UNIT-IV

Objectives:

To gain the knowledge of identifying different requirements and showing them in interactions, use cases and activities in the application.

Syllabus: Behavioral Modeling

Interactions, Interaction Diagrams, UseCases, UseCase Diagrams, Activity Diagrams

Outcomes:

Students will be able to

Identify interactions and usecases in the application.

Illustrate and design Interaction diagrams, Usecase diagrams for different cases.

Illustrate Activity diagrams for different cases.

3.1 INTERACTIONS:

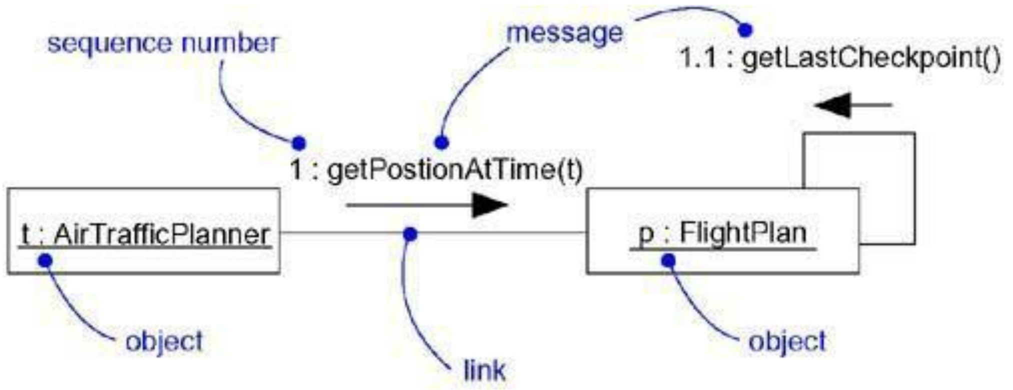
An interaction is a behavior that comprises a set of messages exchanged among a set of objects within a context to accomplish a purpose.
A message is a specification of a communication between objects that conveys information with the expectation that activity will ensue

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.



` Figure: Messages, Links, and

Sequencing We find an interactions

wherever objects are linked to one another.

in the collaboration of objects that exist in the context of wer system or subsystem.

in the context of an operation.

in the context of a class. (We use interactions to visualize, specify, construct, and document the semantics of a class)

An interaction may also be found in the representation of a component, node, or use case, each of which, in the UML, is really a kind of classifier.

<u>Objects and Roles</u>

The **objects** that participate in an interaction are either concrete things or prototypical things.

As a concrete thing, an object represents something in the real world. For example, p, an instance of the class Person, might denote a particular human. Alternately, as a prototypical thing, p might represent any instance of Person.

Links

A link is a semantic connection among objects.

A link is an instance of an association.

The following figure shows

wherever a class has an association to another class, there may be a link between the instances of the two classes;
wherever there is a link between two objects, one object can send a message to the other object.

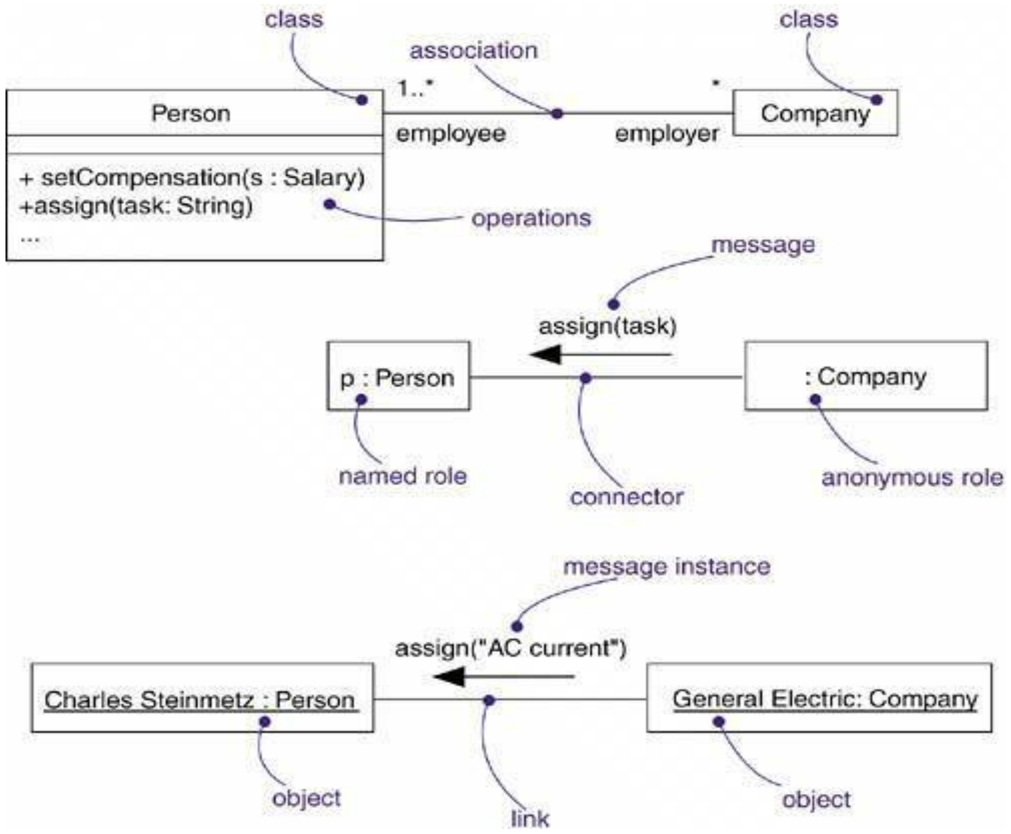


Figure: Links, Associations and Connectors

We can adorn the appropriate end of the link with any of the following standard stereotypes:

association	Specifies that the corresponding object is visible by association
self	Specifies that the corresponding object is visible because it is the dispatcher of the operation
global	Specifies that the corresponding object is visible because it is in an enclosing scope
local	Specifies that the corresponding object is visible because it is in a local scope
parameter	Specifies that the corresponding object is visible because it is a Parameter

MESSAGE

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

The receipt of a message instance may be considered an instance of an event.

In the UML, we can model several kinds of actions.

Call	Invokes an operation on an object; an object may send a message to itself, resulting in the local invocation of an operation
Return	Returns a value to the caller
Send	Sends a signal to an object
Create	Creates an object
Destroy	Destroys an object; an object may commit suicide by destroying itself

The UML provides a visual distinction among these kinds of messages, as shown below

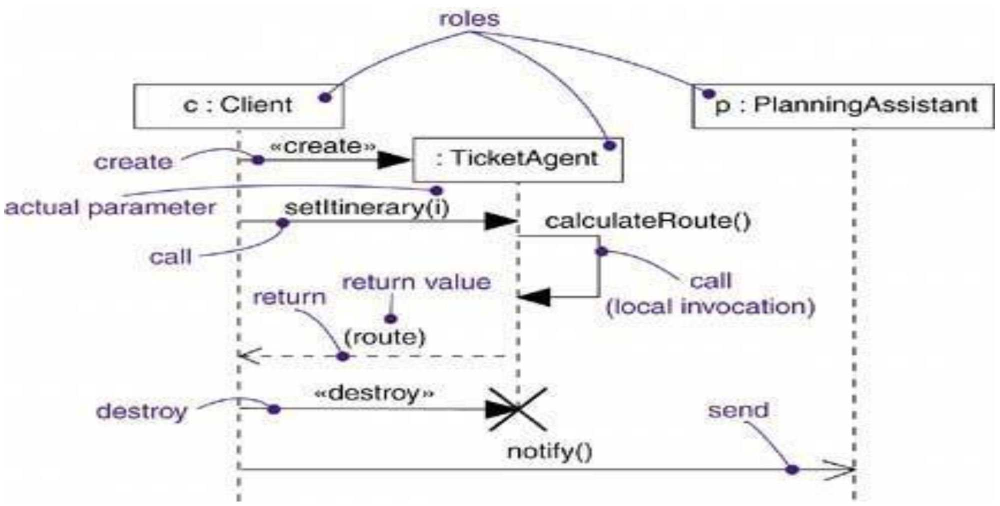


Fig: Messages

Messages can also correspond to the sending of signals

A signal is an object value communicated to a target object asynchronously

SEQUENCING

When an object passes a message to another object, the receiving object might in turn send a message to another object, which might send a message to yet a different object, and so on. This stream of messages forms a **sequence**.

