UNIT – II

UML Diagrams: Use case diagram – UML class diagram – interaction diagram – state diagram – activity diagram – Requirements for ATM banking system – case study

UML DIAGRAMS: The UML is a language for specifying, constructing, visualizing and documenting the software system and its components. The UML is a graphical language with sets of rules and semantics, in a form known as Object constraint language (OCL). The primary goals in the design of the UML were as follows:

1. Provide users a ready-to-use, expensive visual modeling language so they can develop and exchange meaningful models.
2. Provide extensibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development processes.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of the OO tools market.
6. Support higher-level development concepts.
7. Integrate best practices and methodologies.

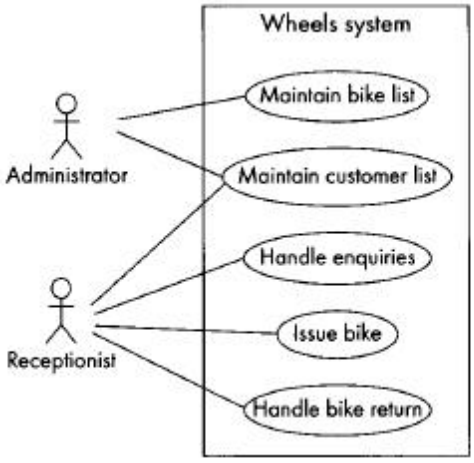
The UML defines 6 graphical diagrams:

1. Use-case diagram
2. Class Diagram
3. Interaction diagram
    a. Sequence diagram
    b. Collaboration diagram
4. State chart diagram
5. Activity diagram
6. Implementation diagram
    a. Component diagram
    b. Deployment diagram

USECASE DIAGRAM:

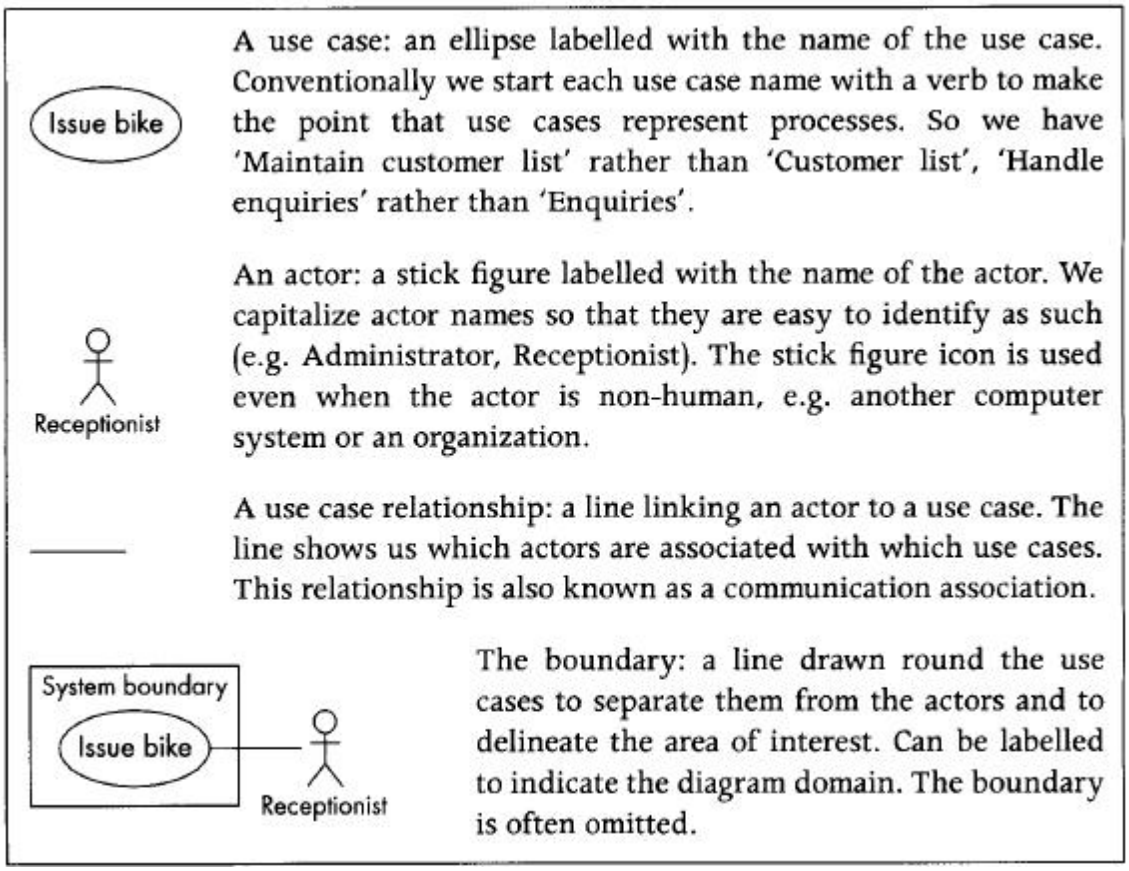
Use case Diagram concept was introduced by Ivar Jacobson in the OOSE method. This corresponds to a sequence of transactions, in which each transaction is invoked from outside the system (actors and engages internal objects to act with one another.

