Discuss various types of Models:

Dynamic Model→ The dynamic model describes those aspects of the system concerned with the sequencing of operations and time - events that cause state changes, sequences of events, states that define the context for events and the organization of events and states. The dynamic model captures control information without regard for what the operations act on or how they are implemented. The dynamic model is represented graphically by state diagrams. A model of the dynamic behaviour of a user object. It defines significant states of the user object, the way that actions depend on the state, and affect the state. The dynamic model consists of a dynamic model diagram, showing states and transitions and supplementary notes, specifying states and actions in more detail.

Object Model→ Object models are used for describing the objects in the system and their relationship among each other in the system. The object model focuses on logical data structures. Each object model consists of many classes, associations, generalizations, and attributes. Object models are effective for communicating with application experts and reaching a consensus about the important aspects of a problem. Object models help developers achieve a consistent, understandable, efficient, and correct database design. object models contain the definition of objects in the system, which includes: the object name, the object attributes, and the objects relationships to other objects.

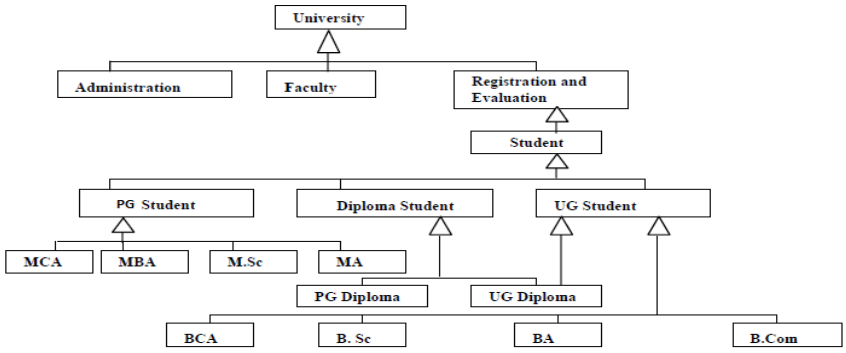


Figure Object model for university system

Function Model→ The functional model specifies the results of a computation specifying how or when they are computed. The data transformations in the system are described by a functional model. The functional model is the main model for such programs, although the object model is important for any problem with nontrivial data structures. Many interactive programs also have a significant functional model. By contrast, databases often have a trivial functional model, since their purpose is to store and organize data, not to transform it. For example, spreadsheet is a kind of functional model. The only interesting object structure in the spreadsheet is the cell. The aim of the spreadsheet is to specify values in terms of other values.

Object Modeling→ Object modelling is very important for any object oriented development, object modeling shows the static data structure of the real world system. Basically, object modeling means identifying objects and classes of a system. It describes real world object classes and their relationships to each other. To develop an object model first identify the classes and their associations as they affect the overall problem structure and approach. Then prepare a data dictionary. i) Identify associations between objects. ii) Identify attributes of objects and links. iii) Organise and simplify object classes using inheritances.

Dynamic Modeling→ In Dynamic modeling a way of describing how an individual object responds to events, either internal events triggered by other objects or external events triggered by the outside world. The dynamic model consists of a dynamic model diagram, showing states and transitions and supplementary notes, specifying states and actions in more detail. Process of Dynamic modeling are: • Analyse applicability of actions • Identify object states • Draw dynamic model diagram • Express each state in terms of object attributes • Validate dynamic model

Functional Modeling→ Functional model shows how values are computed. It describes the decisions or object structure without the regard for sequencing. It gives dependency between the various data and the functions that relate them, giving the flow of data. Each process needs to be implemented as an operation in one or more of the objects. Each data item arising from an object must have a corresponding attribute, or set of attributes in the source object. Steps in constructing a Functional Model are: • Identify input and output values • Build data flow diagrams showing functional dependencies • Describe functions • Identify constraints • Specify optimization criteria.

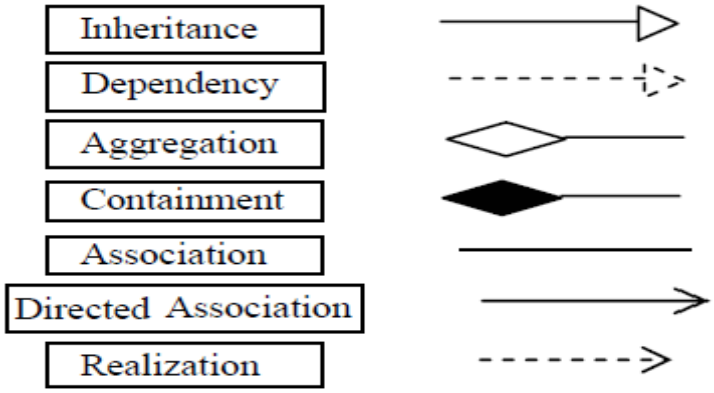
Discuss about Abstraction, Polymorphism and Encapsulation :

Abstraction→ Abstraction in object orientation is a concept which provide opportunity to express essential properties of the object without providing much details of implementation of these properties. Abstraction is one of the very important concepts of object oriented systems Abstraction focus on the essential, inherent aspects of an object of the system. It does not represent the accidental properties of the system. Abstraction helps to focus on what an object is supposed to do, before deciding how it should be implemented. The use of abstraction protects the freedom to make decisions for as long as possible, by avoiding intermediate commitments in problem solving. Generalization and inheritance are powerful abstractions for sharing the structure and/or behaviour of one or more classes.

Polymorphism→ Polymorphism means one thing having many forms. It means it is the ability to take multiple forms for different operations. There are two types of polymorphism 1. Compile time polymorphism 2. Run time polymorphism. When all information's are available at compile time for association it is called compile time polymorphism. Compile time polymorphism is also called compile time binding or static binding or early binding. The major advantages of this polymorphism is its efficiency because all associations are completed at compile time. Function overloading and operator overloading are example of compile time polymorphism. When information's are not decided at compile time and association is completed by the run time system, it is called run time polymorphism. Run time polymorphism is also known as run time binding or dynamic binding or late binding. The major advantage of late binding is its flexibility. Virtual function is an example of run time polymorphism.

Encapsulation→Encapsulation, or information hiding, is the feature of separating the external aspects of an object, from the internal implementation details of that object. It helps in hiding the actual implementation of characteristics of objects. Encapsulation helps in system enhancement. This separates the interface of an abstraction from its implementation. To hide the details of a class, you can declare that data or implementation in its private part so that any other class clients will not be able to know about. It will have the ability to change the representation of an abstraction (data structures, algorithms) without disturbing any of its clients. **Difference between Relational Database and Object Oriented Database→**

Relational Databases	Object-Oriented Databases
• Based on mathematical principles called relational algebra • Data are represented by a two dimensional table with columns and rows • Implements standard query language called SQL • Most RDBMS supports various constraints, like referential integrity.	• Supports all fundamental object modeling concepts: Classes, Attributes, Methods, Associations, Inheritance • Support for complex objects • Provides for mapping an object model to an OO-database • Determine which objects are persistent • Perform normal requirement analysis and object design • Create single attribute indices to reduce performance bottlenecks.



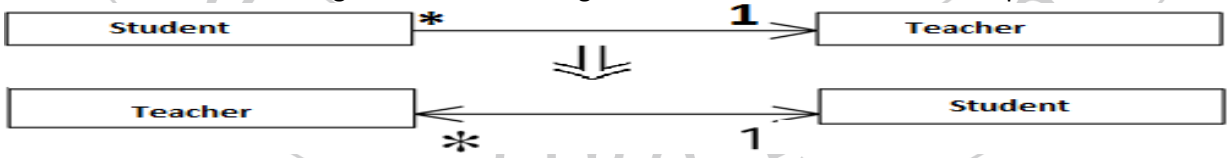
Common relationship types

Explain the Object Oriented System development approach in brief.

It is entirely a new way of thinking about problems. This methodology is all about visualizing the things by using models organized around real world concepts. Object oriented models help in understanding problems, communicating with experts from a distance, modeling enterprises, and designing programs and database. with the increasing complexity of systems, importance of modeling techniques increases. Because of its characteristics Object Oriented Modeling is a suitable modeling technique for handling a complex system. There are several characteristics of object-oriented technology. Abstraction, Encapsulation, Polymorphism, Sharing of Structure and Behaviour.

What is the advantage of two-way association over one-way association? Explain with the help of an example.

Association is a relationship which describes the reasons for the relationship and the rules that govern the relationship.Let's take an example of Teacher and Student. Unidirectional Association is a specialized form of association where one object is associated with another object, but the reverse is not true. It is like one way communication. Bidirectional Association is a type of association where one object is related with other objects and the reverse is also true. It is like two way communication.Let's take examples of multiple students can associate with a single teacher and a single student can associate with multiple teachers.



Differentiate Association, Aggregation and Composition -

Association→Association is a relationship between two classes. In other words, association defines the multiplicity between objects. We have one-to-one, one-to-many, many-to-one, many-to-many types of association seen in a class and all these words define an association between objects. Aggregation is a special form of association. Composition is a special form of aggregation.

Example: A Student and a Faculty are having an association.

Aggregation→Aggregation is a special case of association. A directional association between objects. When an object 'has-a' another object, then you have got an aggregation between them. Direction between them specified which object contains the other object. Aggregation is also called a "Has-a" relationship.