Each process and thread within a system defines a distinct flow of control, and within each flow, messages are ordered in sequence by time.

A communication diagram shows message flow between roles within a collaboration

Messages flow along connections of the collaborations as shown below

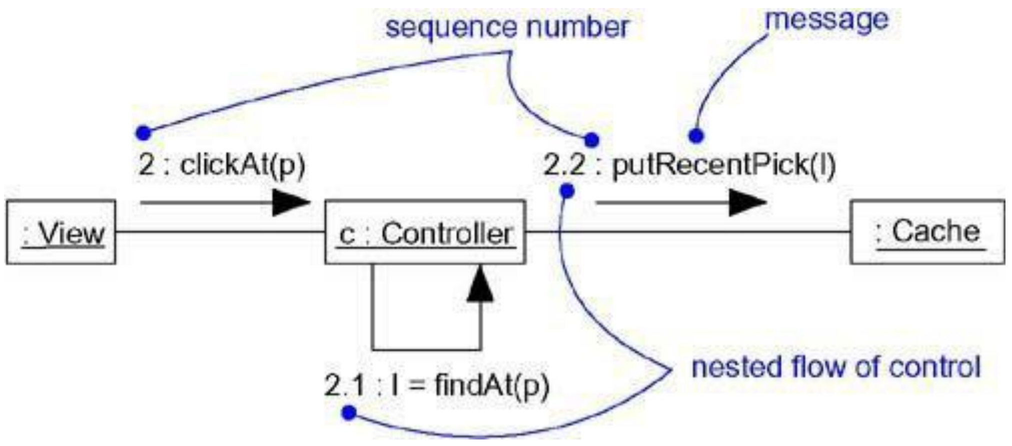


Fig: Procedural Sequence

The following figure specify a flat flow of control, rendered using a stick arrowhead, to model the nonprocedural progression of control from step to step.

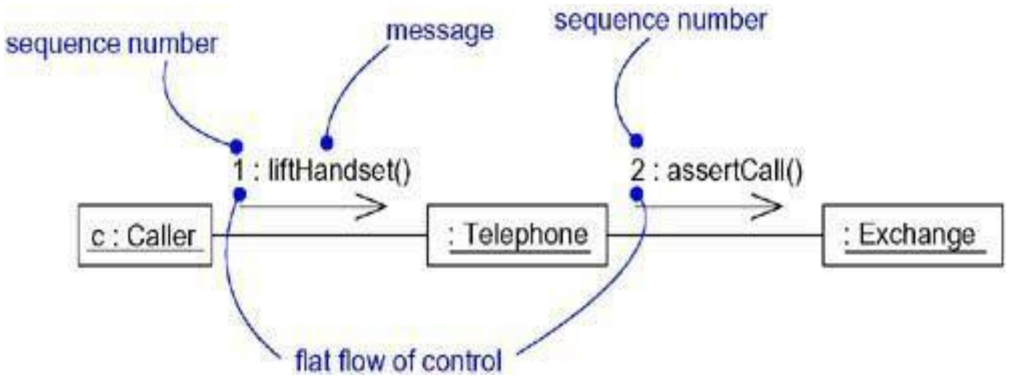


Fig: Flat Sequence

Creation, Modification, and Destruction

The relationships among objects may come and go.

To specify if an object or link enters and/or leaves during an interaction, we can attach one of the following constraints to the element

new	Specifies that the instance or link is created during execution of the enclosing interaction
destroyed	Specifies that the instance or link is destroyed prior to completion of execution of the enclosing interaction
transient of	Specifies that the instance or link is created during execution the enclosing interaction but is destroyed before completion of execution

Note: The modification of an object in a sequence diagram is represented by showing the state or values on the lifeline

Representation

When we model an interaction, we typically include both objects (each one playing a specific role) and messages (each one representing the communication between objects, with some resulting action).

We can visualize those objects and messages involved in an interaction in two ways:

- by emphasizing the time ordering of its messages (sequence diagram)
- by emphasizing the structural organization of the objects that send and receive messages (collaboration diagram)

Note: Sequence diagrams and collaboration diagrams are largely isomorphic.

Common Modeling Techniques:

Modeling a Flow of Control

The most common purpose for which we use interactions is to model the flow of control that characterizes the behavior of a system as a whole, including use cases, patterns, mechanisms, and frameworks, or the behavior of a class or an individual operation.

To model a flow of control:

Set the context for the interaction, whether it is the system as a whole, a class, or an individual operation.

Set the stage for the interaction by identifying which objects play a role; Set their initial properties, including their attribute values, state, and role. If our model emphasizes the structural organization of these objects, identify the links that connect them, relevant to the paths of communication that take place in this interaction.

In time order, specify the messages that pass from object to object.

To convey the necessary detail of this interaction, adorn each object at very moment in time with its state and role.

The following figure shows a set of objects that interact in the context of publish and subscribe mechanism.

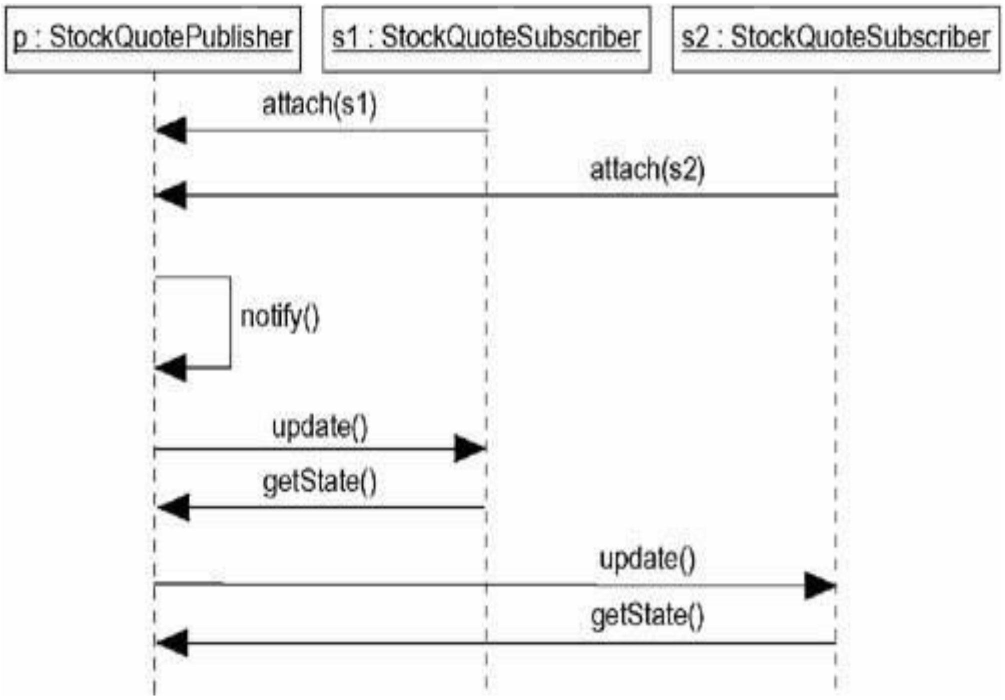


Fig: Flow of control by time

The following figure is semantically equivalent to the previous one, but it is drawn as a collaboration diagram, which emphasizes the structural organization of the objects.