This is a graph of actors, a set of use cases enclosed by a system boundary, communication association between the actors and the use cases and generalization among the use cases.



IDENTIFYING USE CASES FROM THE ACTORS One is to identify the actors, the users of the system, and for each one, to establish how they use the system, what they use it to achieve. Scenarios belonging to the same use case have a common goal each scenario in the group

describes a different sequence of events involved in achieving (or failing to achieve) the use case goal.



The UML Symbols for use case diagram

These relationships are shown in a use case diagram:

1. ***Communication:*** Connecting actor symbol to the use case symbol with a solid path.
2. ***Uses:*** relationship between use cases is shown by a generalization arrow from the use case.
3. ***Extends:*** Used when one use case that is similar to another use case but does a bit more. It is like a subclass.

UML CLASS DIAGRAM:

The UML Class diagram, also referred to as object modeling, is the main static analysis diagram. This shows the static structure of the model.

It is a collection of static modeling elements, such as classes and their relationships, connected as graph to each other and to their contents. It does not show temporal information, which is required in dynamic modeling.

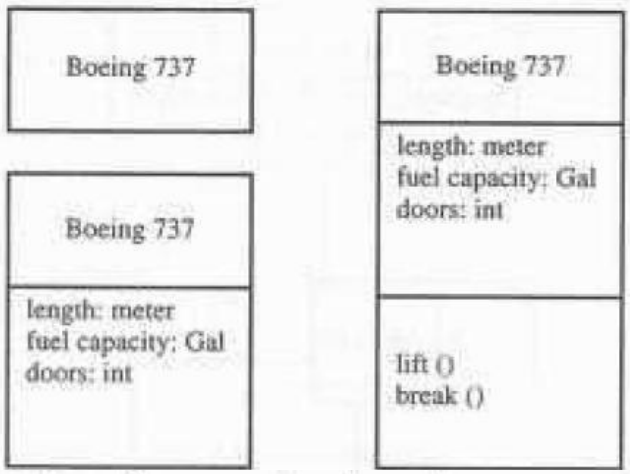
The main task of object modeling is to graphically show what each object will do in the problem domain, describe the structure (such as class hierarchy) and relationship among the objects (such as associations) by visual notations, and determine what behavior fall within and outside the problem domain.

CLASS NOTATION: STATIC STRUCTURE

A class is drawn by a rectangle with three components separated by horizontal lines.

- The top name compartment holds the class name
- The general properties of the class, such as attributes are in the middle component.
- The bottom components contain the list of operations.

A separator line is not drawn for a missing compartment if a component is suppressed; no inference can be drawn about the presence or absence of elements in it.



In class notation, either or both the attributes and operation compartments may be suppressed.

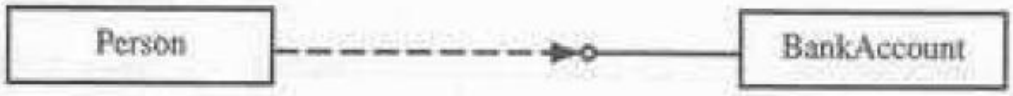
OBJECT DIAGRAM

A static object diagram is an instance of a class diagram. It shows the detailed state of the system at a point in time. Class diagram contains object, so a class diagram with object and no classes is an object diagram

CLASS INTERFACE NOTATION

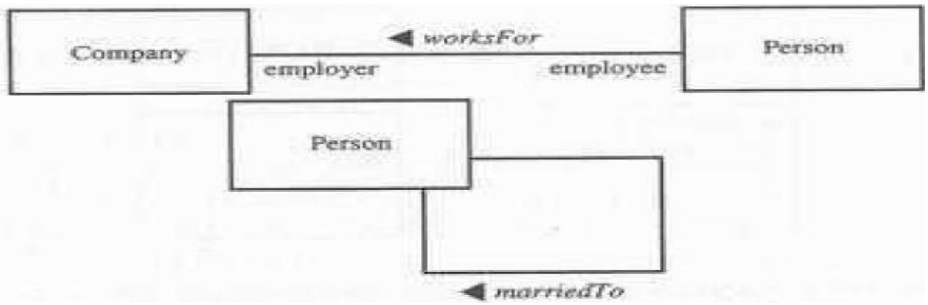
This is used to describe the externally visible behavior of a class; example, an operation with public visibility.

