Internal Assesment Test – III, Septemeber 2022									
Sub:	SOFTWARE ENGINEERING							Code:	20MCA24
Date:	22-9-2022	Duration:	90 mins	Max Marks:	50	Sem:	II	Branch:	MCA

Answer **ONE FULL QUESTION** from each part	Marks	OBE	
		CO	RBT

Part-I

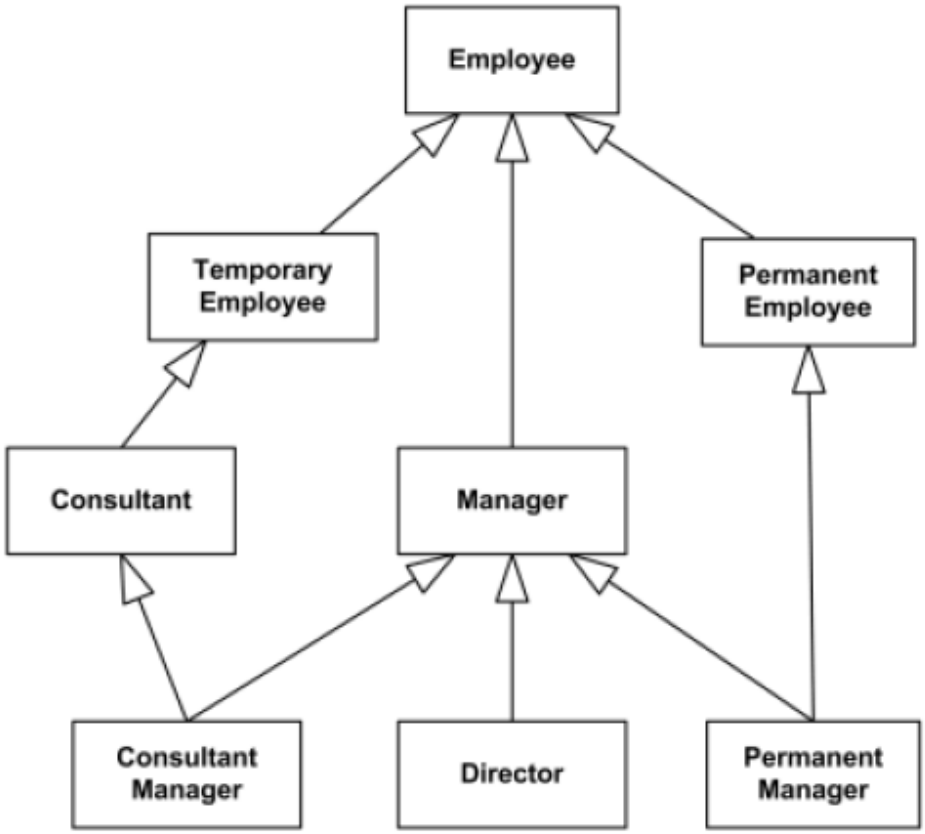
1 (a)	What is Object Orientation? Explain the stages involved in Object oriented Methodology The term object-oriented (OO) means that we organize software as a collection of discrete objects that incorporate both data structure and behaviour. Characteristics of object oriented Approach o Identity: o Classification o Inheritance o Polymorphism a. Identity: - Each object has its own inherent identity. Two objects are distinct even if all their attribute values (such as name and size) are identical. - Each object has a unique handle by which it can be referenced. Such as an address, array index, or artificial number. - Example Variable name Address aCredit 10000007 aDebit 20044502 anAcccount 45672347 b. Classification: - Objects with the same data structure (attributes) and behaviour(operations) are grouped into a class. Each object is said to be an instance of its class. c. c. Inheritance: - Inheritance is the sharing of attributes and operations (features) among classes based on a hierarchical relationship. d. d. Polymorphism: - Means that the same operation may behave differently for different classes. 1.1.2. What is OO development? ---------- ----------	[6]	CO2	L2
(b)	Explain the three kinds of models used in Object Oriented Design method The three models to describe a system from different viewpoints are: o Class model o State model o Interaction model Class model - Describes The structure of objects in a system o their identity, o their relationships to other objects, o their attributes, and o their operations. – Class diagrams express the class model. The state model - Describes - The behavior of the objects of a given class. The state model describes the states and events in a system for one class of objects. - The State diagrams express state model. - State diagram shows states like : Active, Idle, Out of Service, Current, Visible The interaction model - Describes how the objects in a system cooperate to achieve broader results. The interaction model consists of sequence, activity and use case diagrams.	[4]	CO2	L3

