Unit-3

Behavioral Modelling

Behavioral Modelling:

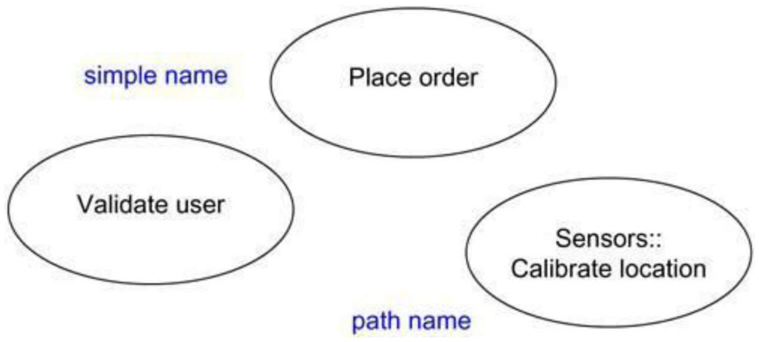
Behavioral model describes the interaction in the system. It represents the interaction among the structural diagrams. Behavioral modeling shows the dynamic nature of the system. They consist of the following −

- Interaction diagrams
- Use case diagrams
- Activity diagrams

Use Case:

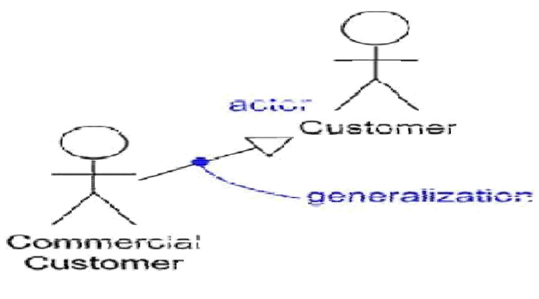
A *use case* is a description of a set of sequences of actions, including variants, that a system performs to yield an observable result of value to an actor. Graphically, a use case is rendered as an ellipse.

Names: Every use case must have a name that distinguishes it from other use cases. A *name* is a textual string. That name alone is known as a *simple name;* a p*ath name* is the use case name prefixed by the name of the package in which that use case lives.



Use Cases and Actors :

An actor represents a coherent set of roles that users of use cases play when interacting with these use cases. Typically, an actor represents a role that a human, a hardware device, or even another system plays with a system.



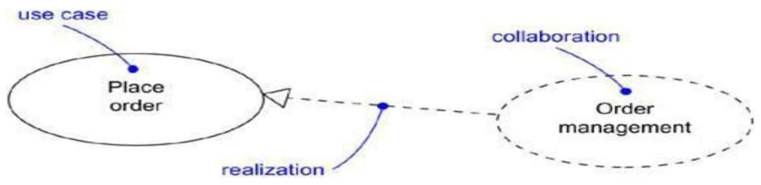
Use Cases and Flow of Event:

A use case describes *what* a system (or a subsystem, class, or interface) does but it does not specify *how* it does it. When you model, it's important that you keep clear the separation of concerns between this outside and inside view.

- You can specify the behavior of a use case by describing a flow of events in text clearly enough for an outsider to understand it easily. When you write this flow of events, you should include how and when the use case starts and ends.
    - Main flow of Events
    - Exceptional flow of Events.

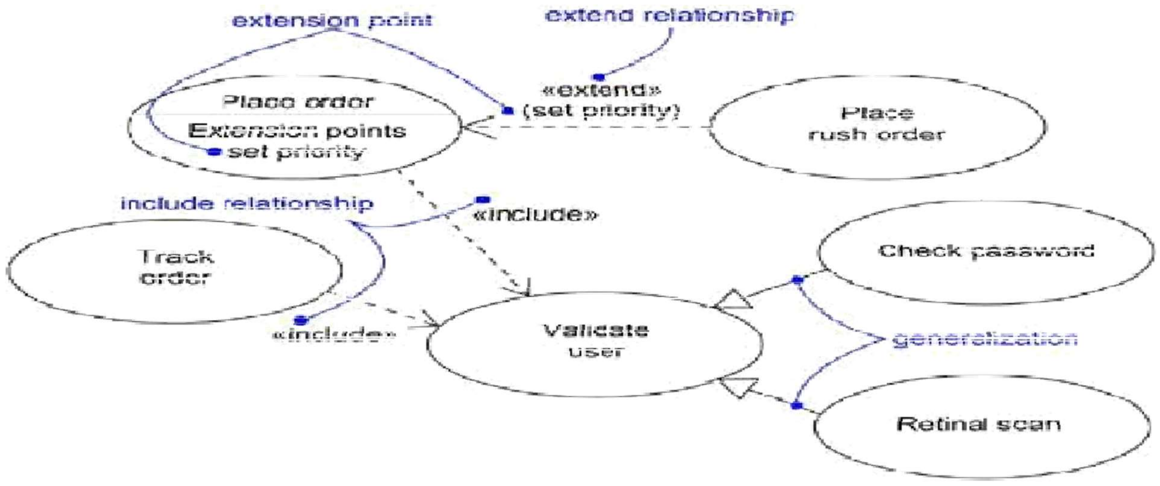
Use Cases and Collaborations:

Finding the minimal set of well-structured collaborations that satisfy the flow of events specified in all the use cases of a system is the focus of a system's architecture.