Composition→Composition is a special case of aggregation. In a more specific manner, a restricted aggregation is called composition. When an object contains the other object, if the contained object cannot exist without the existence of container object, then it is called composition.

Example: A class contains students. A student cannot exist without a class. There exists composition between class and students.

Inheritance Adjustment→As object design progresses, the definitions of classes and operations can often be adjusted to increase the amount of inheritance. In this case, the designer should: • Rearrange and adjust classes and operations to increase inheritance • Abstract common behavior out of groups of classes• Use delegation to share behavior when inheritance is semantically invited.

Discuss the role of Object model and Dynamic model in object oriented modelling.

A dynamic model represents the behaviour of an object over time. It is used where the object's behaviour is best described as a set of states that occur in a defined sequence. The components of the dynamic model are: States.

An object model is a logical interface, software or system that is modeled through the use of object-oriented techniques. It enables the creation of an architectural software or system model prior to development or programming. An object model is part of the object-oriented programming (OOP) lifecycle. The properties of objects in general in a specific computer programming language, technology, notation or methodology that uses them. Examples are the object models of Java, the Component Object Model (COM), or Object-Modeling Technique (OMT).

Differentiate Data Abstraction, Aggregation and Generalization -

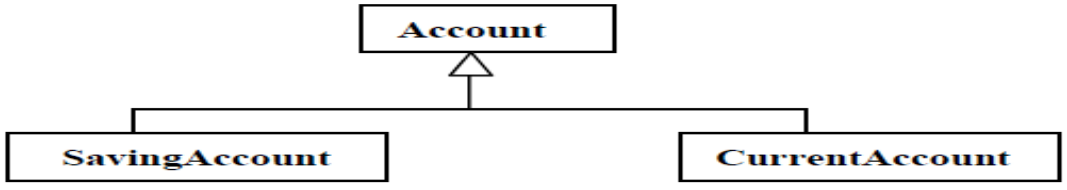
Data Abstraction→Data abstraction is the process of selecting important data sets for an Object in the software , and leaving out the insignificant ones. Data abstraction is one of the most essential and important feature of object oriented programming in C++. Abstraction means displaying only essential information and hiding the details. Data abstraction refers to providing only essential information about the data to the outside world, hiding the background details or implementation. Consider a real life example of a man driving a car. The man only knows that pressing the accelerators will increase the speed of car or applying brakes will stop the car but he does not know about how on pressing accelerator the speed is actually increasing, he does not know about the inner mechanism of the car or the implementation of accelerator, brakes etc. in the car. Abstraction is used as Abstraction using Classes, Abstraction using Header files, Abstraction using access specifiers.

Aggregation→An aggregation is a structural relationship that specifies that one class represents a large thing which constitute of smaller things. It represents "has-a" relationship. In UML, it is shown as association with an open diamond at the large end. The notation for aggregation is shown as:

Aggregation

Aggregation is a stronger form of association. It represents has-a or part-of relationship. An aggregation association depicts a complex object that is composed of other objects. Aggregation is a concept that is used to express "part of" types of associations between objects. An aggregate is, a conceptually, an extended object viewed as a Unit by some operations, but it can actually be composed of multiple objects. One aggregate may contain multiple whole-part structures, each viewable as a distinct aggregate.

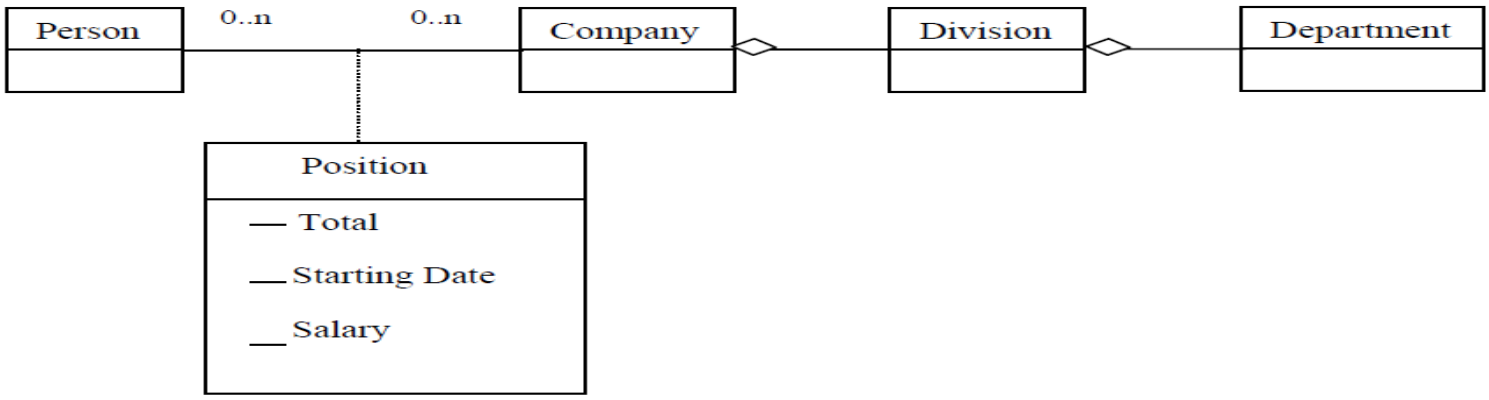
Generalization→A **generalization** is a relationship between a general thing and a specific kind of thing. It is also called "is-a-kind-of" relationship. Inheritance may be modeled using generalization. In UML, it is shown as a solid directed line with a large open arrow pointing to the parents.



Generalization of account class

What is a composition and what is its relation with aggregation?

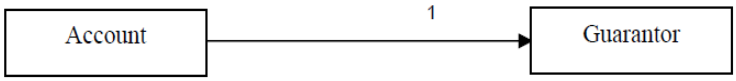
A stronger form of aggregation is called composition, which implies exclusive ownership of the part of classes by the whole class. This means that parts may be created after a composite is created, but such parts will be explicitly removed before the destruction of the composite. In UML, filled diamonds indicate the composition relationship.



Example of a composition

Association and its types→ Association is used for establishing relationships between classes. Association describe links between (among) classes. Associations are useful for finding access paths between objects in the system. During object design itself, we should implement associations. The association is one-way association or two-way association. One-way Associations→ When an association is traversed in only one direction, then it is implemented as a pointer, i.e., an attribute that contains an object reference.

One-to-One Association→ Some of the properties of associations can be implemented directly by providing suitable declarations of data members in the relevant classes. Other semantic features of an association can be enforced by providing only a limited range of operations in the class's interface, or by including code in the implementation of member functions that ensures that the necessary constraints are maintained.



A one-to-one association

This association describes a situation where bank accounts must have a guarantor who will pay any debts incurred by the account holder in exceptional conditions. It may frequently be needed to find out the guarantor of an account, but it is not necessary to find the details of the account of the guarantor for which s/he is responsible. So, the implementation of the association will be only in the direction from account to guarantor.

Two-way Associations→ Two-way association has independent of classes. It is Useful for existing predefined classes which are not modified. It is traversed in both directions. It can be implemented by using the following three methods:

- Implement as an attribute in one direction only, and perform a search when a backward traversal is required.
- To implement the attributes in both directions
- An association object is a set of pairs of associated objects stored in a single variable-size object.

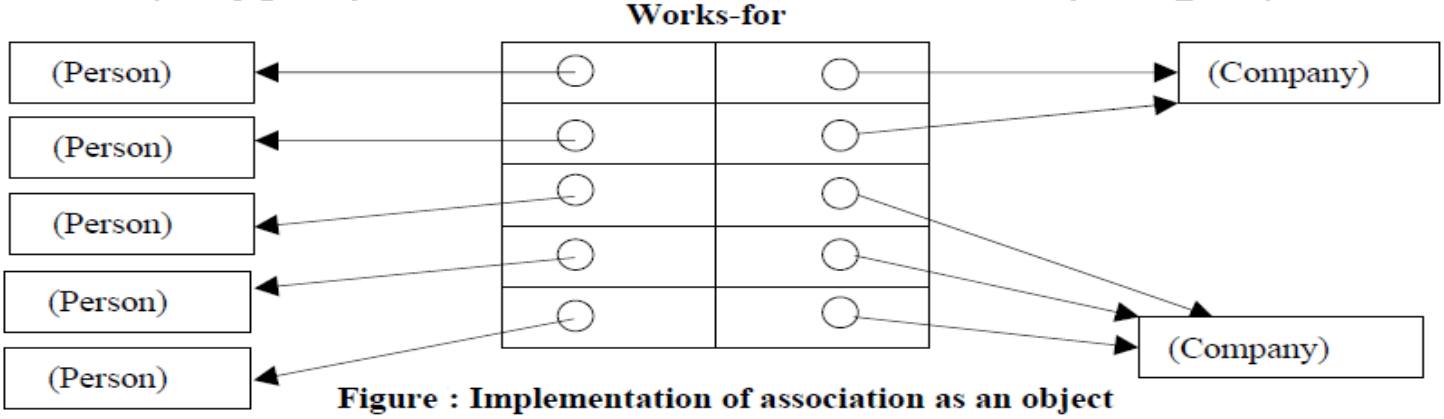
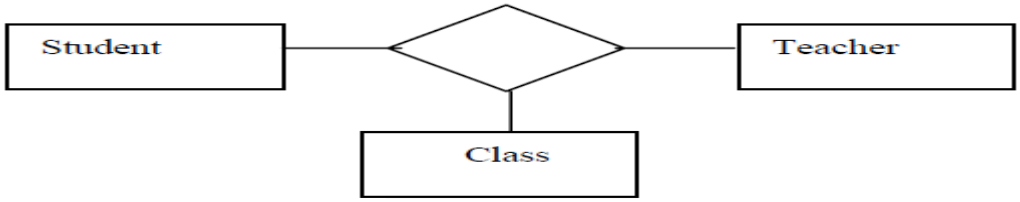


Figure : Implementation of association as an object

Ternary association→ A ternary association is formed compulsion; they cannot be converted into binary association. If a ternary association is decomposed in some other association; some information will be lost.