(OR)

2	Define the following terms with example i) Multiplicity ii) Enumerators iii) Links and Association iv) Aggregation v) Abstract Class **Links** - Link is a connection between two object instances. Link is an instance of an association. - Links are represented by object diagram. Example: Several number of persons WorksFor the Wipro Company. Kumar : Person name = "Kumar" John : Person name = "John" WorksFor Wipro : Company name = "Wipro" Kavitha : Person name = "Kavitha" Figure 3.1: Object diagram with Links The UML notation for a link is a line between objects; a line may consist of several line segments. If the link has a name, it is underlined. **Association** - Association is a connection between two or more classes. Association describes a group of links with common structure and semantics. All the links in an association connect objects from same classes. It is basically bidirectional. - Association is represented by class diagram. - The UML notation for an association connects related classes and is also denoted by a line (with possibly multiple line segments). **Multiplicity** - It specifies the number of instances of one class is related to a single instance of an associated class. - Multiplicity can be applied to Links or attributes. Multiplicity for association lines: UML diagrams specifies the multiplicity with an interval such as "1" (exactly one), "1..*"(one or more), "3.5" (three to five), "*" (zero or more) shortened to many. **Association classes**. Given a link of an association class, you can find the constituent objects. Alternatively, given a constituent object, you can find the multiple links of an association class. **Enumeration:** An enumeration is a data type that has a finite set of values. Figure 4.1 illustrates. Figure.pen-Type is an enumeration that includes solid, dashed, and dotted. TwoDimensional.fillType is an enumeration that includes solid, grey, none, horizontal lines, and vertical lines.	[10]	CO2	L2

Part-II

3	What is Multiple Inheritance? Explain different kind of inheritance with examples. Multiple inheritance is a feature of some object-oriented computer programming languages in which an object or class can inherit characteristics and features from more than one parent object or parent class.	[10]	CO2	L2



4.6.1 Kinds of Multiple Inheritance

The most common form of multiple inheritance is from sets of disjoint classes. Each subclass inherits from one class in each set. In Figure 4.15 *FullTimeIndividualContributor* is both *FullTimeEmployee* and *IndividualContributor* and combines their features. *FullTimeEmployee* and *PartTimeEmployee* are disjoint; each employee must belong to exactly one of these. Similarly, *Manager* and *IndividualContributor* are also disjoint and each employee must be one or the other. The model does not show it, but we could define three additional combinations: *FullTimeManager*, *PartTimeIndividualContributor*, and *PartTimeManager*. The appropriate combinations depend on the needs of an application.