Interface notation of a class.



BINARY ASSOCIATION NOTATION

This is drawn as a solid path connecting two paths, or both ends may be connected to the same class. An association may have an association rule.



5. 0 Association Notation

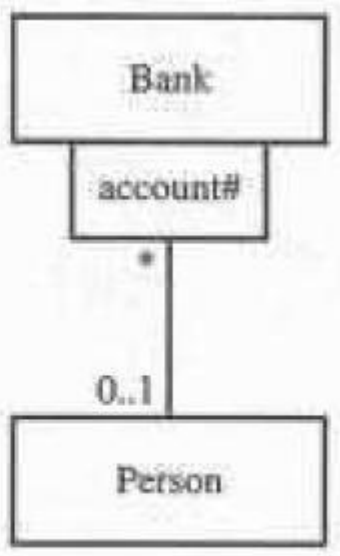
Association name may have an optional black triangle in it, the point of the triangle indicating the direction in which to read the name, where it is connected to a class is called ***association role.***

ASSOCIATION ROLE

The technical term for it is binary association-is drawn as a solid line connecting two class symbols. The UML uses the term association navigation or navigability to specify a role affiliated with each end of an association relationship.

In fig 5.0 the association is navigable in only one direction, from the Bank Account to Person, but not the. Reverse

QUALIFIER



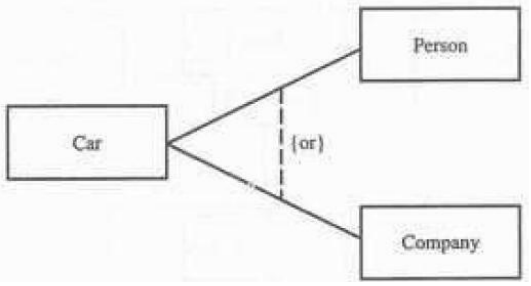
Qualifier is an association attribute. For example, a person object may be associate to a Bank object. An attribute of this association is account#. The account# is the qualifier of this association.

A qualifier is shown as small rectangle attached to end of an association path, between the final path segment and the symbol of the class to which it connects.

MULTIPLICITY

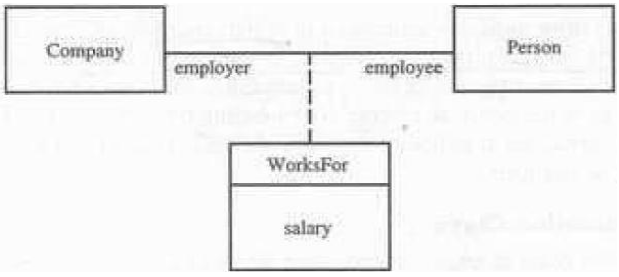
It specifies the range of allowable associated class. It is given for roles within association, parts with compositions, repetitions and other purposes. It shows as a text string comprising a period-separated sequence of integer intervals, where an interval represents a range of integers in the figure above. Example Lower bound…. Upper bound 0…..1 0..*

OR ASSOCIATION



It indicates the situation in which only one of several potential associations may be instantiated at one time for any single object. This shown as dashed line connecting two or more associations, all of which have a class common, with the constraint string {or} labeling dashed line