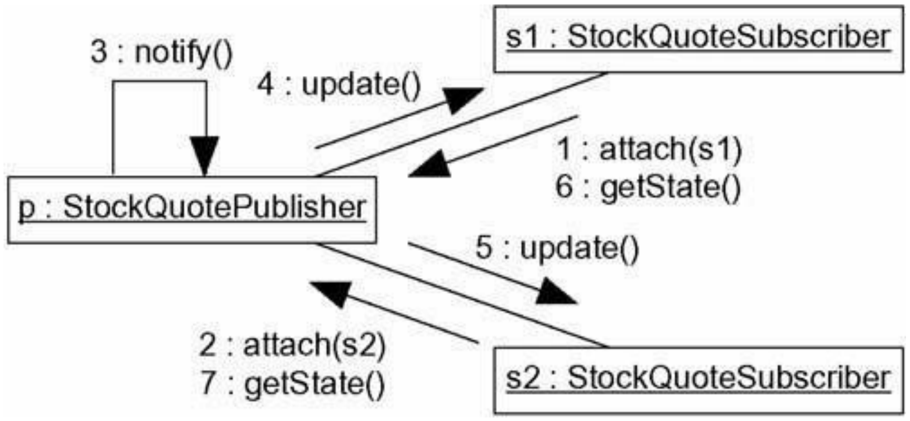


Fig: Flow of Control by Organization

3.2 INTERACTION DIAGRAMS

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

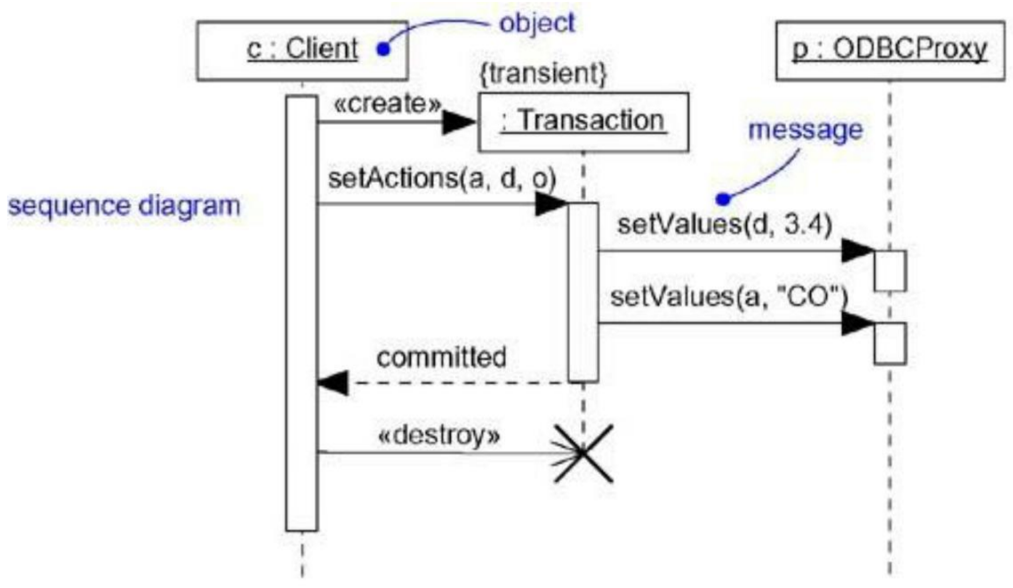
We use interaction diagrams to model the dynamic aspects of a system.

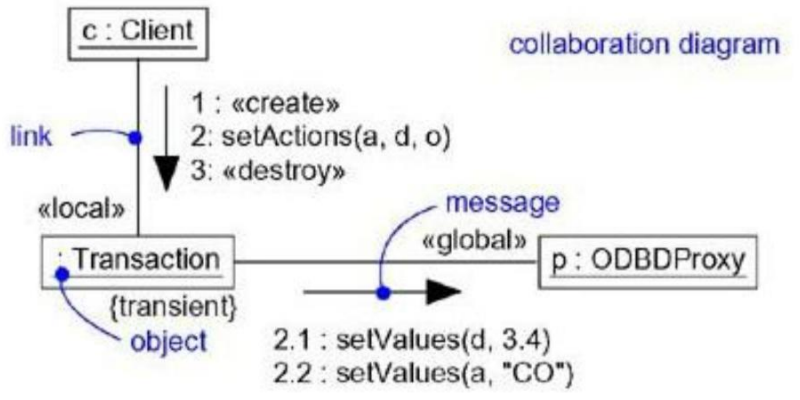
Sequence diagrams and collaboration diagrams are collectively called Interaction Diagrams.

- 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 DIAGRAMS

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.

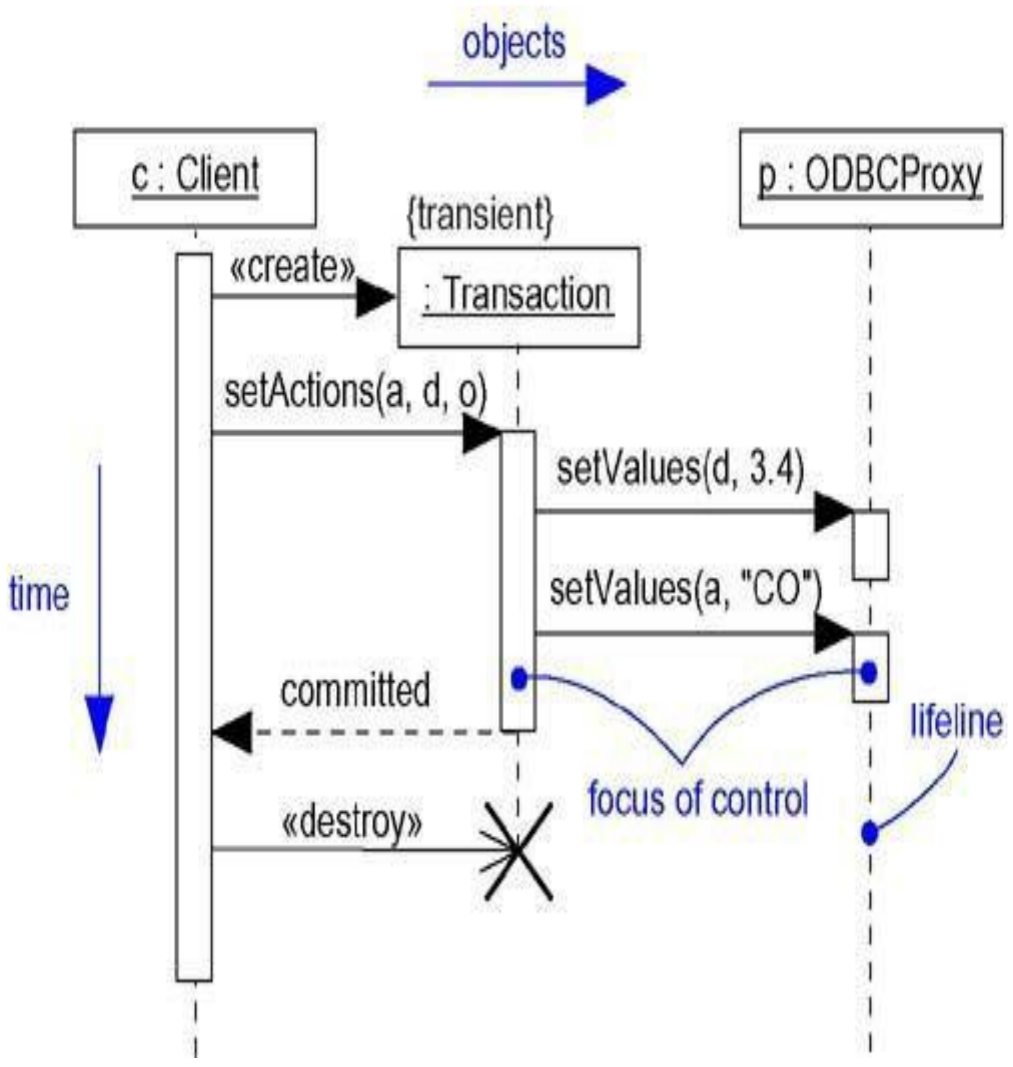


Fig: Sequence Diagram

COLLABORATION DIAGRAMS

A collaboration diagram emphasizes the organization of the objects that participate in an interaction.