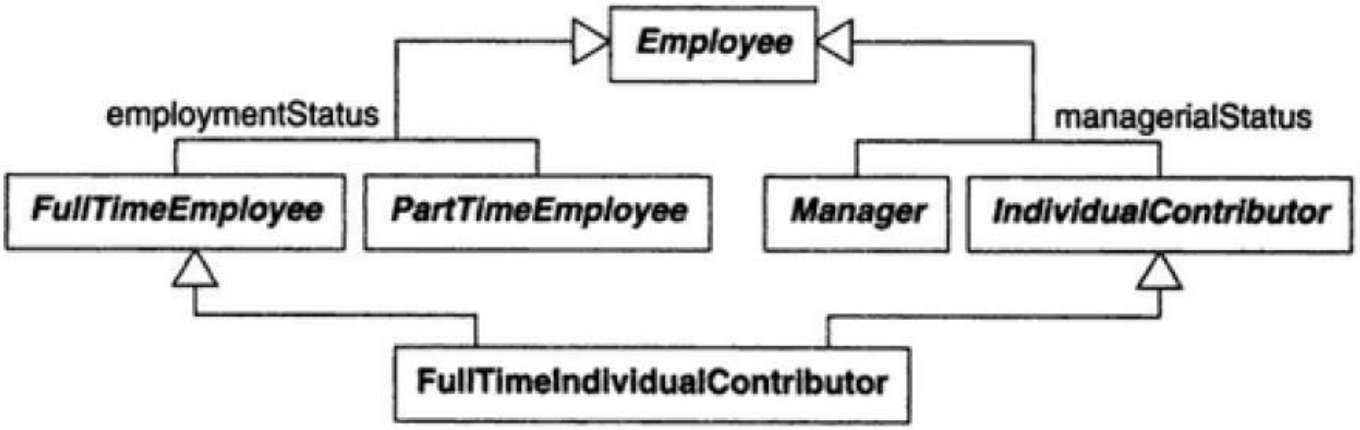


Figure 4.15 Multiple inheritance from disjoint classes. This is the most common form of multiple inheritance.

Conflicts among parallel definitions create ambiguities that implementations must resolve. In practice, you should avoid such conflicts in models or explicitly resolve them, even if a particular language provides a priority rule for resolving conflicts. For example, suppose that *FullTimeEmployee* and *IndividualContributor* both have an attribute called *name*. *FullTimeEmployee.name* could refer to the person's full name while *IndividualContributor.name* might refer to the person's title. In principle, there is no obvious way to resolve such clashes. The best solution is to try to avoid them by restating the attributes as *FullTimeEmployee.personName* and *IndividualContributor.title*.

Multiple inheritance can also occur with overlapping classes. In Figure 4.16, *AmphibiousVehicle* is both *LandVehicle* and *WaterVehicle*. *LandVehicle* and *WaterVehicle* overlap, because some vehicles travel on both land and water. The UML uses a constraint (see Section 4.9) to indicate an overlapping generalization set; the notation is a dotted line cutting across the affected generalizations with keywords in braces. In this example, *overlapping* means that an individual vehicle may belong to more than one of the subclasses. *Incomplete* means that all possible subclasses of vehicle have not been explicitly named.

Each generalization should cover a single aspect. You should use multiple generalizations if a class can be refined on several distinct and independent aspects. In Figure 4.15, class *Employee* independently specializes on employment status and managerial status. Consequently the model has two separate generalization sets.

A subclass inherits a feature from the same ancestor class found along more than one path only once; it is the same feature. For example, in Figure 4.15 *FullTimeIndividualContributor* inherits *Employee* features along two paths, via *employmentStatus* and *managerialStatus*. However, each *FullTimeIndividualContributor* has only a single copy of *Employee* features.

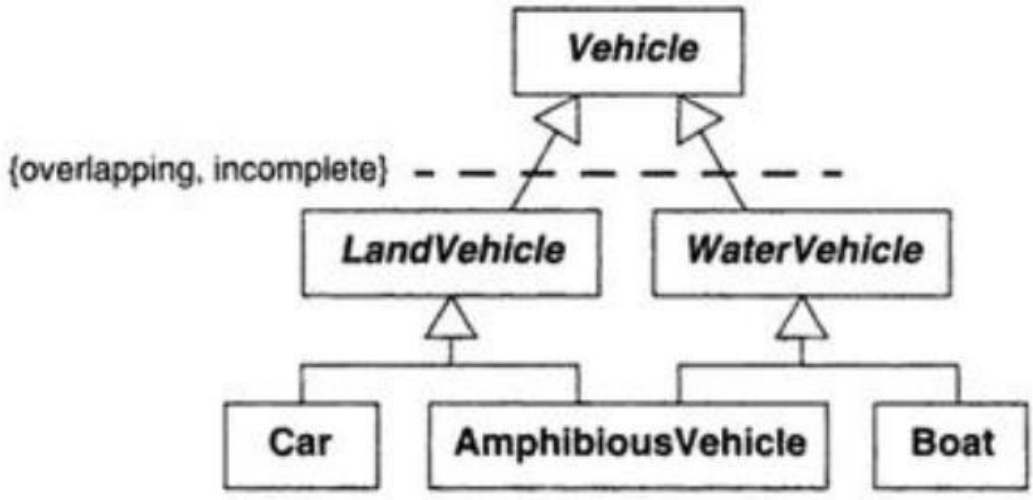
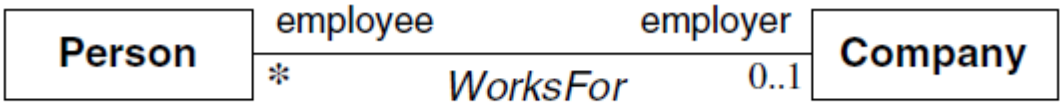


Figure 4.16 Multiple inheritance from overlapping classes. This form of multiple inheritance occurs less often than with disjoint classes.