ASSOCIATION CLASS

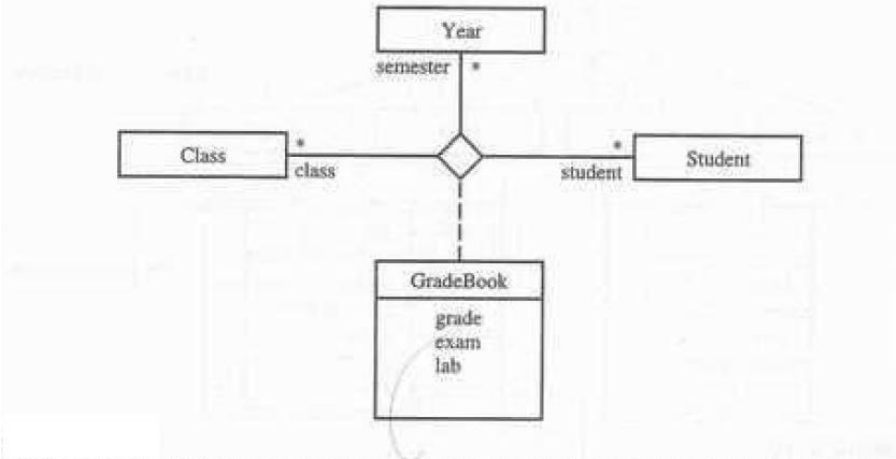


It is also has class properties. An association class is shown as a class symbol by a dashed line to an association path.

If an association class has attributes but no operation or other association, then the name may be displayed on the association path and omitted from the association class to emphasize its ***"association nature".*** If it has operations and attributes, then the name may be omitted from the path and placed in the class rectangle to emphasize its ***"class nature".***

N-ARY ASSOCIATION

This is an association among more than two classes. N-ary is difficult to understand, it is better to convert it into binary association. It is shown as a large diamond with a path from diamond to each participant class. Multiplicity may be indication; however, qualifiers and aggregations are not permitted. In the below figure diamond by a dashed line, indicating N-ary association that has attributes, operations and associations.



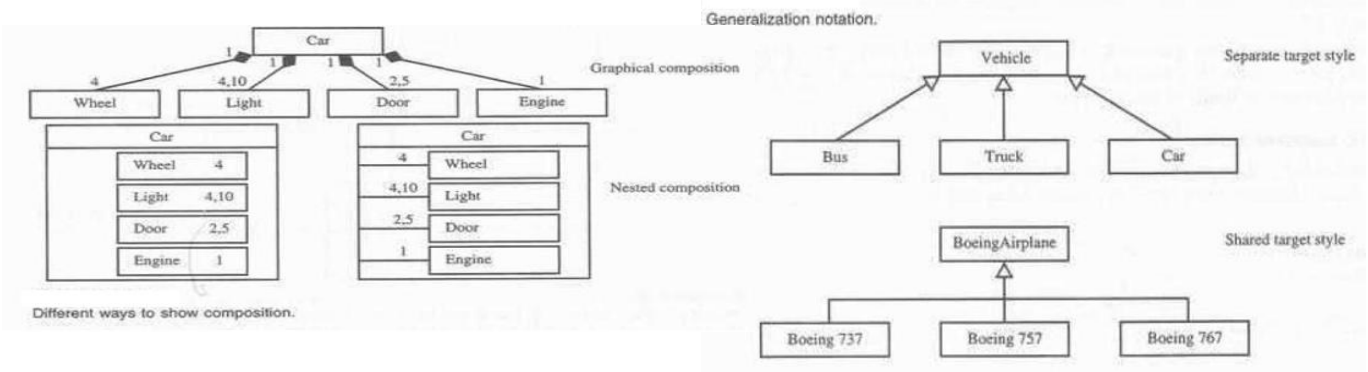
An n-ary (ternary) association that shows association among class, year, and student classes. The association class GradeBook which contains the attributes of the associations such as grade, exam, and lab.

AGGREGATIONS AND COMPOSITION An aggregation is in the form of associations. A hollow diamond is attached at the end of the path indicates aggregation. Diamond may not be attached both ends.

Composition is also known as ***a part-of,*** is a form of aggregation with strong ownership to represent the component of a complex object. Composition also referred as ***a part-whole relationship***

GENERALIZATION

Generalization is the relationship between a more general class and more specific class. It is displayed with a directed line with closed, hollow arrowhead at the super class end.



The UML allows discriminator label to be attached to a generalization of the superclass. In the above example Ellipse (…) indicates the generalization is incomplete and more subclass exist. The constructor complete indicates that generalization is complete and no more subclasses are needed.