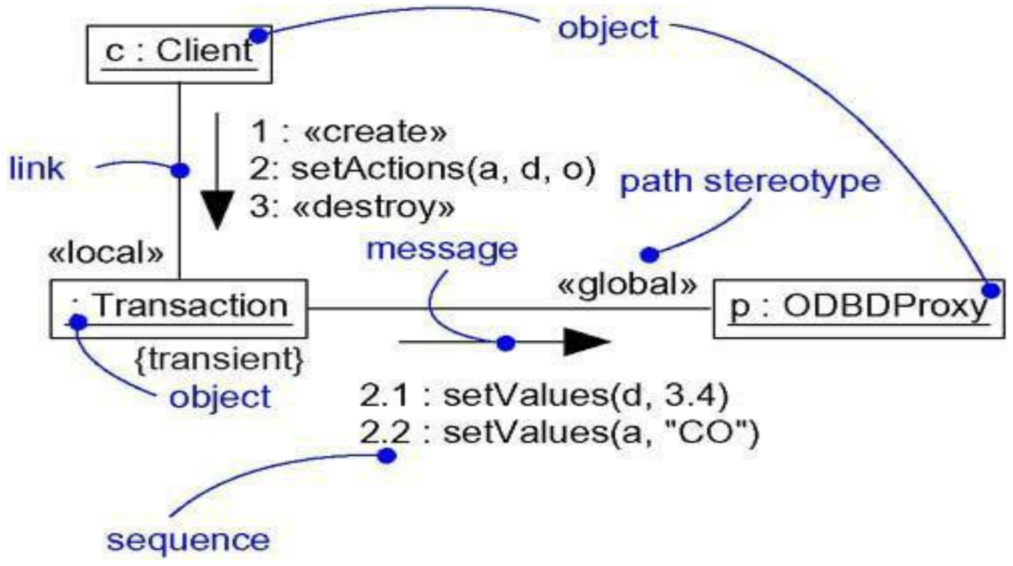


Fig: Collaboration Diagram

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 attach a 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 , andso on).

SEMANTIC EQUIVALENCE

sequence diagrams and collaboration diagrams are semantically equivalent. We can take a diagram in one form and convert it to the other without any loss of information

COMMON USES

When we model the dynamic aspects of a system, we typically use interaction diagrams in two ways:

To model flows of control by time ordering

- Sequence diagrams do a better job of visualizing simple iteration and branching than do collaboration diagrams

To model flows of control by organization

- Collaboration diagrams do a better job of visualizing complex iteration and branching and of visualizing multiple concurrent flows of control

COMMON MODELING TECHNIQUES

1. Modeling Flows of Control by Time Ordering
2. Modeling Flows of Control by Organization
3. Forward and Reverse Engineering

Modeling Flows of Control by Time Ordering

To model a flow of control by time ordering

Set the context for the interaction, whether it is a system, subsystem, operation, or class, or one scenario of a use case or collaboration.

Set the stage for the interaction by identifying which objects play a role in the interaction.

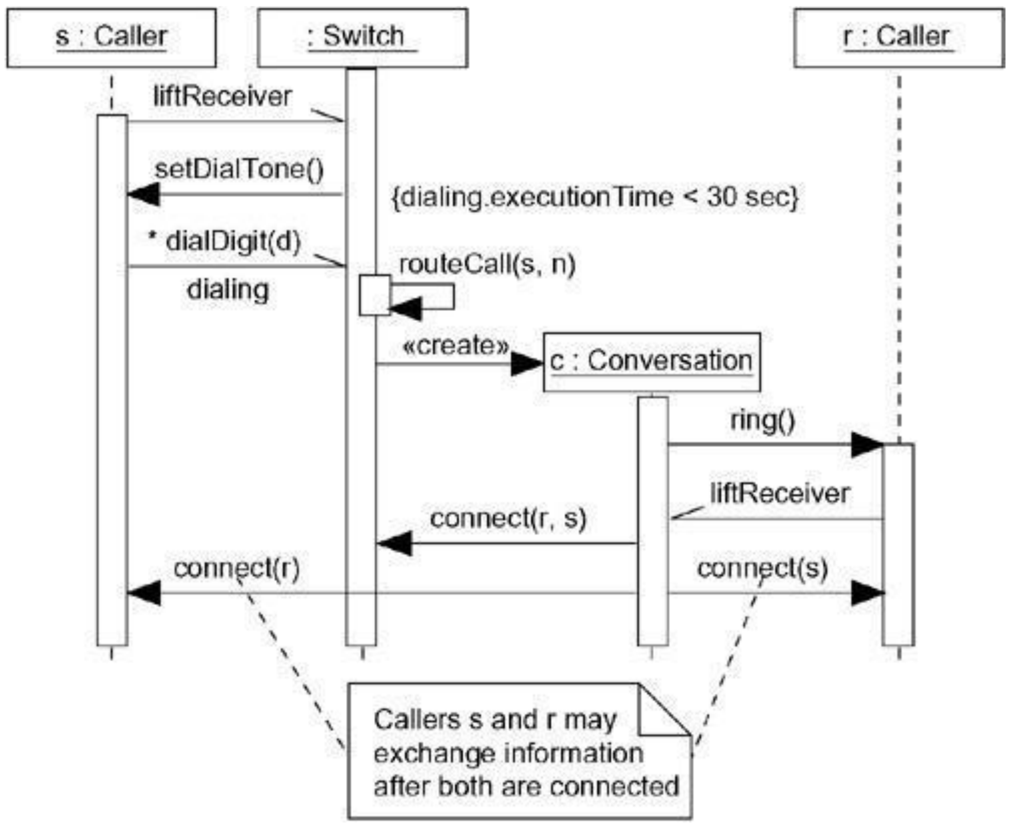
Set the lifeline for each object

Starting with the message that initiates this interaction, lay out each subsequent message from top to bottom between the lifelines, showing each message's properties

If we need to visualize the nesting of messages or the points in time when actual computation is taking place, adorn each object's lifeline with its focus of control

If we need to specify time or space constraints, adorn each message with a timing mark and attach suitable time or space constraints.

If we need to specify this flow of control more formally, attach pre- and postconditions to each message.



Modeling Flows of Control by Organization

To model a flow of control by organization:

Set the context for the interaction, whether it is a system, subsystem, operation, or class, or one scenario of a use case or collaboration.

Set the stage for the interaction by identifying which objects play a role in the interaction.