4	Mention two situations where association end names are useful? Describe in details	[10]	CO3	L2

Association End Names :

Association end names often appear as nouns in problem descriptions. As Figure belowshows, a name appears next to the association end. In the figure *Person* and *Company* participate in association *WorksFor*. A person is an *employee* with respect to a company; a company is an *employer* with respect to a person. Use of association end names is optional, but it is often easier and less confusing to assign association end names instead of, or in addition to, association names.



employee	employer
Joe Doe	Simplex
Mary Brown	Simplex
Jean Smith	United Widgets

Association end names are especially convenient for traversing associations, because you can treat each one as a pseudo attribute. Each end of a binary association refers to an object or set of objects associated with a source object. From the point of view of the source object, traversal of the association is an operation that yields related objects. Association end names provide a means of traversing an association, without explicitly mentioning the association.

Ordering: Often the objects on a "many" association end have no explicit order, and you can regard them as a set. Sometimes, however, the objects have an explicit order. For example, Figure 3.15 shows a workstation screen containing a number of overlapping windows. Each window on a screen occurs at most once. The windows have an explicit order, so only the topmost window is visible at any point on the screen. The ordering is an inherent part of the association. You can indicate an ordered set of objects by writing "{ordered}" next to the appropriate association end.

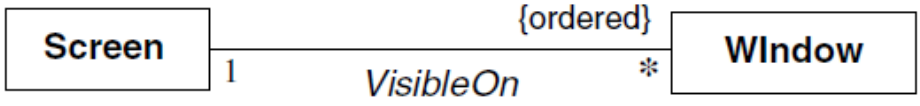


Figure 3.15 Ordering the objects for an association end. Ordering sometimes occurs for "many" multiplicity.

Bags and Sequence: A ***bag*** is a collection of elements with duplicates allowed. A ***sequence*** is an ordered collection of elements with duplicates allowed. In Figure 3.16 an itinerary is a sequence of airports and the same airport can be visited more than once. Like the *{ordered}* indication,*{bag}* and *{sequence}* are permitted only for binary associations.
Note that the *{ordered}* and the *{sequence}* annotations are the same, except that the first disallows duplicates and the other allows them. A sequence association is an ordered bag, while an ordered association is an ordered

set.

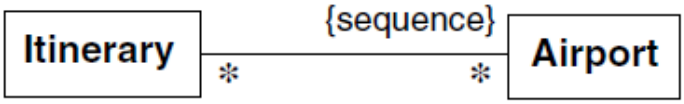


Figure 3.16 An example of a sequence. An itinerary may visit multiple airports, so you should use *{sequence}* and not *{ordered}*.

(h) Association Classes: An ***association class*** is an association that is also a class. Like the links of an association, the instances of an association class derive identity from instances of the constituent classes. Like a class, an association class can have attributes and operations and participate in associations. You can find association classes by looking for adverbs in a problem
statement or by abstracting known values.
In Figure 3.17, *accessPermission* is an attribute of *AccessibleBy*. The sample data at the bottom of the figure shows the value for each link. The UML notation for an association class is a box (a class box) attached to the association by a dashed line.
In

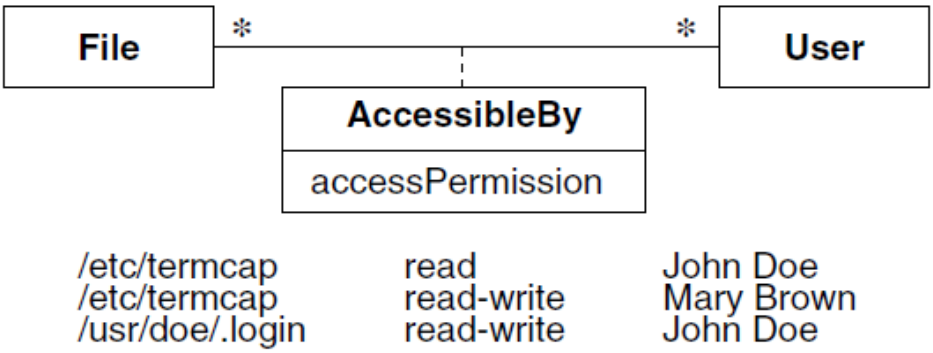


Figure 3.17 An association class. The links of an association can have attributes.

