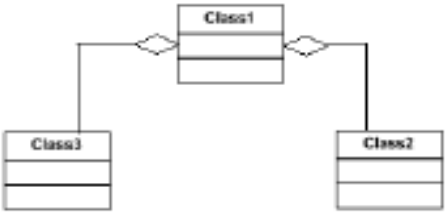
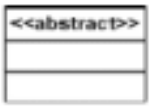
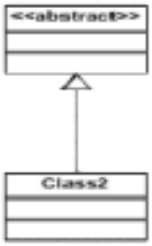
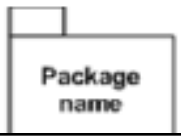
- Analysis and design of the static view of an application.
- Describe responsibilities of a system.
- Base for component and deployment diagrams.

When to use : Class Diagram

- Useful for Forward and Reverse engineering.
- Class diagrams are useful both for abstract modeling and for designing actual programs.
- Developer uses class diagram for implementation decision.
- Business analysts can use class diagrams to model systems from the business perspective.

Sr. No.	Name	Symbol	Meaning
1.	Class		Class is an entity of the class diagram. It describes a group of objects with same properties & behavior.
2.	Object		An object is an instance or occurrence of a class.
3.	Link		A link is a physical or conceptual connection among objects
4.	Association		An association is a description of a links with common structure & common semantics.
5.	Multiplicity	Ex. 1 to 1 1 to * * to * * to 1 1 to 0....2 	Multiplicity specifies the number of instances of one class that may relate to a single instance of an associated class. It is a constraint on the cardinality of a set.

6.	Association class		It is an association that is a class which describes the association with attributes.
7.	cardinality		It describes the count of elements from collection.
8.	ordering		It is used to indicate an ordered set of objects with no duplication allowed.
9.	bag		A bag is a collection of unordered elements with duplicates allowed.
10.	sequence		A sequence is an ordered collection of elements with duplicates allowed.
11.	qualified association		Qualification increases the precision of a model. It is used to avoid many to many multiplicities and it converts into one to one multiplicity.

12.	generalization		Generalization organizes classes by their super-class and sub-class relationship.
13.	enumeration		An enumeration is a data type that has a finite set of values.
14.	aggregation		It is a strong form of association in which an aggregate object is made of constituent parts.
15.	composition		It is a form of aggregation. Composition implies ownership of the parts by the whole.
16.	Abstract class		It is a class that has no direct instances.
17.	Concrete class		It is a class that is intangible; it can have direct instances. Class-2 is example of concrete class
18.	package		A package is a group of elements with common theme.

State Chart Diagram

Introduction

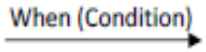
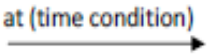
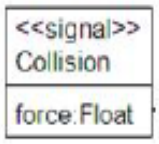
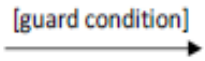
- *A **state diagram** is a graph in which nodes correspond to states and directed arcs correspond to transitions labeled with event names.*
- A state diagram combines states and events in the form of a network to model all possible object states during its life cycle, helping to visualize how an object responds to different stimuli.
- A state diagram is a graph whose nodes are states and whose directed arcs are transitions between states.
- A state diagram specifies the state sequence caused by event sequence.
- State names must be unique within the scope of a state diagram.
- All objects in a class execute the state diagram for that class, which models their common behavior.
- We can implement state diagrams by direct interpretation or by converting the semantics into equivalent programming code.

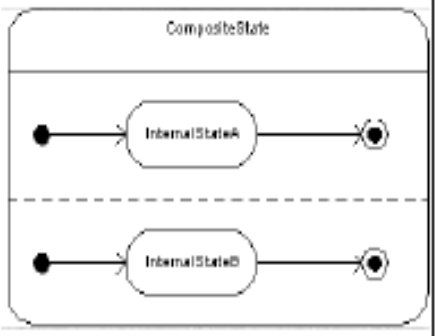
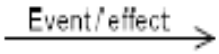
Purpose

- The state model describes those aspects of objects concerned with time and the sequencing of operations events that mark changes, states that define the context for events, and the organization of events and states.
- They are used to give an abstract description of the behavior of a system.
- It provides direction and guidance to the individual counties within the states.
- It specifies the possible states, what transitions are allowed between states.
- It describes the common behavior for the objects in a class and each object changes its behavior from one state to another.
- It is used to describe the dependence of the functionality on the state of the system that is how the functionality of an object depends on its state and how its state changes as a result of the events that it receives.
- It describes dynamic behavior of the objects of the system.

When to use: State Diagram

- They are perfectly useful to model behavior in real time system.
- Each state represents a named condition during the life of an object during which it satisfies some condition or waits for some event.
- It determines how objects of that class react to events.
- For each object state, it determines what actions the object will perform when it receives an event.

No.	Name	Notation	Description
1	State		A state is an abstraction of the values and links of an object. State models a situation during which some (usually implicit) invariant condition holds.
2	Transition		A transition is a directed relationship between a source state and a target state. It may be part of a compound transition, which takes the state machine from one state configuration to another
3	Event		A transition is an instantaneous change from one to another state
4	Change Event		A change in value of a Boolean expression
5	Time Event		The arrival of an absolute time or the passage of a relative amount of time
6	Signal Event		Receipt of an explicit, named, asynchronous communication among objects.
7	Guarded transition		A guard condition is a Boolean expression that must be true in order for a transition to occur.
8	Do activity		A do activity an activity that continuous for extended time within state.