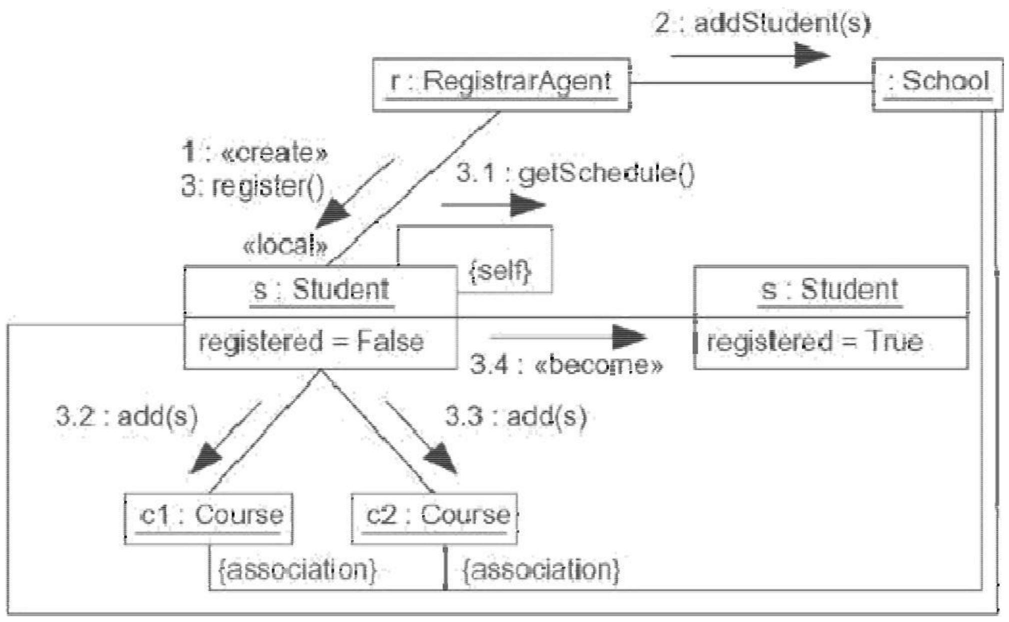
Set the initial properties of each of these objects

Specify the links among these objects, along which messages may pass

Starting with the message that initiates this interaction, attach each subsequent message to the appropriate link,setting its sequence number, as appropriate.

If we need to specify time or space constraints, adorn each message with a timing mark and attach suitable time or space constraints.

If we need to specify this flow of control more formally, attach pre- and postconditions to each message.



FORWARD AND REVERSE ENGINEERING

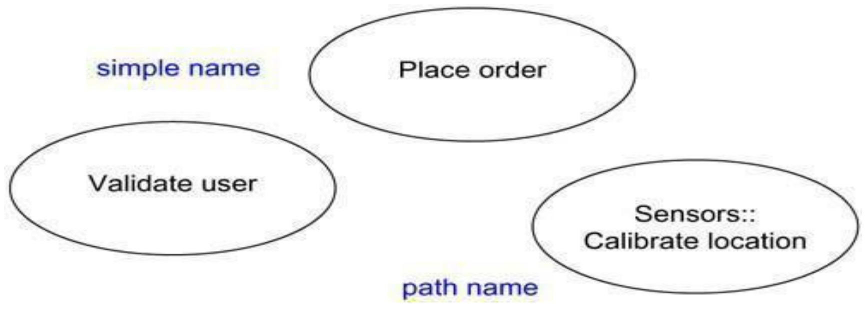
Forward engineering (the creation of code from a model) is possible for both sequence and collaboration diagrams, especially if the context of the diagram is an operation

Reverse engineering (the creation of a model from code) is also possible for both sequence and collaboration diagrams, especially if the context of the code is the body of an operation.

3.3 USE CASES

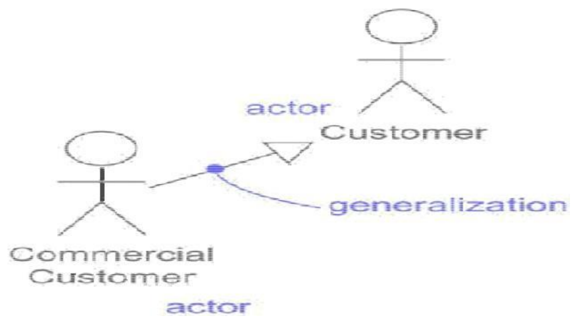
A *use case* is a description of a set of sequences of actions, including variants, that a system performs to yield an observable result of value to an actor. Graphically, a use case is rendered as an ellipse.

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 *path 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.



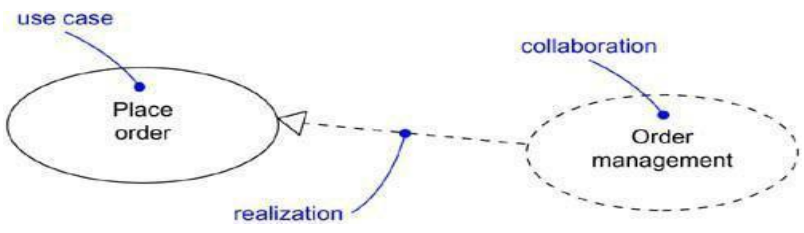
Use Cases and Flow of Event : A use case describes *what* a system (or a subsystem, class, or interface) does but it does not specify *how* it does it. When you model, it's important that you keep clear the separation of concerns between this outside and inside view.

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.

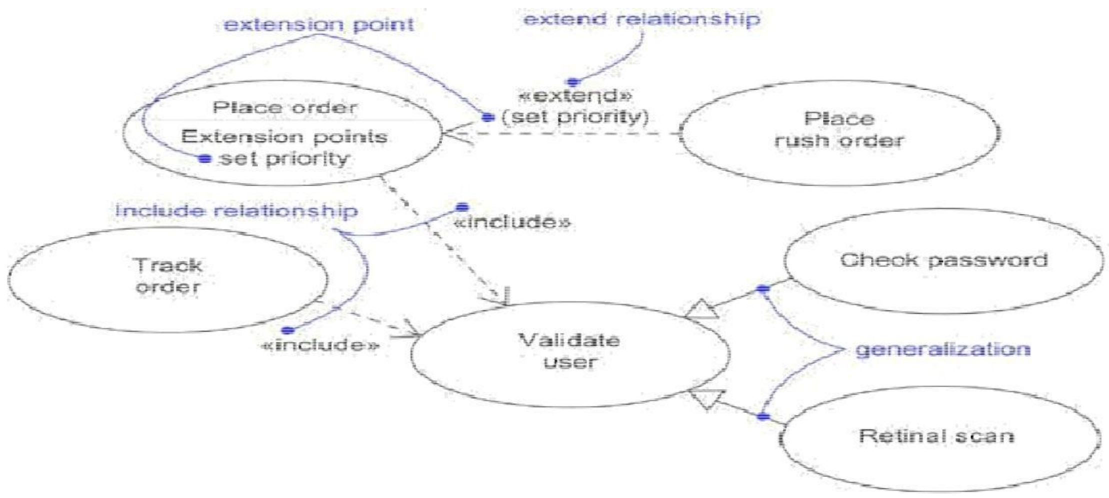
Main flow of Events

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.



Common Modeling Techniques:

Modeling the Behavior of an Elements.