Organizing Use Cases:

We can Organize use cases by specifying generalization, include, and extend relationships among them. You apply these relationships in order to factor common behavior (by pulling such behavior from other use cases that it includes) and in order to factor variants.



Use Case Diagrams:

A *use case diagram* is a diagram that shows a set of use cases and actors and their relationships .A use case diagram is just a special kind of diagram and shares the same common properties as do all other diagramsa name and graphical contents that are a projection into a model.

Use case diagrams commonly contain

- Use cases
- Actors
- Dependency, Generalization, Association and Realization

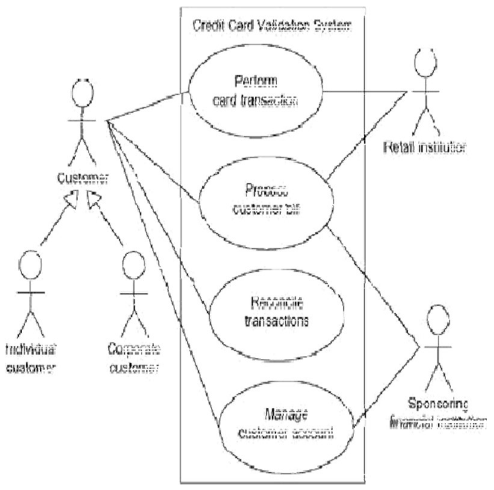
Like all other diagrams, use case diagrams may contain notes and constraints.

- Use case diagrams may also contain packages, which are used to group elements of your model into larger chunks.

Common Modeling Techniques:

To model the context of a system: In the UML, you can model the context of a system with a use case diagram, emphasizing the actors that surround the system. Deciding what to include as an actor is important because in doing so you specify a class of things that interact with the system.

- Identify the actors that surround the system by considering which groups require help from the system to perform their tasks; which groups are needed to execute the system's functions; which groups interact with external hardware or other software systems.
- Organize actors that are similar to one another in a generalization/specialization hierarchy.
- Where it aids understandability, provide a stereotype for each such actor.
- Populate a use case diagram with these actors and specify the paths of communication from each actor to the system's use cases.

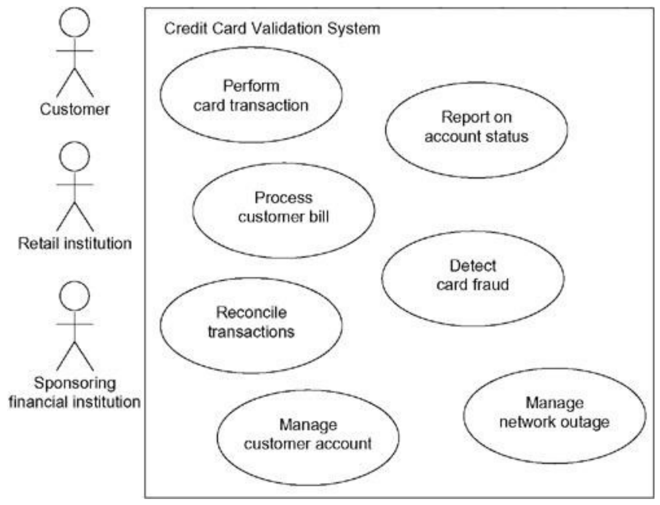


To model the requirements of a system: A requirement is a design feature, property, or behavior of a system. When you state a system's requirements, you are asserting a contract, established between those things that lie outside the system and the system itself.

To model the requirements of a system,

- Establish the context of the system by identifying the actors that surroundit.
- For each actor, consider the behavior that each expects or requires the system to provide.
- Name these common behaviors as use cases.

- Factor common behavior into new use cases that are used by others; factor variant behavior into new use cases that extend more main line flows.



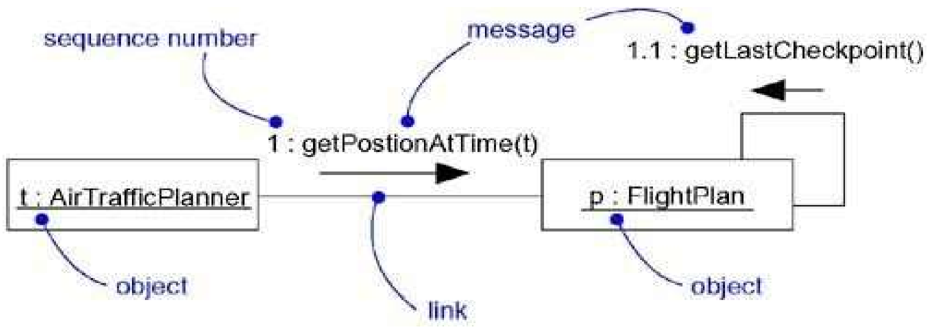
Interactions:

An interaction is a behavior that comprises a set of messages exchanged among a set of objects within a context to accomplish a purpose.