Ternary association

Multiplicity and its types→Multiplicity in an association specifies how many objects participate in a relationship. Multiplicity decides the number of related objects. Multiplicity is generally explained as "one" or "many," but in general it is a subset of the non-negative integers.

Types of multiplicity

0..1	Zero or one
1	One only
0..*	Zero or more
1..*	One or more
n	Only *n* (where *n* > 1)
0..n	Zero to *n* (where *n* > 1)
1..n	One to *n* (where *n* > 1)

Justify that "Aggregation is a special form of Association." Use suitable example to justify your answer.

Aggregation is a special form of association, which models the "part-whole" or "a part of" relationship as an aggregate (the whole) and parts. The most considerable property of aggregation is transitivity, that is, if X is part of Y and Y is part of Z, then X is part of Z. Aggregation is seen as a relationship in which an assembly class is related to component class. In this component objects are not having separate existence, they depend on composite objects. Exam Schedule is not having separate existence.

Association and whole-part relationship

What is the UML notation for the following? Explain briefly.

(i) Interface→An **interface** is a collection of operations that are used to specify a service of a class or a component.

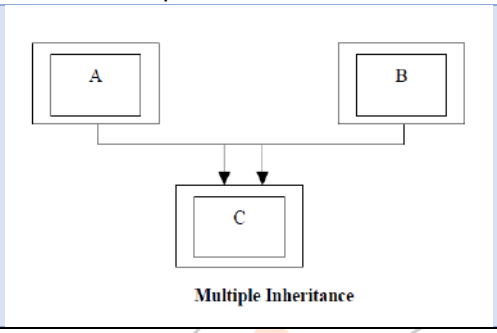
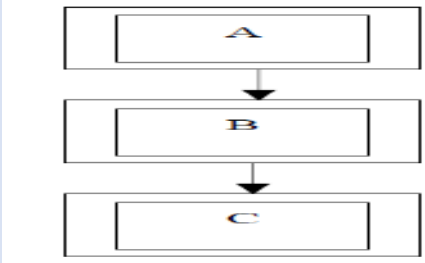
Realizing an interface

(ii) Aggregation→An **aggregation** is a structural relationship that specifies that one class represents a large thing which constitute of smaller things. It represents "has-a" relationship. In UML, it is shown as association with an open diamond at the large end.

Aggregation

Difference Between

Static binding	Dynamic binding
Events occur at compile time are "Static Binding".	Events occur at run time are "Dynamic Binding".
All information needed to call a function is known at compile time.	All information need to call a function come to know at run time.
Time execution is fast.	Time execution is slow.
Alternate name is Early Binding.	Alternate name is late binding.
It's advantages are efficiency.	It's advantages are flexibility.
The Early Binding just means that the target method is found at compile time.	while in Late Binding the target method is looked up at run time.

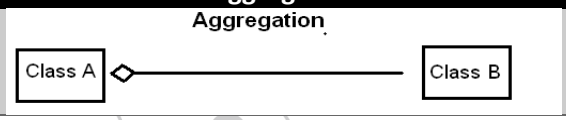
Multiple Inheritance	Multi-level Inheritance
Multiple inheritance extends this concept to allow a class to have more than one parent class, and to inherit features form all parents.	While Multi-level inheritance doesn't extend more than one parent class of a single child.
When a single child class inherited with multiple parent class, it is called multiple inheritance.	When a single parent class having single child class in different level it is called multilevel inheritance.
 Multiple Inheritance	 Multilevel Inheritance

Packages and Subsystem

Package→A package is a grouping of model elements. A package is a general purpose mechanism for organizing elements into groups. It can also contain other packages. The notation for the package shown contains name and attributes. Packages are used widely in a Java based development environment.

Subsystem→Subsystems are used for system decomposition. A subsystem is a grouping of elements of which some constitute a specification of the behavior offered by other contained elements.

Object	Class
Object is an instance of a class.	Class is a blueprint or template from which objects are created.
Object is a real world entity such as pen, laptop, mobile, bed, keyboard, mouse, chair etc.	Class is a group of similar objects.
Object is a physical entity.	Class is a logical entity.
Object is created many times as per requirement.	Class is declared once.
Purpose of object is Data abstraction and further inheritance.	Purpose of class is grouping of data.
Object allocates memory when it is created.	Class doesn't allocated memory when it is created.

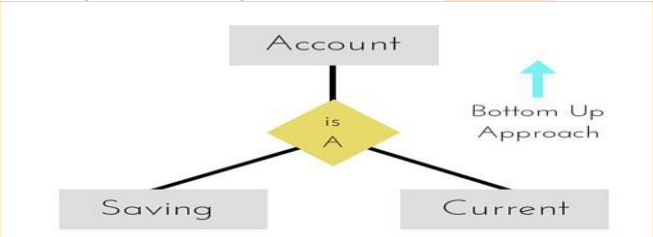
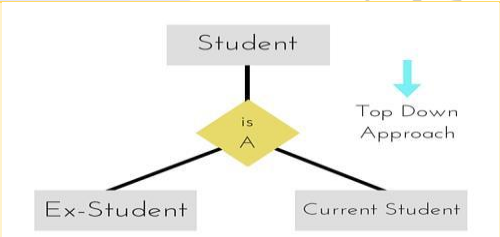
Aggregation	Association
	
Association is a relationship between two classes where one class use another.	Aggregation describes a special type of an association.
Usually called as "has-a" relationship.	Usually called as "is-a" relationship.
It is a specialized form of Association where all object have their own lifecycle but there is ownership.	It represents a relationship between two or more objects where all objects have their own lifecycle and there is no owner.

Actor	Stakeholder
Actors are always stakeholders.	But stakeholders are not actors.
Actors may appear on use cases.	stakeholders may not appear on use cases.
Examples of actors: end users, user roles, persons, companies, organizations, networks, computers that the system needs (hardware), databases, applications (software), etc.	Examples of stakeholders: end users of the system, investors, suppliers, customers, vendors, corporations, companies, even developers of their own project.
They are external entities that interact with a system in same way, in other words, "anything" with some behavior that acts on the system.	Stakeholder may be "an individual who is materially affected by the outcome of the project" .
Actors represents a role which an user can play in our project, and they must be able to make decisions.	clear that the fact of being a stakeholder does not necessarily imply to get a benefice from the project.

Relational Database	Object Oriented Database

Relational database rely on relational model.	Object database rely on OOP.
Relational database organizes the given data in the form of tables contains rows and columns.	In Object oriented database each element resembles an object.
Relational data base handle a single data.	Object database can handle different types of data.

Abstract Class	Concrete Class
A partial implemented class is called an abstract class.	Whereas fully implemented class is commonly known as concrete or normal class.
It is the type of Base class	It is the type of Default class
Abstract class contain partially implemented methods	All methods are completely implemented
Some or all declared functions are purely virtual	No purely virtual functions
Interface implementation is possible.	Interface implementation is possible.
The abstract methods should implement in the derived classes. If not , the derived class also become an abstract class.	There is no abstract methods in any level to implement.
Variables are not final by default. We can able to reassign values.	Variables are not final by default.
It is not possible to instantiate an abstract class.	Instantiation is possible for a concrete class.
May or may not contain abstract methods.	Should not contain abstract methods.
It is must to declare a class with an abstract access modifier.	Should not declare a concrete class with an abstract access modifier.

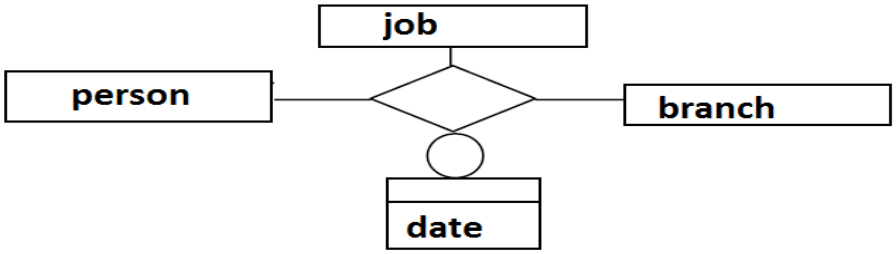
Generalization	Specialization
If many similar existing objects are combined to form a superclass to do the job of its subclass, then it is known as Generalization.	if some new subclasses are created from an existing superclass to do specific job of the superclass, then it is known as specialization.
Generalization is the process of extracting shared characteristics from two or more classes, and combining them into a generalized superclass. Shared characteristics can be attributes, associations, or methods.	If some new subclasses are created from an existing superclass to do specific job of the superclass, then it is known as specialization.
Animal is a generalization.	Dog is specialization.
when overall view is given or talking about a subject or topic by taking its broader view then it will be generalization.	when something is certain or specific then it comes under the term of "specialization".
It follows Bottom-up approach.	It follows Top-down approach.
Generalization Takes all the information that have universal nature within the entities and then forms a new entity.	Specialization Creates new objects based on the difference between the existing ones and have some features of the parents.
	