3.4 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:

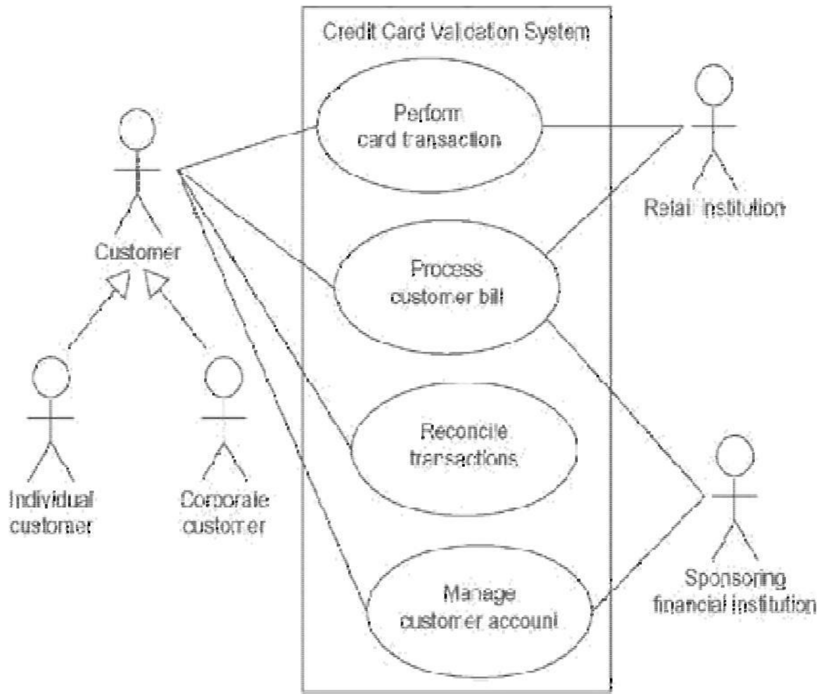
To model the context of a system: In the UML, you can model the context of a system with a use case diagram, emphasizing the actors that surround the system. Deciding what to include as an actor is important because in doing so you specify a class of things that interact with the system.

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.

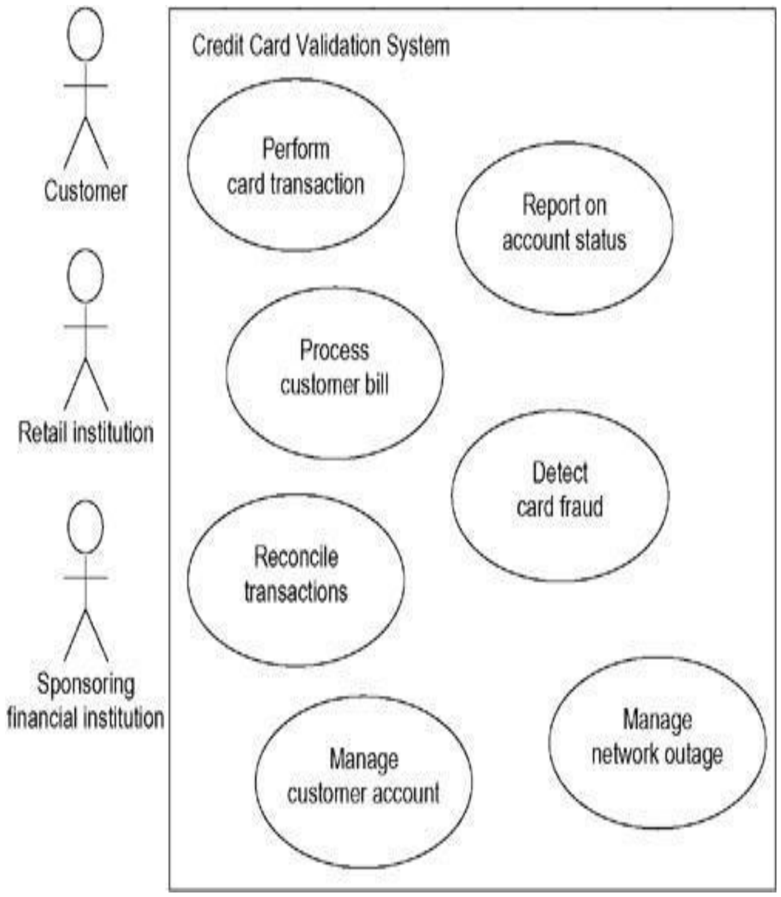
To model the requirements of a system,

Establish the context of the system by identifying the actors that surround it.

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.



Forward and Reverse Engineering

3.5. Activity Diagrams :

Activity diagrams are one of the five diagrams in the UML for modeling the dynamic aspects of systems. An activity diagram is essentially a flowchart, showing flow of control from activity to activity.

Activity diagrams are not only important for modeling the dynamic aspects of a system, but also for constructing executable systems through forward and reverse engineering.

Activities ultimately result in some *action,* which is made up of executable atomic computations that result in a change in state of the system or the return of a value.

Activity diagrams commonly contain

Activity states and action states

Transitions

Objects

Like all other diagrams, activity diagrams may contain notes and constraints.