INTERACTION DIAGRAM Interaction diagram describes how group of objects collaborate to get the job done. It captures the behavior of a single use case, showing the pattern of interaction among objects. There are two kinds of interaction models:

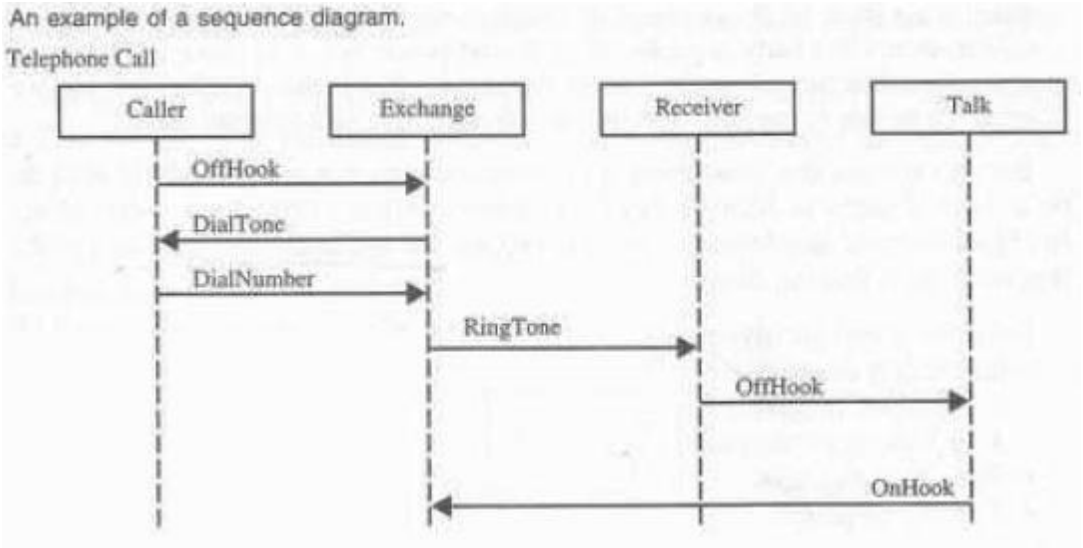
1. Sequence diagram
2. Collaboration diagram

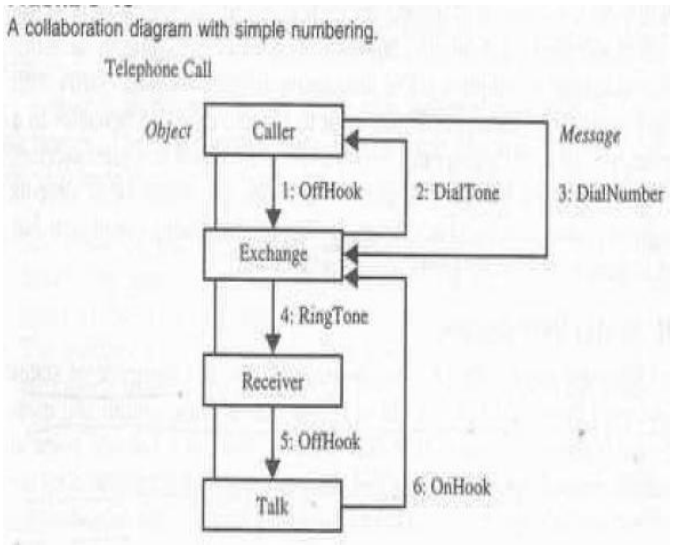
UML SEQUENCE DIAGRAM Sequence diagram describes the behavior of the system viewing the interaction between the system and its environment. It shows an interaction arranged in a time sequence. It has two dimensions:

1. Vertical dimension represents time – called object ***lifeline object existence during interactions).***
2. Horizontal dimension represents different objects.

The sequence of execution in an object-oriented program is complicated; it is hard to follow the flow of control as it is passed backwards and forwards between objects.

This demonstrates the usefulness of the sequence diagram as a map to guide us through the code. Sequence diagrams provide an overview of the inter-object messaging sequence; this is useful for software designers and for programmers, both when they are writing the code and when they are maintaining it.