Link and Reference

The basic difference between links and references is that links are symmetrical whereas references refer only in one direction. If two objects are linked, a single link serves as a channel for sending messages in either direction. By using a reference, however, one object can send messages to another, but the other object is not aware of the object that is referring to it. So, it has no way of sending messages back to the first object. if a link has to support message passing in both directions, it will require a pair of references for the implementation for each direction.

How do you map ternary association to table? Illustrate.

Whenever a ternary relationship is there between the different classes then each class is mapped to a table with the inclusion of object ID. Apart from this a new ternary table is also created which has attributes from the different classes involved in the relationship. The attributes will the object ID of all the classes involved in the relationship and the attributes of the relation.



Object model

Candidate key: (person_ID, branch_ID, job_ID)

Ternary Table:

Attribute Name	Nulls?	Domain
person_ID	N	ID
job_ID	N	ID
branch_ID	N	ID
hours	Y	Time

Table Model

Candidate key: (person_ID, branch_ID, job_ID)
Primary key: (person_ID, branch_ID, job_ID)
Frequently accessed: (person_ID, branch_ID, job_ID)

Person Table:

Attribute Name	Nulls?	Domain
person_ ID	N	ID
person_name	N	Name
Address	Y	Address

Similarly, the job and branch tables will have the required attributes.

What is serialization? Where it is used and why?

Serialization is a generic term used for mechanisms that enable objects and object structures to be converted into a portable form, removing the volatility created by object addresses. This interface defines no methods, so in order to make a class serializable it is sufficient to declare that it implements this interface. Once this has been done, the methods 'writeObject' and 'readObject' can be used to transfer objects to and from streams, and persistence can then easily be implemented. Serialization therefore provides a convenient and straightforward way of making data persistent. It is most appropriate when the amount of data involved is relatively small. If larger amounts of data are to be stored, serialization may no longer be appropriate.

What is inheritance? Explain the different types of inheritance by giving examples.

Inheritance is one of the cornerstones of object-oriented programming language because it allows a creation of hierarchical classifications. Using inheritance, we can create a general class that defines trails common, to a set of related items. More specific classes can inherit this class, and each could add a certain unique thing to the resulting new class. Inheritance is the ability of a class to make several generic class at lower level so that lower level class will get the features of upper level class automatically.

Types of Inheritance 1. Single inheritance→A Single base class having single derived class. 2. Multiple inheritance→A single derived class inherited with multiple base class. 3. Hierarchical inheritance→A single base class having multiple derived class in hierarchical manner. 4. Multilevel inheritance→A Single base class having single derived class in different level.

What is message passing? Explain two significant benefits of message passing.

A single object alone is generally not very useful. Objects usually appear as a components of a larger program or a system. Through the interaction of these objects, functionality of systems is achieved. Software objects interact and communicate with each other by message passing to each other. When object X wants object Y to perform one of methods of object Y, object X sends a message to object Y. Message passing provide two significant benefits:

• An object's characteristics are expressed through its methods, so message passing supports all possible interactions between objects. • It closes the gap between objects. Objects do not need to be in the same process, or even on the same machine, to send and receive messages back and forth to each other.

Data Dictionary→ Data dictionary is the repository of information about data items such as origin of data, data structure, data uses and other metadata information related to data elements. It is used as system of record for structure chart and for other references. Data dictionaries help to organize and document the information related to data flows, data processes and data elements in a structure fashion. The main benefits of having data dictionaries are that it: 1. Provides a highly structured definition and details of data elements. 2. Identifies all alias and reduces duplicates within data elements. 3. Helps in developing logic for processes. 4. Helps in development in report.

Briefly discuss two disadvantages of both structured analysis and object oriented analysis approach.

Advantage of Structured Analysis

•It provides structured models for information exchange with user groups or customers. •Structured methods (functional decomposition) provide a natural vehicle for discussing, modeling, and deriving the requirements of the system.

Disadvantage of Structured Analysis

•The disadvantage with structured methods is that they do not readily support the use of reusable modules.

•The top down process of functional decomposition does not lead to a set of requirements which map well to existing component.

Advantage of Object Oriented Analysis

•Main advantage of object oriented (OO) is the focus on data relationships. • An OO model provides all of the insight of an ER diagram and contains additional information related to the methods to be performed on the data. • The OO approach inherently makes each object a standalone component that can be reused not only within a specific stat problem domain, but also is completely different problem domains, having the requirement of similar objects.

Disadvantage of Object Oriented Analysis Approach

• Disadvantage of the object oriented analysis design (OOAD) is in system modeling for performance and sizing. The object oriented (OO) models do not easily describe the communications between objects.• The object oriented (OO) analysis design itself does not provide support for identifying which objects will generate an optimal system design.

Meta class→ In object-oriented computer programming, a meta class is one whose instances are also classes. A meta class is able to modify information from the class and can be linked to one or many classes, depending on the coding structure. The use of meta classes is most prevalent in object-oriented languages in which classes are "first-class" objects; meaning they can be used just like any other object in the language:

(1) They can be named variables;

(2) They can be passed as n argument to methods, functions, and procedures;

(3) They can be returned as the result of a method, function, or procedure; and

(4) They can be included as part of a data structure.

What do you mean by object ID? What are its advantages?

Each class-derived table has an ID for the primary key, one or more object IDs form the primary key for association derived tables. An object ID is the equivalent database construct. IDs are never changing and completely independent of changes in data value and physical location. The stability of IDs is particularly important for associations since they refer to objects.

The advantages of Object IDs are as follows:

i) IDs are never changing. ii) IDs are completely independent of changes in data value and physical location. iii) IDs provide a uniform mechanism for referencing objects. iv) IDs may prevail when database access is restricted via programs.

What is object oriented decomposition of systems? Explain briefly.

Object-oriented decomposition aims at identifying individual autonomous objects that encapsulate both a state and a certain behavior. An object-oriented decomposition is favorable because, the object-oriented approach provides for a semantically richer framework that leads to decompositions that are more closely related to entities from the real world. Object-oriented decompositions of systems tend to be better able to cope with change. Each subsystem has a well-defined interface that communicate with rest of the system. Object-oriented decompositions are closer to the problem domain, as they directly represent the real-world entities in their structure and behavior.

Object Oriented Database→ Object oriented databases provide seamless database support for applications designed using object oriented methods. It Supports all fundamental object modeling concepts: Classes, Attributes, Methods, Associations, Inheritance. It also Support for complex objects. It Provides for mapping an object model to an OO-database. It Determine which objects are Persistent. It Perform normal requirement analysis and object design. It Create single attribute indices to reduce performance bottlenecks. Object-oriented databases provide support for all fundamental object modeling concepts like Classes, Attributes, Methods, Associations, and Inheritance. Object oriented designs are efficient, coherent, and less prone to the update problems that are not present in many other database design techniques presently. It is not used widely because only few database vendors have supported it. It is still not in the commercial stream. It is in the development stage.

Explain the use of constraints in functional model, with the help of suitable examples.

A constraint shows the relationship between two objects at the same time, or between different values of the same object at different times. A constraint may be expressed as a total function or as a partial function. For example, a co-ordinate transformation might specify that the scale factor for the x-coordinate and the y-coordinate will be equal; this constraint totally defines one value in terms of the other. Constraints can appear in each kind of model. **Object constraints** describe where some objects depend entirely or partially on other objects. **Dynamic constraints** represent relationships among the states or events of different objects. Similarly, the **functional constraints** show the restrictions on operations, such as the scaling transformation.

Code Reusability /Reuse of Code :-

•**Reusing the implementation**. Place an existing class directly inside a new class. The new class can be made up of any number and type of other objects, in any combination that is needed to achieve the desired functionality. This concept is called *composition* (or more generally, *aggregation*). Composition is often referred to as a "has-a" relationship or "part-of" relationship, as in "automobile has an engine" or "engine is part of the automobile." •**Reusing the interface**. Take an existing class and make modifications or additions to achieve desired functionality. This concept is called *inheritance*. The original class is called the Base class or Parent class.

Object Orientation→ object orientation was largely associated with the development of graphical user interfaces (GUIs), and a few other applications became widely known. In object orientation the major emphasis is on specifying the characteristics of the objects in a system, rather than implementing these characteristics. The conceptual structure of object orientation helps in providing an abstraction mechanism for modeling, which includes Classes, Objects, Inheritance, Association etc. It is the basic characteristic of object orientation which makes it possible to develop systems in such a way that the system is open for reusability. Abstraction in object orientation is a concept which provide opportunity to express essential properties of the object without providing much details of implementation of these properties. Inheritance is an object orientation concept which allow reusability of design/code. Object-orientation has the advantage of continuity throughout analysis, design implementation, and persistent representation.