5	Explain effort estimation Techniques and Describe in detail about Constructive Cost Model for effort estimation of planning a software project development Cocomo (Constructive Cost Model) is a regression model based on LOC, i.e **number of Lines of Code**. It is a procedural cost estimate model for software projects and is often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time, and quality. It was proposed by Barry Boehm in 1981 and is based on the study of 63 projects, which makes it one of the best-documented models. The key parameters which define the quality of any software products, which are also an outcome of the Cocomo are primarily Effort & Schedule: • **Effort:** Amount of labor that will be required to complete a task. It is measured in person-months units. • **Schedule:** Simply means the amount of time required for the completion of the job, which is, of course, proportional to the effort put in. It is measured in the units of time such as weeks, months. Different models of Cocomo have been proposed to predict the cost estimation at different levels, based on the amount of accuracy and	[10]	CO3	L2

correctness required. All of these models can be applied to a variety of projects, whose characteristics determine the value of constant to be used in subsequent calculations. These characteristics pertaining to different system types are mentioned below. Boehm's definition of organic, semidetached, and embedded systems:

1. **Organic –** A software project is said to be an organic type if the team size required is adequately small, the problem is well understood and has been solved in the past and also the team members have a nominal experience regarding the problem.
2. **Semi-detached –** A software project is said to be a Semi-detached type if the vital characteristics such as team size, experience, knowledge of the various programming environment lie in between that of organic and Embedded. The projects classified as Semi-Detached are comparatively less familiar and difficult to develop compared to the organic ones and require more experience and better guidance and creativity. Eg: Compilers or different Embedded Systems can be considered of Semi-Detached type.
3. **Embedded –** A software project requiring the highest level of complexity, creativity, and experience requirement fall under this category. Such software requires a larger team size than the other two models and also the developers need to be sufficiently experienced and creative to develop such complex models.
    1. Basic COCOMO Model
    2. Intermediate COCOMO Model
    3. Detailed COCOMO Model
4. **Basic Model –**

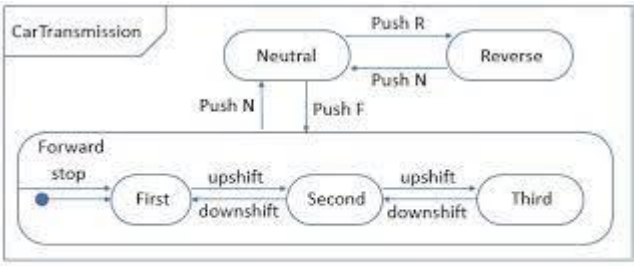
1. The above formula is used for the cost estimation of for the basic COCOMO model, and also is used in the subsequent models. The constant values a,b,c and d for the Basic Model for the different categories of system:

Software Projects	a	b	c	d
Organic	2.4	1.05	2.5	0.38
Semi Detached	3.0	1.12	2.5	0.35
Embedded	3.6	1.20	2.5	0.32

(OR)

6 (a)	What is an Event? Explain nested state and nested state diagrams with example. Events are **some occurrences that can trigger state transition of an object or a group of objects**. Events have a location in time and space but	[5]	CO3	L2

do not have a time period associated with it. Events are generally associated with some actions.