A message is a specification of a communication between objects that conveys information with the expectation that activity will ensue

- We use interactions to model the flow of control within an operation, a class, a component, a use case, or the system as a whole.
- Using interaction diagrams, we can reason about these flows in two ways.
  - First, we can focus on how messages are dispatched across time.
  - Second, we can focus on the structural relationships among the objects in an interaction and then consider how messages are passed within the context of that structure.

The UML provides a graphical representation of messages, as shown below:



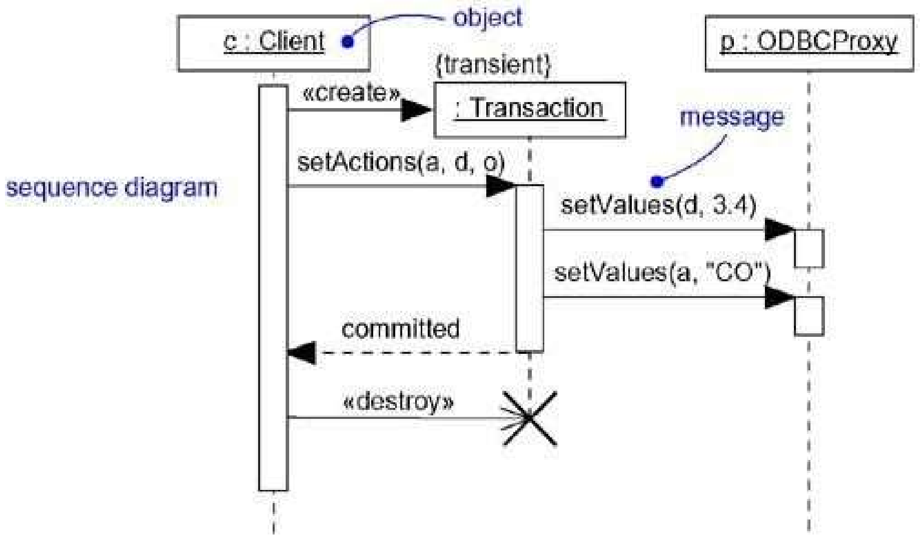
Interaction Diagrams:

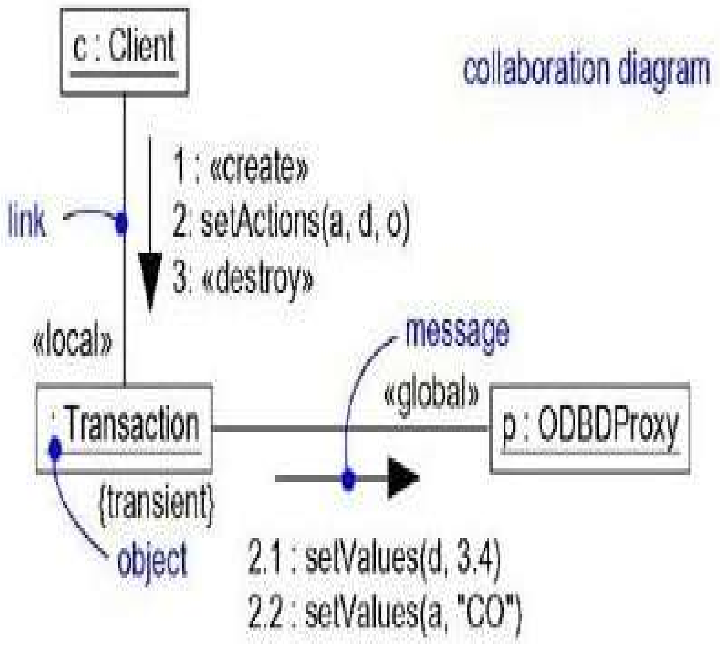
An interaction diagram shows an interaction, consisting of a set of objects and their relationships, including the messages that may be dispatchedamong them.

We use interaction diagrams to model the dynamic aspects of a system. Sequence diagrams and collaboration diagrams are collectively called Interaction Diagrams.

- A sequence diagram is an interaction diagram that emphasizes the time ordering of messages
- A collaboration diagram is an interaction diagram that emphasizes the structural organization of the objects that send and receive messages.

They are also used for constructing executable systems through forward and reverse engineering. Sequence diagrams and collaboration diagrams are semantically equivalent. We can convert one to the other without loss of information.