9	Entry activity		An state is entered by any incoming transition the entry activity is performed
10	Exit activity		When the state is exited by any outgoing transition the exit activity is performed
11	Nested State Diagram Sub machine Diagram		A submachine state specifies the insertion of the specification of a submachine. The state machine that contains the submachine state is called the containing state machine.
12	Composite State		A state can be refined hierarchically by composite states.
13	Activity effect		An activity is actual behavior that can be invoked by any number of effects
14	Initial state point		It shows the starting state of object.
15	Final state point		It shows the terminating state of object.

Sequence Diagram

Introduction

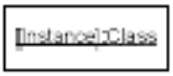
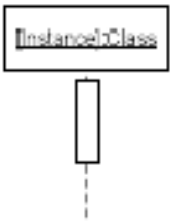
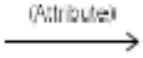
- Sequence diagrams model the dynamic aspects of a software system.
- The emphasis is on the "sequence" of messages rather than relationship between objects.
- A sequence diagram maps the flow of logic or flow of control within a usage scenario into a visual diagram enabling the software architect to both document and validate the logic during the analysis and design stages.
- Sequence diagrams provide more detail and show the message exchanged among a set of objects over time.
- Sequence diagrams are good for showing the behavior sequences seen by users of a diagram shows only the sequence of messages not their exact timing.
- Sequence diagrams can show concurrent signals.

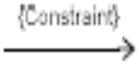
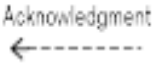
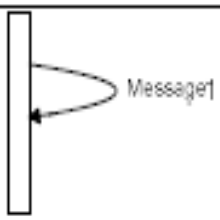
Purpose

- The main purpose of this diagram is to represent how different business objects interact.
- A sequence diagram shows object interactions arranged in time sequence.
- It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario.

When to use : Sequence Diagram

- Sequence diagram can be a helpful modeling tool when the dynamic behavior of objects needs to be observed in a particular use case or when there is a need for visualizing the "big picture of message flow".
- A company's technical staff could utilize sequence diagrams in order to document the behavior of a future system.
- It is during the design period that developers and architects utilize the diagram to showcase the system's object interactions, thereby putting out a more fleshed out overall system design.

Sr. No.	Name	Notation	Desription
1	Object		It represents the existence of an object of a particular time.
2	Life line		Lifeline represents the duration during which an object is alive and interacting with other objects in the system. It is represented by dashed lines.
3	Scope		It shows the time period during which an object or actor is performing an action.
4	Message transition		To send message from one object to another.
5	Message with attribute		To send message with some particular attribute

6	Message with constraint		To send message from one object to other vy some constraint.
7	Acknowledgement		It represent communication between objects conveys acknowledgement.
8	Self message		Self message occurs when an object sends a message to itself.
9	Recursive message		Self message occurs when an object sends a message to itself within recursive scope.

Activity Diagram

Introduction

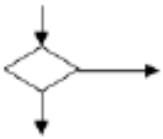
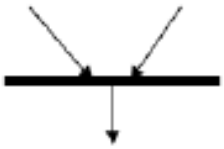
- An activity diagram is a type of flow chart with additional support for parallel behavior.
- This diagram explains overall flow of control.
- Activity diagram is another important diagram in UML to describe dynamic aspects of the system.
- Activity diagram is basically a flow chart to represent the flow from one activity to another activity
- The activity can be described as an operation of the system.
- The control flow is drawn from one operation to another. This flow can be sequential, branched or concurrent. This distinction is important for a distributed system.
- Activity diagrams deals with all type of flow control by using different elements like fork, join etc.

Purpose

- Contrary to use case diagrams, in activity diagrams it is obvious whether actors can perform business use cases together or independently from one another.
- Activity diagrams allow you to think functionally.

When to use : Activity Diagrams

- Activity diagrams are most useful when modeling the parallel behavior of a multithreaded system or when documenting the logic of a business process.
- Because it is possible to explicitly describe parallel events, the activity diagram is well suited for the illustration of business processes, since business processes rarely occur in a linear manner and often exhibit parallelisms.
- This diagram is useful to investigate business requirements at a later stage.
- An activity diagram is drawn from a very high level. So it gives high level view of a system. This high level view is mainly for business users or any other person who is not a technical person.
- This diagram is used to model the activities which are nothing but business requirements.
- So the diagram has more impact on business understanding rather *implementation details*.

No.	Name	Symbol	Description
1.	Activity		Represent individual activity of system.
2.	Transition		Represents flow of data from one activity to another.
3.	Decision		Decision node is a control node that accepts tokens on one or more incoming edges and selects outgoing edge from two or more outgoing flows. The notation for a decision node is a diamond-shaped symbol.
4.	Initial activity		Initial node is a control node at which flow starts when the activity is invoked. Activity may have more than one initial node. Initial nodes are shown as a small solid circle.
5.	Final activity		Final node is a control final node that stops all flows in an activity. Activity final nodes are shown as a solid circle with a hollow circle inside. It can be thought of as a goal notated as "bull's eye," or target.
6.	Fork		A fork in the activity diagram has a single incoming transition and multiple outgoing transitions exhibiting parallel behavior.The incoming transition triggers the parallel outgoing transitions.
7.	Join		A join in the activity diagram synchronizes the parallel behavior started at a fork. Join ascertains that all the parallel sets of activities (irrespective of the order) are completed before the next activity starts. It is a synchronization point in the diagram. Each fork in an activity diagram has a corresponding join where the parallel behavior terminates.