Define Integrity constraints. Explain the types of Integrity constraints.

Integrity constraints→Integrity constraint defines the special properties for a column. It defines the nature of value or certain rule for a column. Following are some common types of integrity constraints: -

1. Not NULL→If a column is define not null, it must take any value. It means we can't leave the not null field.
2. Unique→If a column is defined as unique, it can't take duplicate value.
3. Primary key→Primary key is the integration of not null and unique. Primary key is used as a reference in other table to set the relation.
4. Foreign key→A foreign key is the reference of primary key and the value of foreign key is depends upon the value of primary key. A foreign key can take duplicate value. A foreign key creates the relation between two tables.

Concurrency→The fundamental concept in computer programming is the idea of handling more than one task at a time. Many programming problems require that the program be able to: Stop what it's doing, currently deal with some other problem, and return to the main process. Concurrency in objects can be identified by the way they change state. Current objects can change state independently. Aggregation implies concurrency. If more than one thread running that is expecting to access the same resources, then there is a problem. To avoid this problem, a thread locks a resource, completes its task, and then releases the lock so that someone else can use the resource. It can lock the memory of any object so that only one thread can use it at a time. It is important to handle concurrent running programs/threads properly.

Identification of concurrency: One of the important issues in system design is to find the concurrency in objects. If the objects are concurrent in nature we have to assign them to, different thread of Control. Concurrency is identified in a dynamic model. Two objects are said to be concurrent (parallel) if they can receive events at the same time. Concurrent objects are required to be assigned to different threads of control.

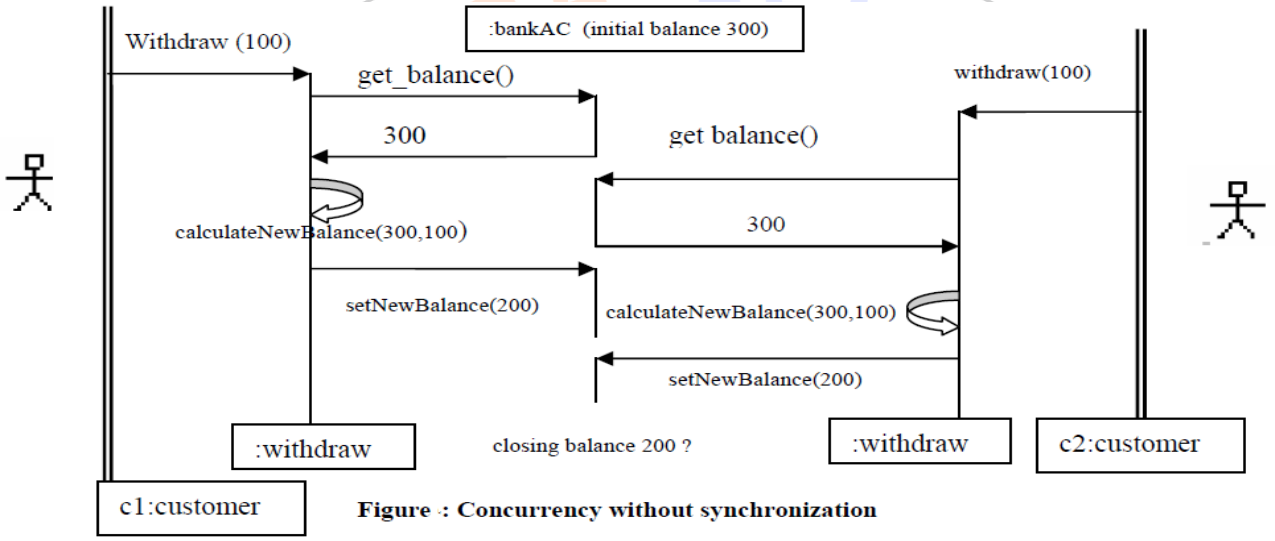


Figure : Concurrency without synchronization

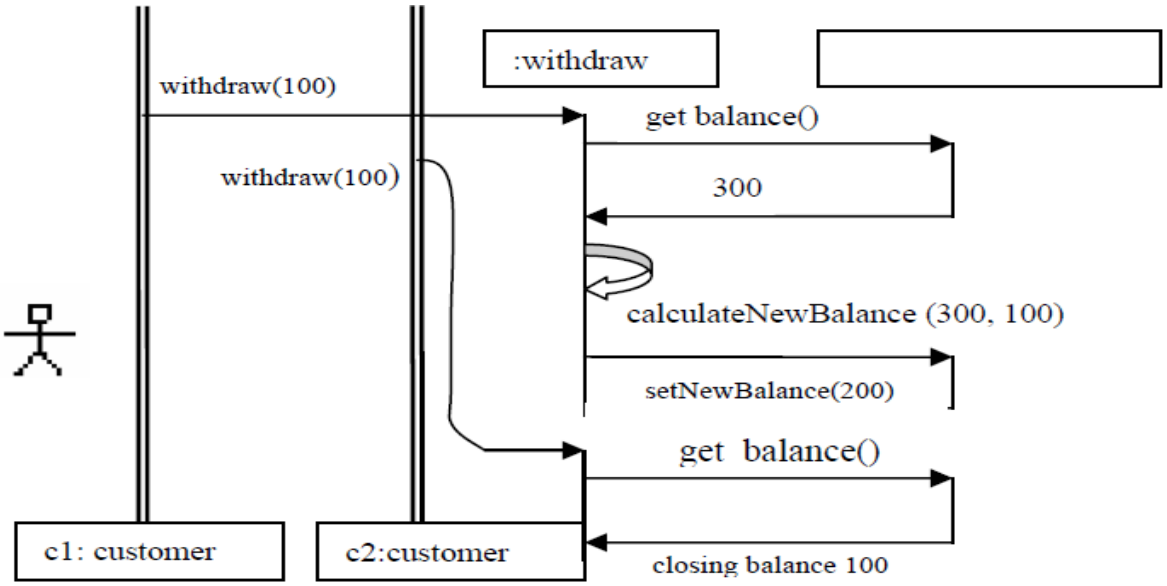


Figure : Concurrency with synchronization

Concurrency issues

• Data integrity: Threads accessing the same object need to be synchronized, for example: banking account.

• Deadlock: One or more threads in the system are permanently blocked Example: Thread A waiting on Thread B, which is waiting on Thread A.

• Starvation: A thread is not getting enough resources to accomplish its work Example: All requests from one user are being handled before another user requests.

How to handle concurrency:

Mechanisms are: • Locks • Semaphores • Monitors • Synchronized methods

Methods are: • Deadlock avoidance • Verification • Simulation

Keys are: •Develop a clear strategy for dealing with all concurrency issues during system design. •Concurrency must be dealt with during the design process as dealing with concurrency after the system is implemented is difficult.

What is persistency? Explain with an example. How can persistent data be identified.

Persistent data is data which has a longer lifetime than the program created it. In the context of an object oriented program, this means that it must be possible to save the objects created in one run of a program and to reload them at a later date. The user should not have to create all the data used by a program from scratch every time the program is run. The usefulness of the program will be rather limited if it is not possible to save diagrams to disk and to continue working on them at a later date. Enabling data to be stored on a permanent storage medium provides persistency. The most common techniques used are to store data is in the form of files or to make use of a back-end database system.

Identifying Persistent Data → The basic problem is that it may not be always clear from a model exactly what data needs to be persistent. Models in UML are not restricted to describing permanent data, or database schemas, and as a result a single model can combine persistent and transient data. The only notation that UML provides for persistency is a tagged value 'persistence'. This has two values 'persistent' and 'transient' and can be applied to classes, associations and attributes.

Make your data persistent→ Enabling data to be stored on a permanent storage medium provides persistency. The most common techniques used are to store data in files, or to make use of a backend database system.

Object Oriented Modeling→ Object Oriented Modeling is a suitable modeling technique for handling a complex system. OOM basically is building a model of an application, which includes implementation details of the system, during design of the system. Object oriented models help in understanding problems, communicating with experts from a distance, modeling enterprises, and designing programs and database. Object oriented models are represented by diagrams. OOM approach is an encouraging approach in which software developers have to think in terms of the application domain through most of the software engineering life cycle. In this process, the developer is forced to identify the inherent concepts of the application. First, developer organize, and understood the system properly and then finally, the details of data structure and functions are addressed effectively.

In OOM the modeling passes through the following processes:

• System Analysis • System Design • Object Design, and • Final Implementation.

Object oriented modeling is covered by three models for a system description. These models are:

• object model→ Object models are used for describing the objects in the system and their relationship among each other in the system.

• dynamic model→ The dynamic model describes interaction among objects and information flow in the system.

• functional model→ The data transformations in the system are described by a functional model.

DESIGN OPTIMIZATION IN OBJECT ORIENTED→ To optimize the design, the following things should be done:

a) Adding Redundant Associations for Efficient Access→ Redundant associations do not add any information, thus during design we should actually examine the structure of object model for implementation, and try to establish whether we can optimize critical parts of the completed system.

b) Rearranging the Execution Order for Efficiency→Algorithm optimization is achieved by removing dead paths as early as possible. For this, we sometimes reverse the execution order of the loop from the original functional model.