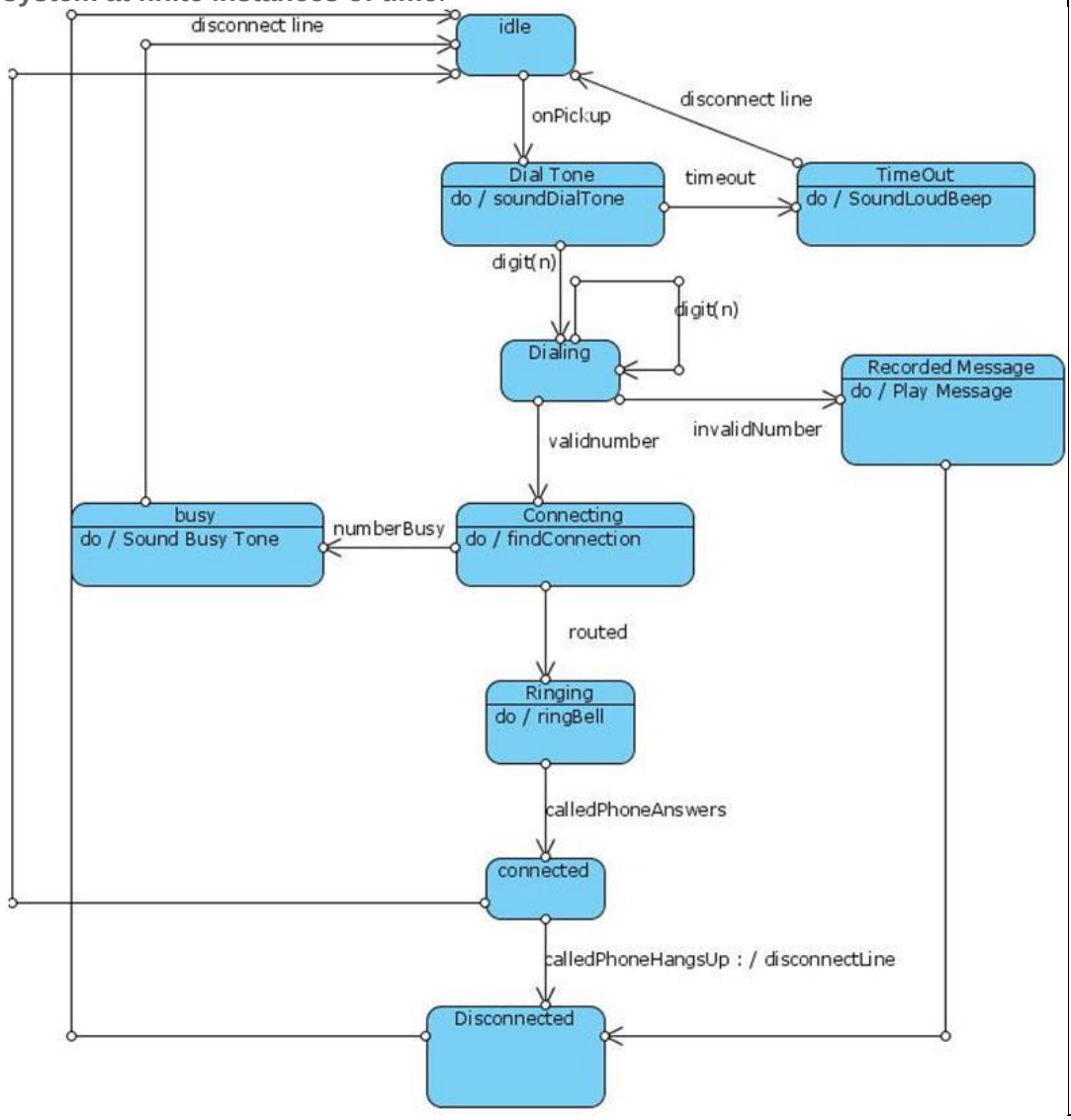


Nested State Diagram of Car Transmission

A state can contain other states, often called as nested states or substates. If you are modeling complex state machines, use nested states to separate detailed behavior into multiple levels. States can also contain actions that identify the tasks that can occur when an object is in a particular state.

(b) Define state diagram and draw the state diagram for telephone line with activities.