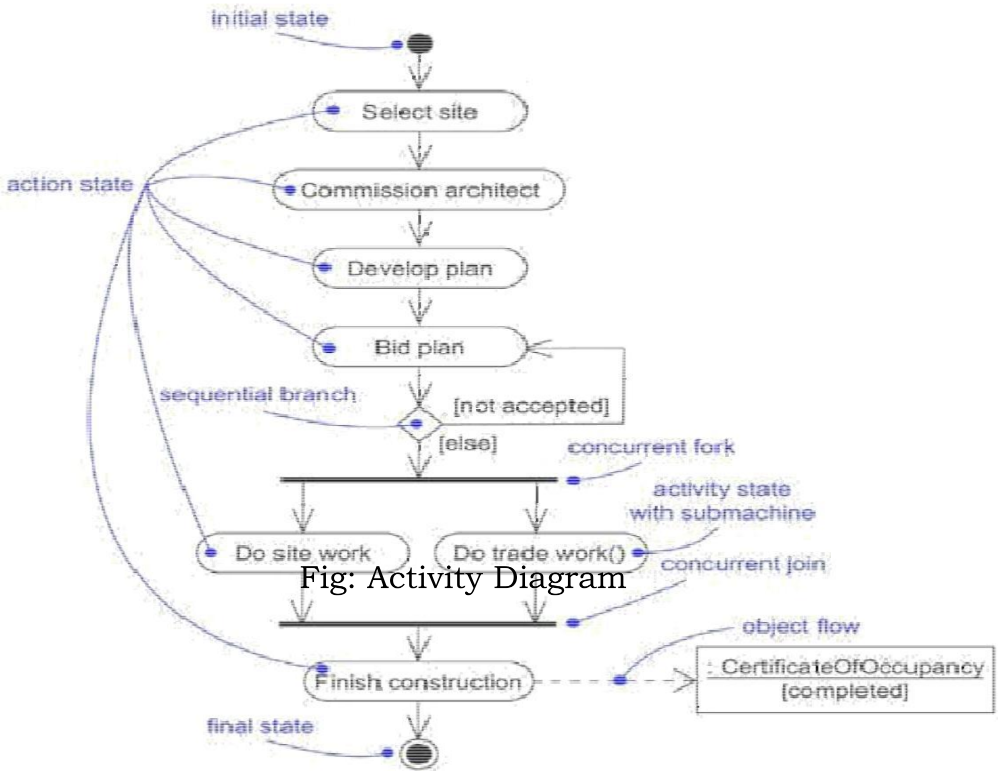


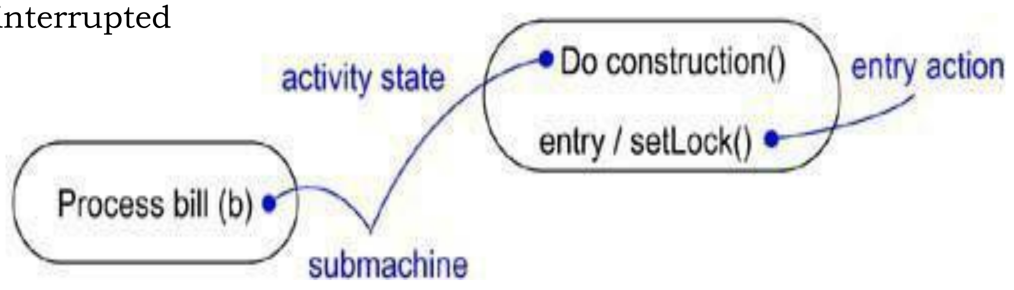
Fig: Activity Diagram

3.5.1. Action States and Activity States:

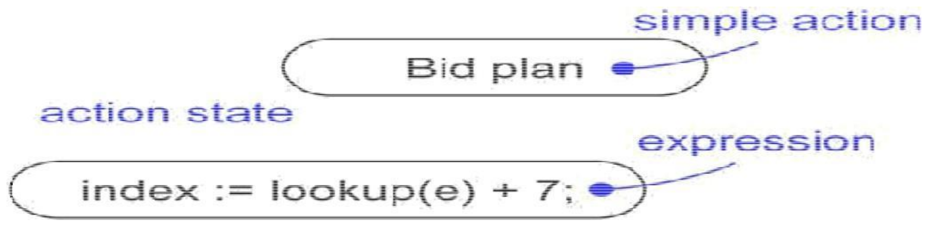
In the flow of control modeled by an activity diagram, might evaluate some expression that sets the value of an attribute or that returns some value. Alternately, you might call an operation on an object, send a signal to an object, or even create or destroy an object.

These executable, atomic computations are called action states because they are states of the system, each representing the execution of an action.

Action states can't be decomposed. Furthermore, action states are atomic, meaning that events may occur, but the work of the action state is not interrupted



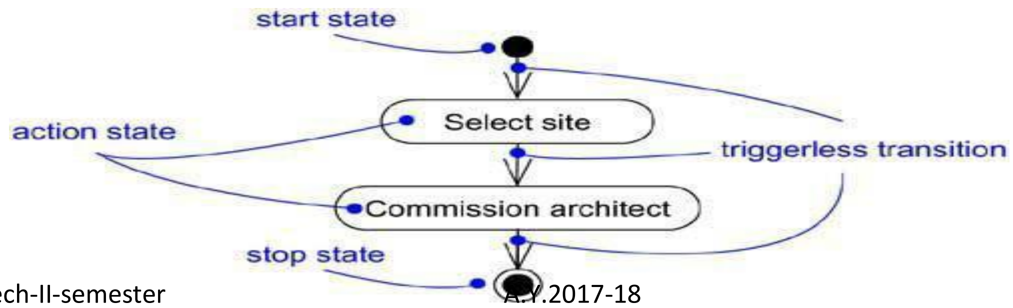
In contrast, activity states can be further decomposed, their activity being represented by other activity diagrams. Furthermore, activity states are not atomic, meaning that they may be interrupted and, in general, are considered to take some duration to complete.



3.5.2 Transition:

Trigger less transitions may have guard conditions, meaning that such a transition will fire only if that condition is met.

When the action or activity of a state completes, flow of control passes immediately to the next action or activity state. You specify this flow by using transitions to show the path from one action or activity state to the next action or activity state

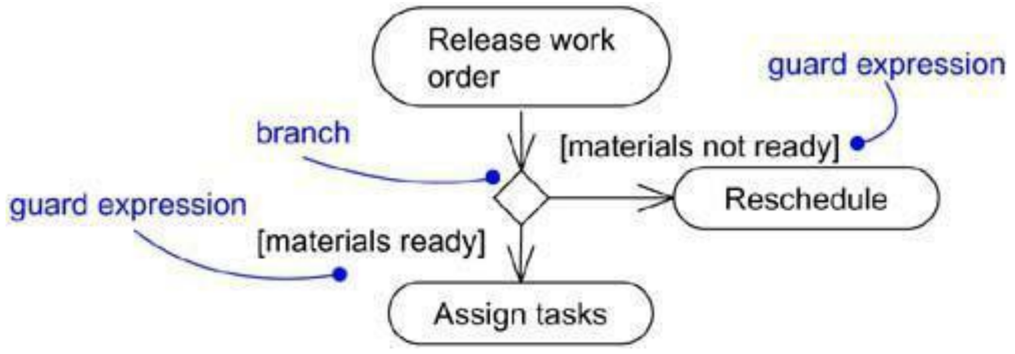


3.5.3. Branching:

Sequential transitions are common; you can include a branch, which specifies alternate paths taken based on some Boolean expression.

Represent a branch as a diamond. A branch may have one incoming transition and two or more outgoing ones.

On each outgoing transition, you place a Boolean expression, which is evaluated only once on entering the branch.



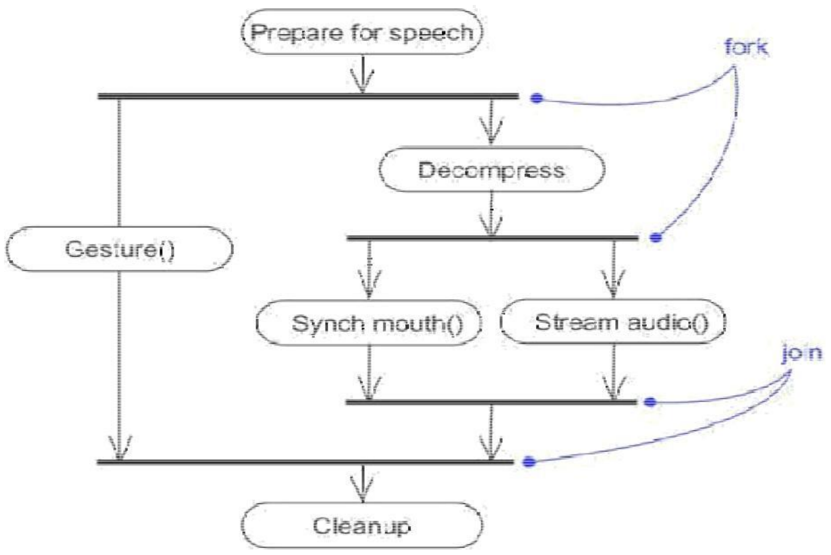
3.5.4. Forking and Joining:

Simple and branching sequential transitions are the most common paths you'll find in activity diagrams. Howeverespecially when you are modeling workflows of business processesyou might encounter flows that are concurrent.

In the UML, you use a synchronization bar to specify the forking and joining of these parallel flows of control. A synchronization bar is rendered as a thick horizontal or vertical line.

A *fork* may have one incoming transition and two or more outgoing transitions, each of which represents an independent flow of control. Below the fork, the activities associated with each of these paths continues in parallel.

A *join* may have two or more incoming transitions and one outgoing transition. Above the join, the activities associated with each of these paths continues in parallel.