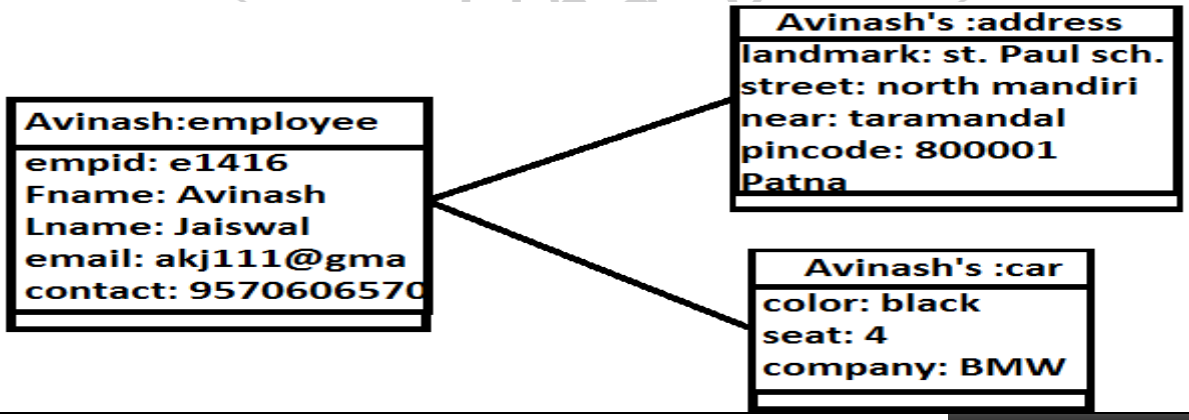
c) Saving Derived Attributes to Avoid Recomputation→ Data which is derived from other data should be stored in computed form to avoid re-computation. we can define new classes and objects, and obviously, these derived classes must be updated if any base object is changed.

Briefly discuss the advantages of two-way associations.

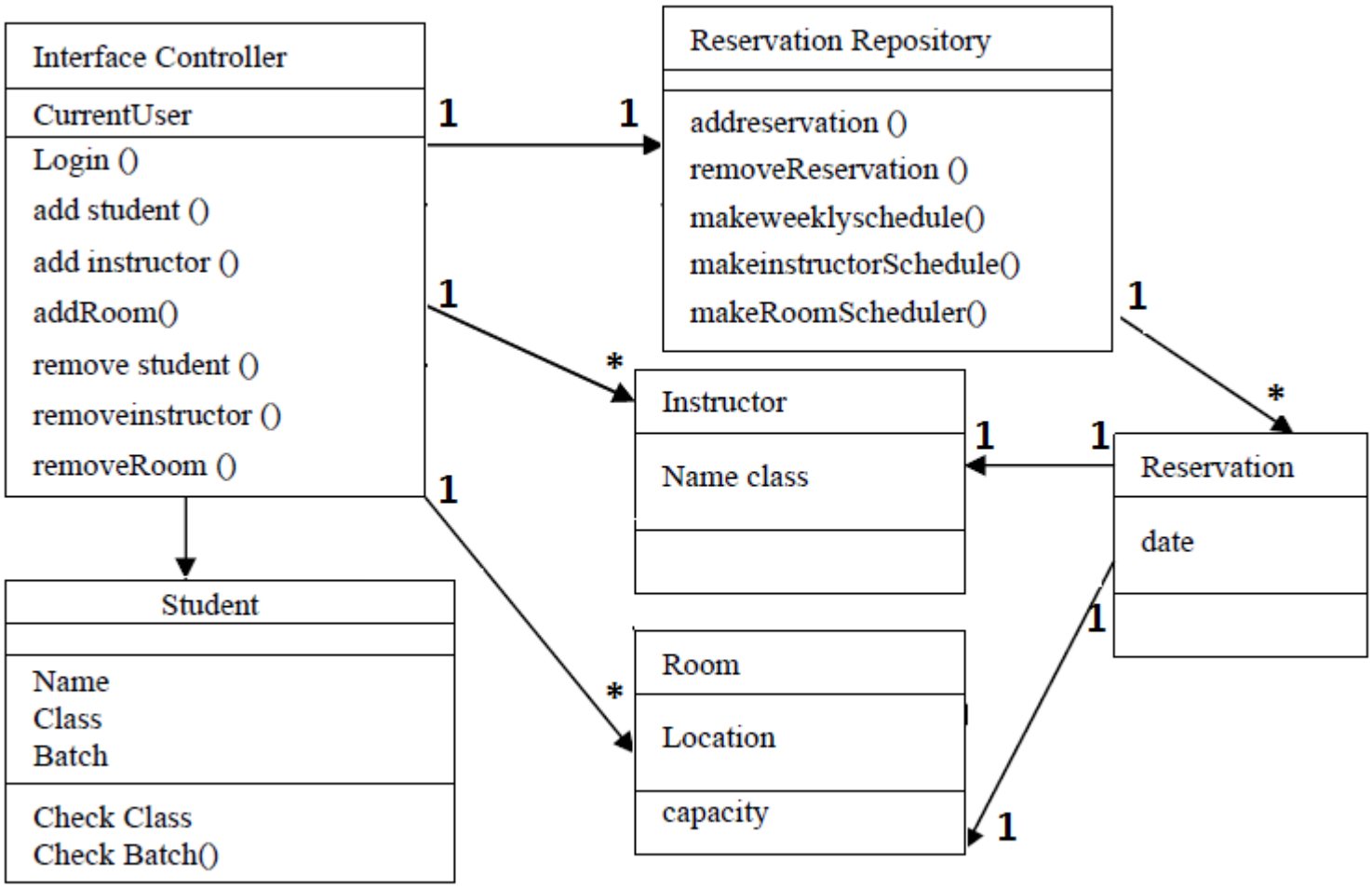
Two-way association has following advantages:

a) Independent of classes. b) Useful for existing predefined classes which are not modified.

Object Diagram→object diagram is used to model interactions that consist of objects that collaborate the without any message passed among them. It contains name, graphical contents, notes, constraints, packages and subsystems. It shows a set of objects and their relationships at a point of time.

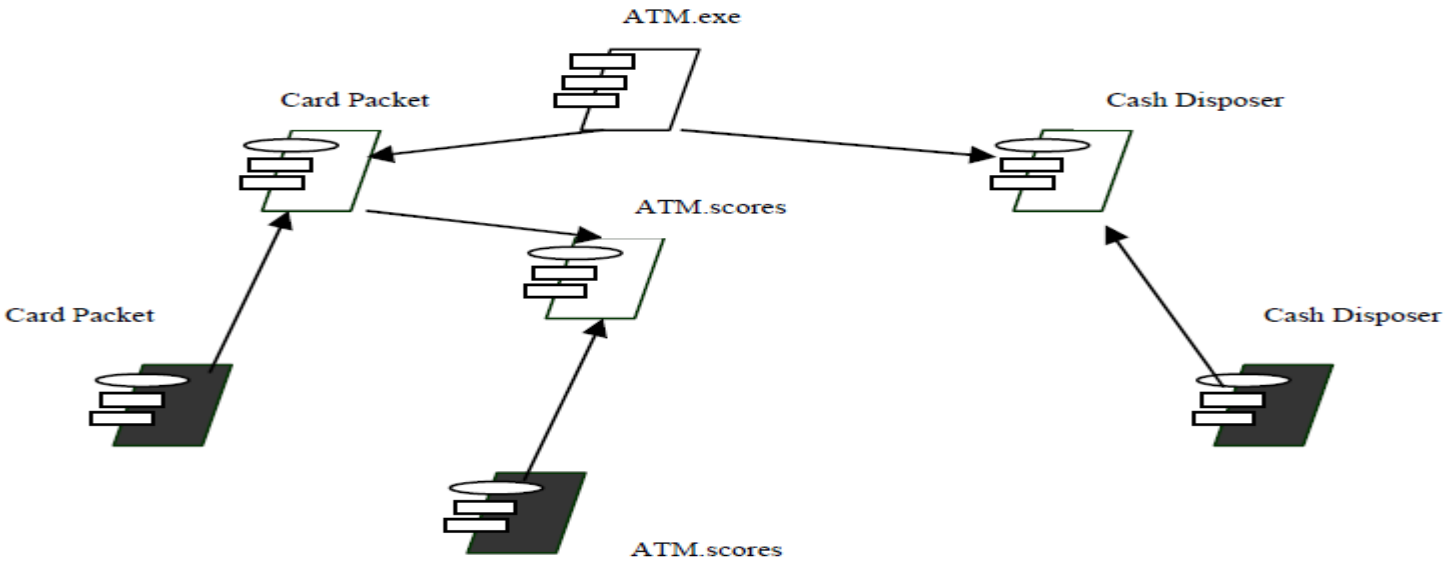


Class Diagram→Class diagram is used to model the vocabulary of the system, simple collaboration, and logical schema. It contains sets of classes, interfaces, collaborations, dependency, generalization and association relationship. Class diagram is used to support functional requirement of system.