Common Properties:

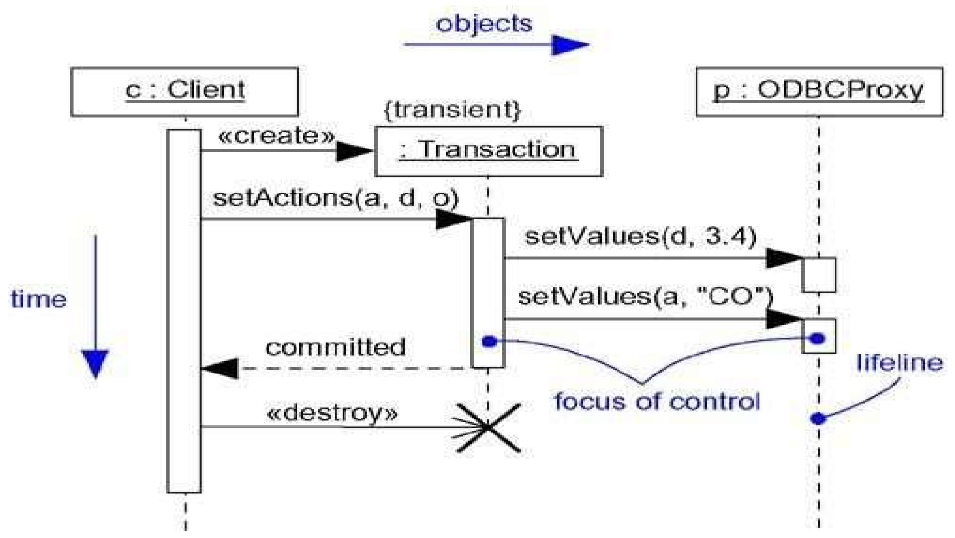
- It shares the same common properties as do all other diagrams - a name and graphical contents that are a projection into a model

Contents:

- Interaction diagrams commonly contain
  - Objects
  - Links
  - Messages

Sequence Diagram:

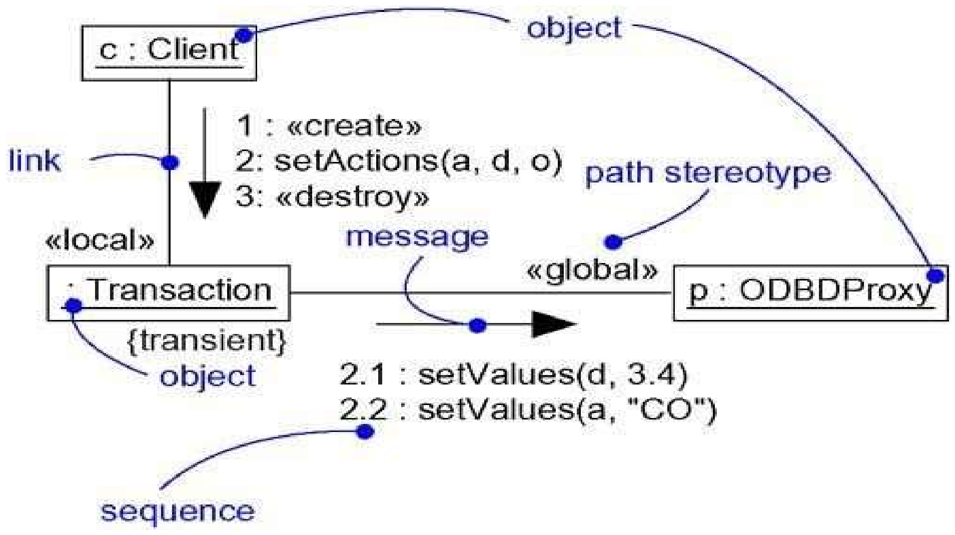
- A sequence diagram emphasizes the time ordering of messages
- Typically, we place the object that initiates the interaction at the left, and increasingly more subordinate objects to the right.
- Next, we place the messages that these objects send and receive along the Y axis, in order of increasing time from top to bottom.



Collaboration Diagram:

A collaboration diagram emphasizes the organization of the objects that participate in an interaction. Collaboration diagrams have two features that distinguish them from sequence diagrams.

- First, there is the path.
  - To indicate how one object is linked to another, we can attacha path stereotype to the far end of a link
- Second, there is the sequence number.
  - To indicate the time order of a message, we prefix the message with a number (starting with the message numbered 1), increasing monotonically for each new message in the flow of control (2 , 3 , and so on).


c : Client
object
1 : «create»
2: setActions(a, d, o)
3: «destroy»
link
path stereotype
«local»
message
«global»
: Transaction
p : ODBDProxy
{transient}
object
2.1 : setValues(d, 3.4)
2.2 : setValues(a, "CO")
sequence