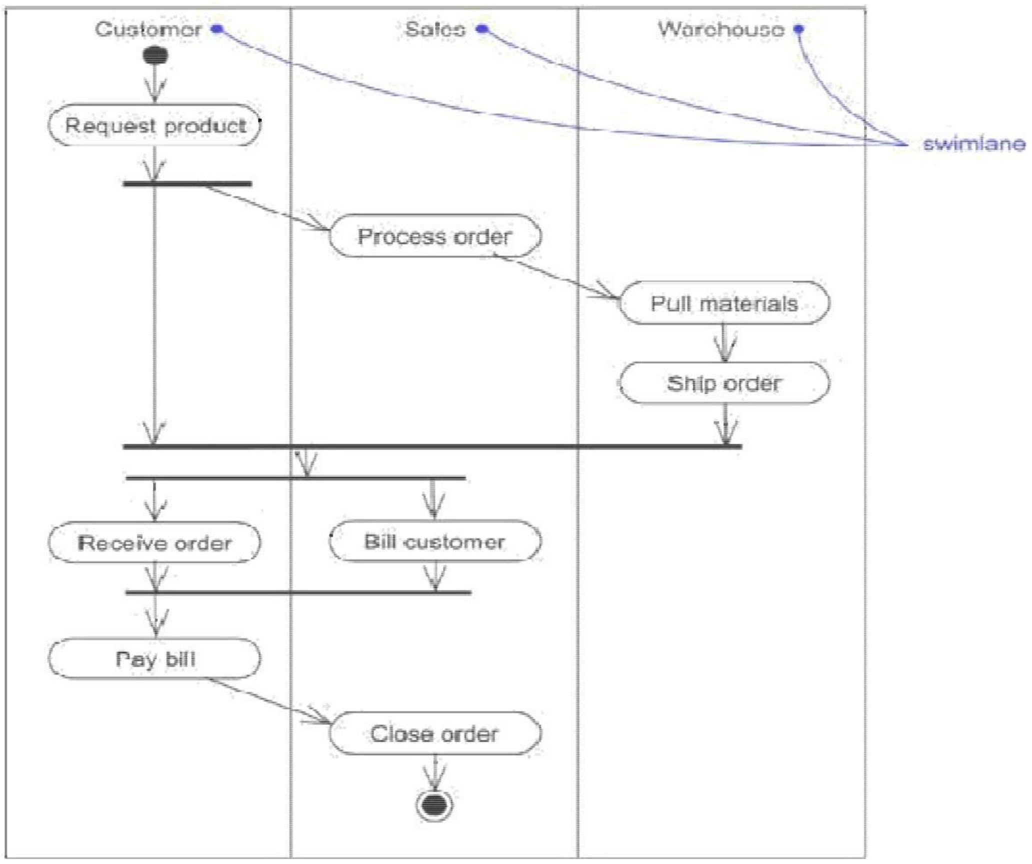


3.5.5.Swimlanes:

Each swimlane has a name unique within its diagram. A swimlane really has no deep semantics, except that it may represent some real-world entity.

Each swimlane represents a high-level responsibility for part of the overall activity of an activity diagram, and each swimlane may eventually be implemented by one or more classes.

In an activity diagram partitioned into swimlanes, every activity belongs to exactly one swimlane, but transitions may cross lanes.

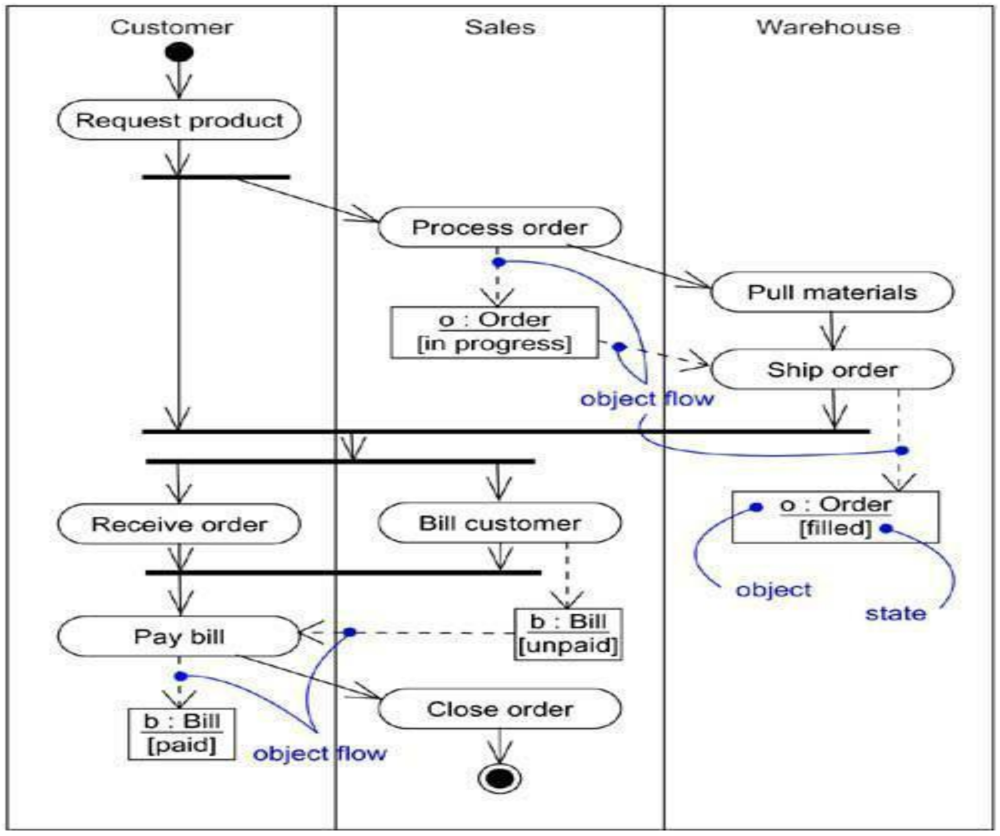


3.5.6. Object Flow:

Objects may be involved in the flow of control associated with an activity diagram.

For example, in the workflow of processing an order as in the previous figure, the vocabulary of your problem space will also include such classes as Order and Bill.Instances of these two classes will be produced by certain activities.

In addition to showing the flow of an object through an activity diagram, you can also show how its role, state and attribute values change.



Common Modeling Techniques of Activity Diagrams:

To Model a Workflow

To model an Operation

Forward and Reverse Engineering

Assignment-Cum-Tutorial Questions

A. Questions testing the remembering / understanding level of students

*I) **Objective Questions***

1. Which of the two diagrams are isomorphic….. [ ] a) Class diagram and Object diagram b) Sequence and Collaboration diagram c) Statechart and Activity diagram d) Component and Deployment diagrams
2. An interaction is a behavior that exchange messages among…………..
3. …………… diagram emphasizes time ordering of messages.
4. ……………….diagram emphasizes the organization of the objects that participates in an interaction.
5. Modeling the flow of control by organization….. [ ]

   a) Sequence Diagrams b) Collaboration Diagrams

   c) Use Case Diagrams d) Activity Diagrams
6. Branching can be represented in an Activity diagram with… [ ]

   a) Ellipse b) Diamond c) Circle d) Rounded Rectangle
7. In activity diagram Action states can't be decomposed [True/False] [ ]
8. ……………… is a one incoming transitions with one or more outgoing transitions.
9. ……….the synchronization of two or more concurrent flow of control.
10. In use case diagram can have packages and notes [True/False] [ ]

II) Descriptive Questions

1. Explain about Use cases and Use case diagrams with an example?
2. Briefly explain about the Interaction diagrams?
3. How sequence and collaboration diagrams are differ with each other. Draw interaction diagram for Library Management System.
4. What is sequencing. What are the uses of different stereotypes used in interaction?
5. Exemplify about Activity diagram?
6. How Forking and Joining will used in Activity diagram. Explain with an example.

]

7. Describe the common modeling techniques of Usecase diagrams.
8. With an example explain about the use of swimlanes and object flow in Activity diagram?
9. Explain how usecase diagrams are useful to specify system requirements.

II) Problems:

1. Draw the collaboration diagram for the use case "Register" that allows the registration of a new student into a University through Online Registration System?
2. Design interaction diagrams for Railway Reservation System?
3. Show different action and activity states, swimlanes in the activity diagram?
4. Draw the object flow in point of sales system?
5. How swimelanes are used in activity diagram explain with an example?
6. Draw different Usecase diagrams for ATM application
7. Sketch use case diagram for Online Shopping?
8. Identify different use cases in Student admission system and draw Usecase diagrams?