Class diagram for a class room scheduling system

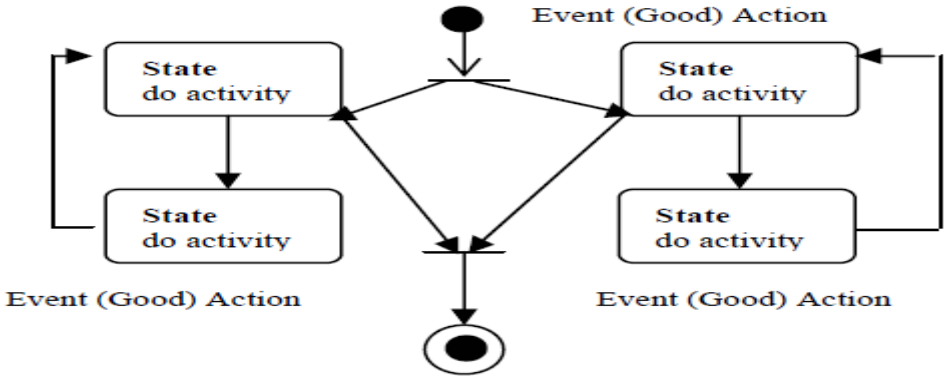
Component Diagram→This diagram depicts how components are wired together to form larger component. They are used to illustrate the structure of arbitrary complex system. This type of diagram is used in component based development to describe systems with service oriented architecture. It doesn't describe the functionality of the system but it describes the components used to make those functionalities.



Component diagram for ATM

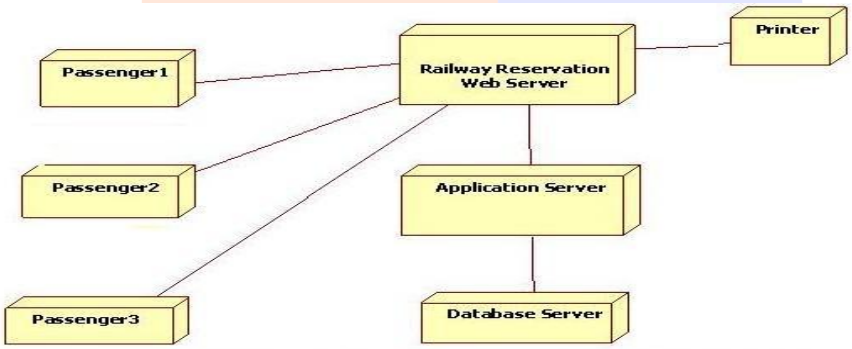
Interaction Diagram→ An interaction diagram shows an interaction, consisting of a set of objects and their relationships, including the messages that may be dispatched among them. These diagrams should be used to model the dynamic aspect of the system. It includes sequence diagrams and collaboration diagrams.

State Chart→ A state chart shows a state machine, emphasizing the flow of control from one state to another. A state machine is a behaviour that specifies the sequence of states that an object goes through during the life time in response to events together with its response to those events. A state is a condition/situation during the object's life which performs some activity, or waits for some event. It contains simple states, composite states, transitions, events, and actions.

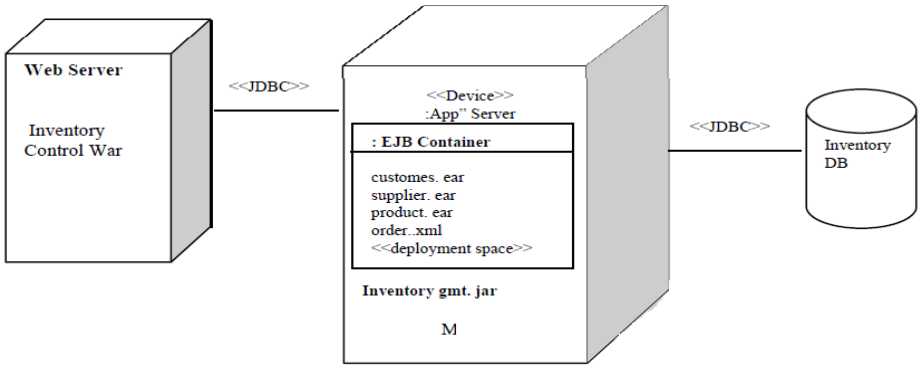


State diagram for multiple activities

Deployment diagram→ A deployment diagram shows all the nodes on the network, their interconnections, and processor execution. In a dynamic model, a deployment diagram is used to represent computational resources. It represents the physical deployment of artifacts or nodes. E.g.- to describe a website, a deployment diagram explain the nodes used in it. i.e. h/w component (eg- web server, an application server, a database server etc) and what are the s/w components run on each node(eg- web application s/w, database s/w etc) and how they are connected with each other(eg-JDBC,REST,RMI etc). The nodes appear as boxes and the artifacts allocated to each node appear as rectangles within the boxes. Nodes may have sub-nodes, which appear as nested boxes.