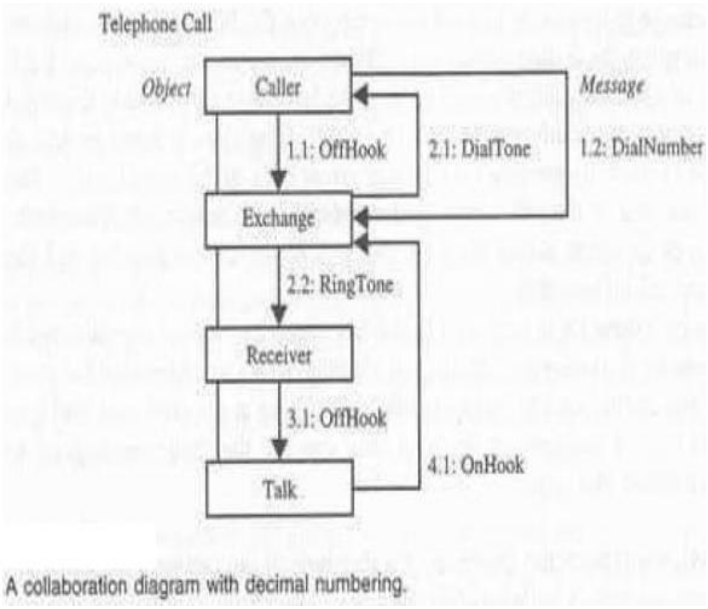
Each message is represented by an arrow between the lifelines of two objects. Each message is labeled with the message name. The sequence diagram is very simple and has immediate visual appeal- this is its great strength.

UML COLLABORATION DIAGRAM It represents a collaboration, which is a set of

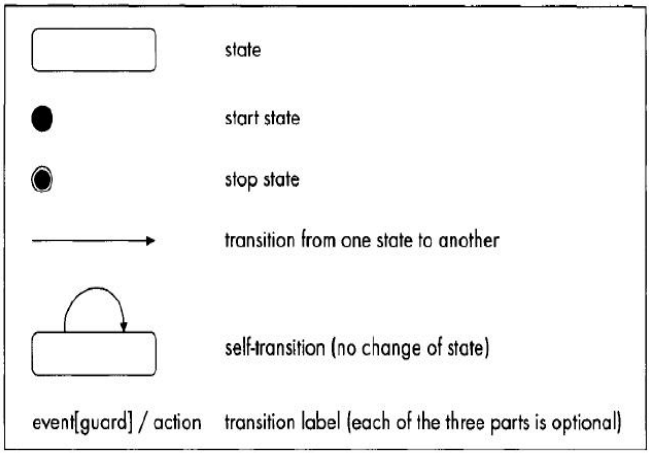
objects related in a particular context, and interaction, which is a set of objects related in a particular context, and interaction,

which is a set of messages exchanged among the objects within the collaboration to achieve a desired outcome. In this diagram, the sequence is indicated by numbering the messages, makes it more difficult to see the sequence than drawing the lines on the page. This layout uses to indicate how objects are statistically connected. It is more compressed.

An interaction diagram is used to examine the behavior of objects within a single use case. It is good at showing collaboration among the objects but not so good at precise definition of the behavior. The interaction diagram loses its clarity with more complex conditional behavior. It is simplicity.

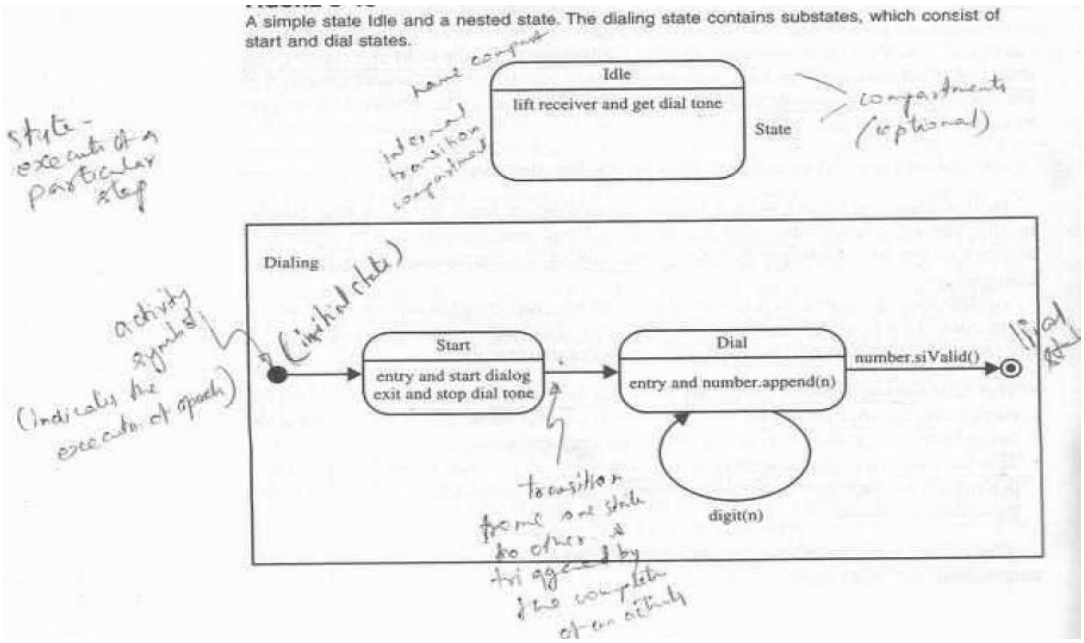


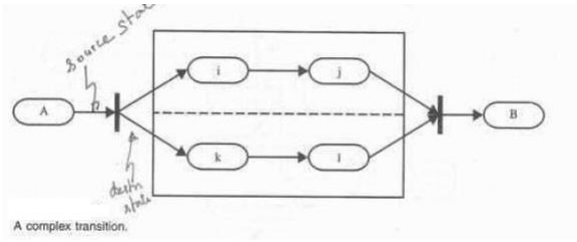
A collaboration diagram with decimal numbering.



STATE DIAGRAM It shows the sequence of states that an object goes through during its life in response to outside stimuli and messages. The state is set of values describes an object in a specific point in time and is represented at state symbol and transitions are represented by arrows connecting the state symbols.

It may contain sub diagrams. It represents the state of the method execution (the state object executes the method), and activities in the diagram represents the activities of the object that performs the method.





A complex transition.

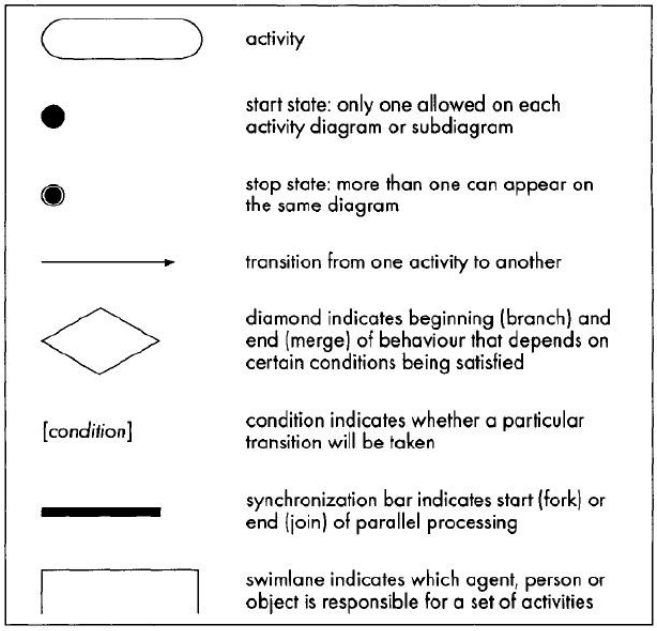
The transitions can be simple or complex. Events are processed one at a time. An event that triggers no transitions is simply ignored. A complex transition may have multiple source and target states. It represents synchronization or splitting of control into concurrent threads. A complex transition is shown as a short heavy bar.

A bar may have one or more solid arrows from states to the bar (these are source states); It may also have one or more solid arrows from the bar to the states (these are destination states). State diagrams are useful, when you have a class that is very dynamic. It emphasizes the use of events and states to determine the overall activity of the system.

ACTIVITY DIAGRAM

An activity diagram, all the states are activities (i.e. a state of doing something) and the transitions between them are triggered by the completion of the activity, rather than by an external event. Activity diagrams show the internal flow of control in a process. Symbols used in activity diagram

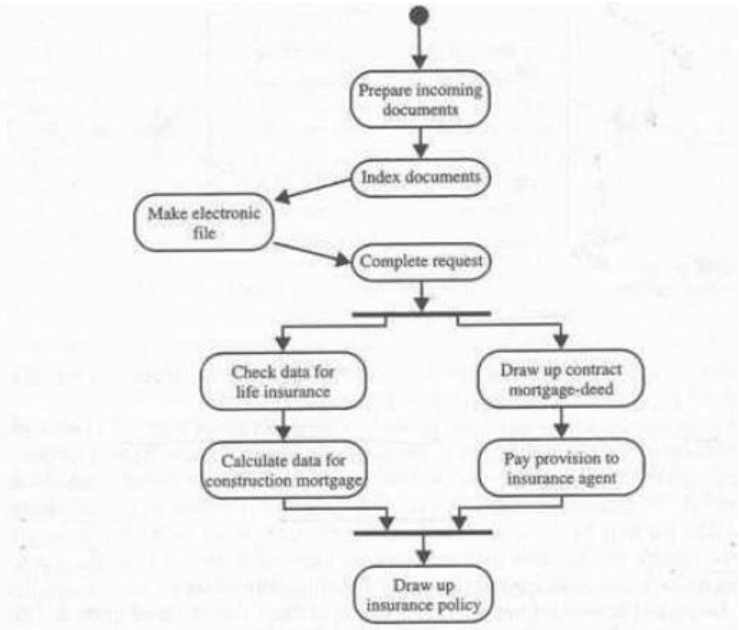
Unlike state diagram that focus on the event occurring to a single object as it responds to messages, an activity diagram can be used to model an entire business process; used to