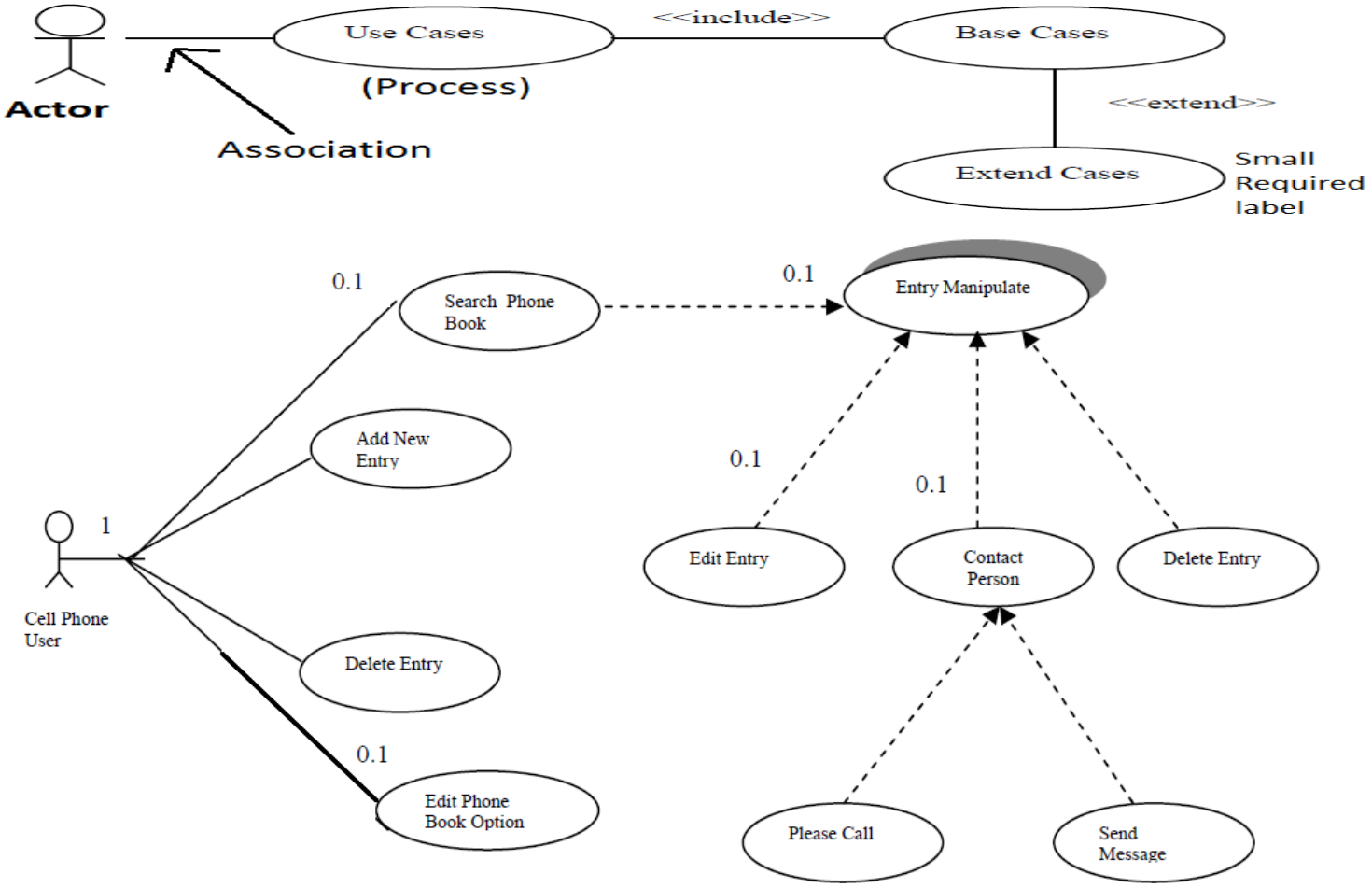


Deployment Diagram for Railway Reservation System



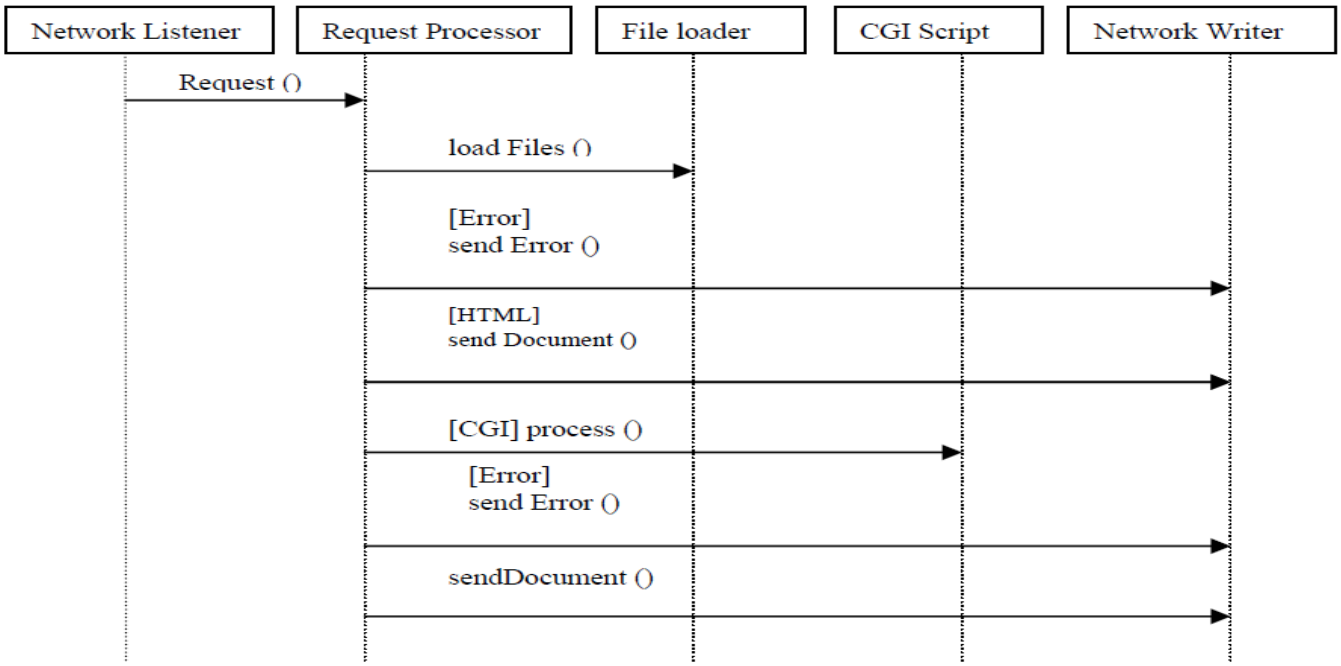
Deployment diagram

Use Case Diagram→ It is the type of behavioral diagram. It describes the behavior of system. Use case diagram shows a set of use cases, cases, actors, and their relationships. These diagrams should be used to model the context or the requirement of a system. It contains use cases, actors, dependency, generalization, association, relationship, roles, constraints, packages, and instances.



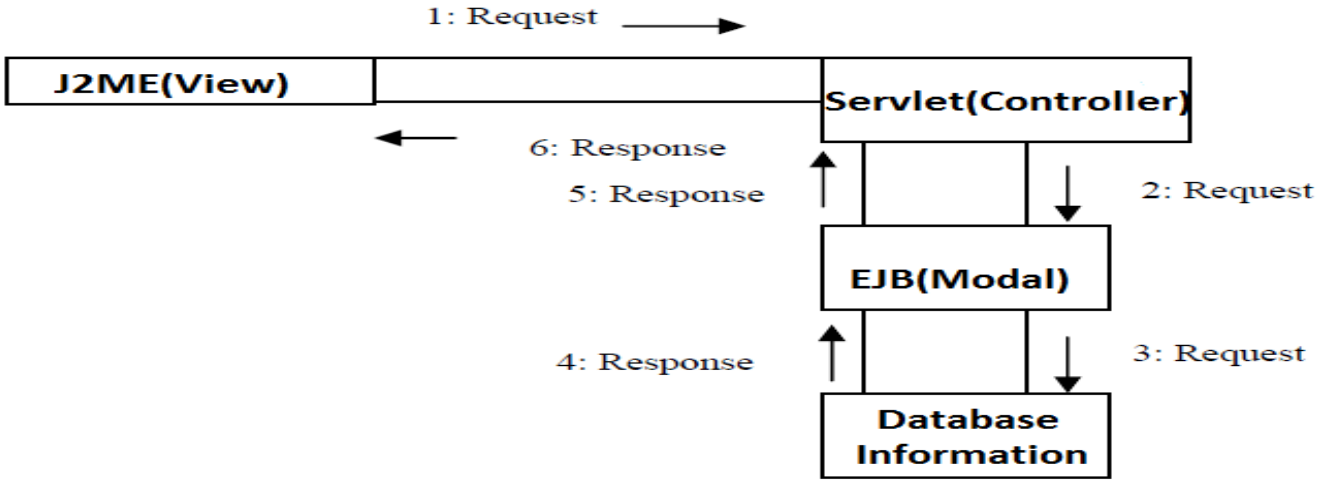
Use case diagram for a cell phonebook

Sequence Diagram→ It is a type of interaction diagram. It emphasizes the time ordering of messages. It has a global life line and focus of the control. An object life line is the vertical dashed line that represents existence of an object over a period of time.



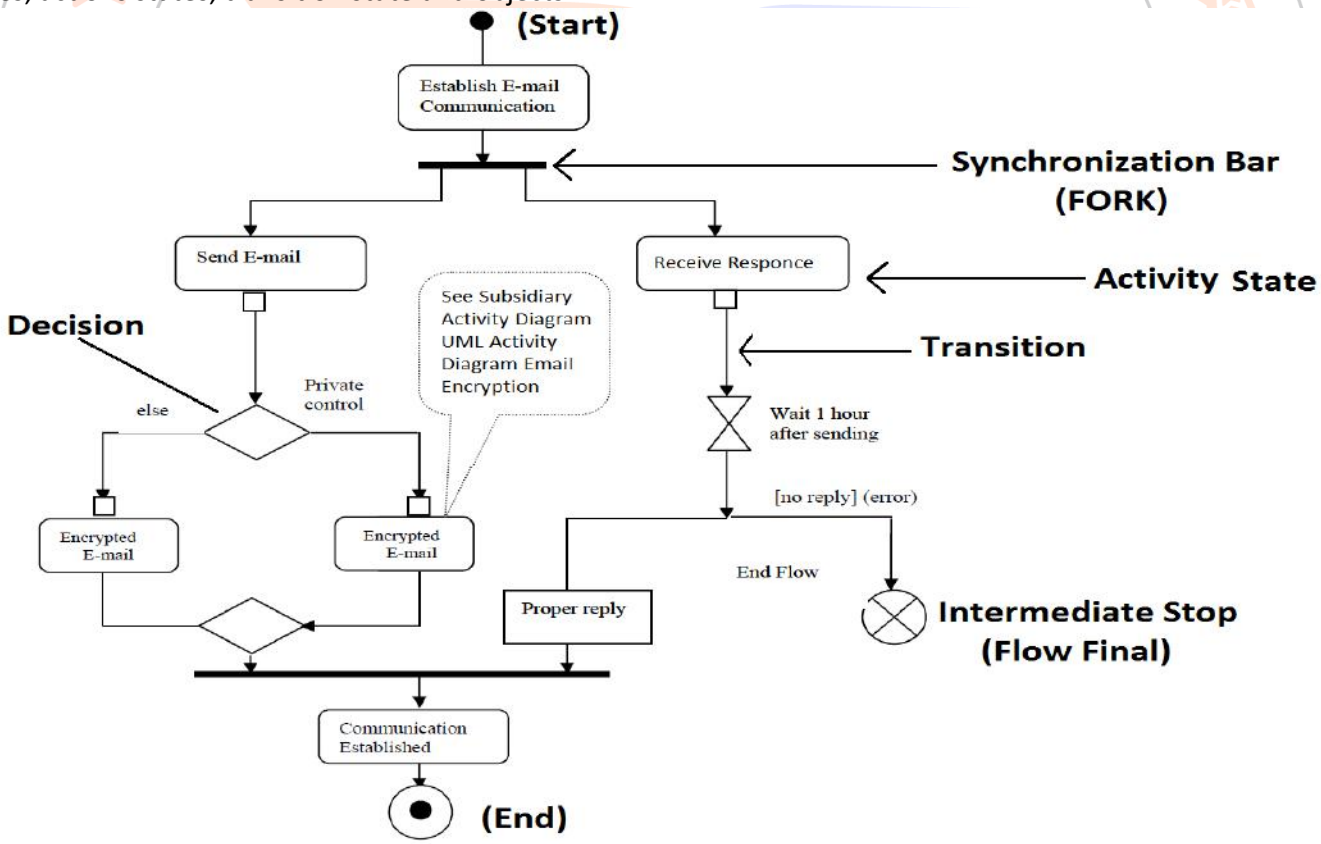
Sequence diagram for sending document

Collaboration Diagram→ Collaboration diagrams are interaction diagrams that emphasize the structural organisation of an object that send and receive messages. There is always a path in collaboration diagrams to indicate how one object is linked to another, and sequence numbers to indicate the time ordering of a message.



Collaboration diagram for execution using J2ME, Servlet and EJB

Activity Diagram→ An activity diagram shows the flow from one activity to another. Activity diagram is used for business process modelling. An activity diagram is a simple & intuitive illustration of what happens in a workflow, what activities can be done in parallel & whether there are alternative paths through the workflow. It shows the flow from one activity to another. It contains activity states, actions states, transition state and objects.



Activity diagram for E-mail encryption