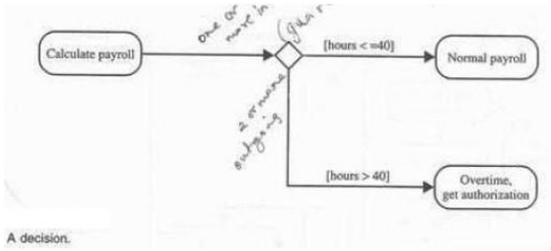
Activity diagram symbols

provide view of flow and what is going on inside the use case or among several classes

It is similar to state chart where a token represents operations. An operation appears within the activity diagram indicates the execution of the operation. An outgoing solid arrow attached to an activity symbol indicates a transition triggered by the completion of the activity. The name of this implicit event need not be written, but the conditions that depend on the result of the activity or values may be included. If conditions are not disjoint, then the branch is nondeterministic. The concurrent control is represented by multiple arrows leaving a synchronization bar, which is represented by a short thick bar with incoming and outgoing bars. This diagram mostly to show internal state of an object, but external events may appear in them. An external event may appears when the object is in a "wait state"(no internal activity by the object is waiting for external event to occur) there are 2 states: ***wait and activity state .***A wait is a ***"normal state"***



An activity diagram for processing mortgage requests (Loan: Processing Mortgage Request).

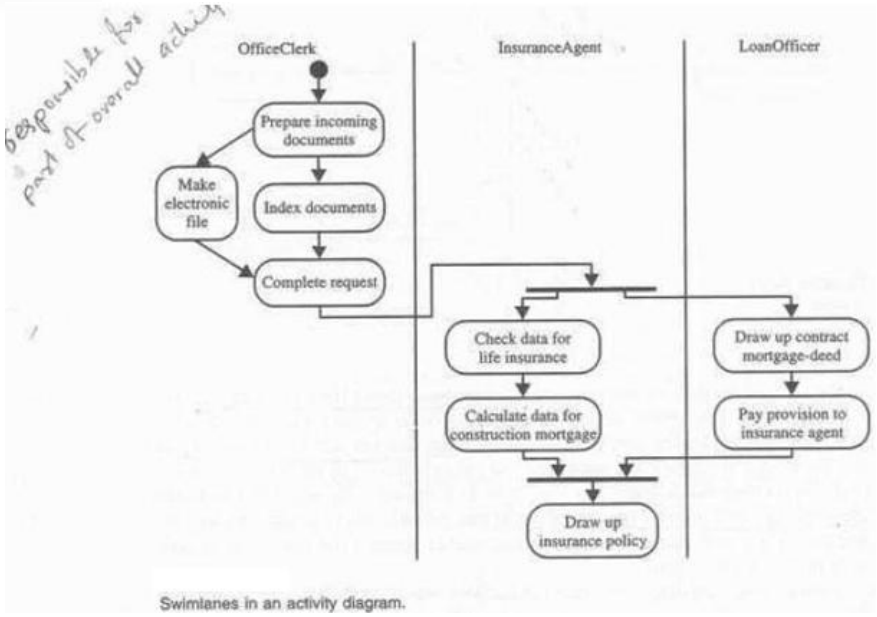


A decision.

The above diagram provided for a decision is the traditional diamond shape, with one or more incoming arrows or two or more outgoing arrows, each labeled by a distinct guard condition. This diagram may be organized into swimlanes, each separated from neighboring swimlanes by vertical solid lines on both sides.

Swimlanes

Represents responsibility for part of the overall activity and may be implemented by one or more objects. The relative ordering of the swimlanes has no semantic significances but might indicate some affinity. Each action is assigned to one swimlane.



Swimlanes in an activity diagram.