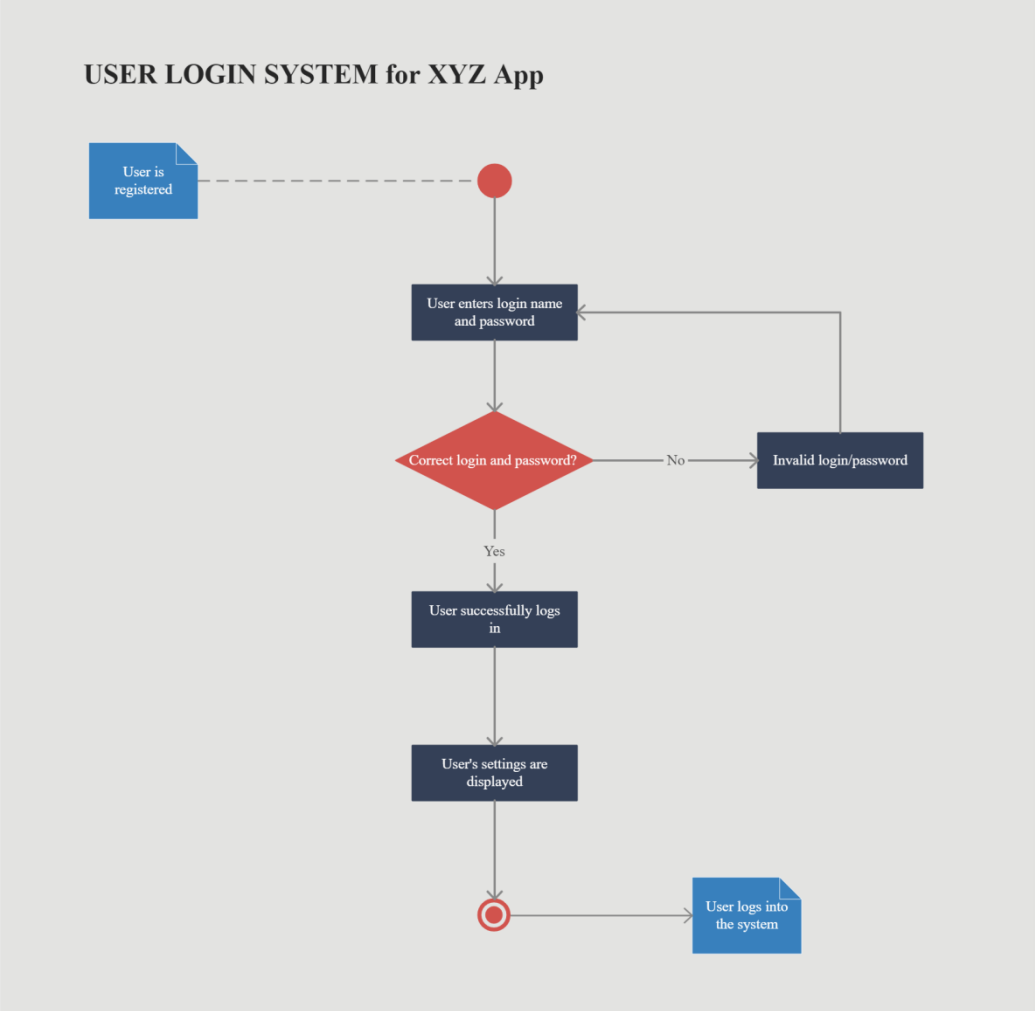
A state diagram is **used to represent the condition of the system or part of the system at finite instances of time**.



[5] CO3 L3

Part-IV

7	Mentioned guideline for activity models and explain with an suitable example. Activity diagrams can be used to model business requirements, create a high-level view of a system's functionalities, analyze use cases and for various other purposes. In each of these cases, here's how to draw an activity diagram from the beginning. **Step 1: Figure out the action steps from the use case** Here you need to identify the various activities and actions your business process or system is made up of. **Step 2: Identify the actors who are involved** If you already have figured out who the actors are, then it's easier to discern each action they are responsible for. **Step 3: Find a flow among the activities** Figure out in which order the actions are processed. Mark down the conditions that have to be met in order to carry out certain processes, which actions occur at the same time and whether you need to add any branches in the diagram. And do you have to complete some actions before you can proceed to others? **Step 4: Add swimlanes** You have already figured out who is responsible for each action. Now it's time to assign them a swimlane and group each action they are responsible for under them.	[10]	CO2	L2



(OR)

8 How to make Project Scheduling and Staffing and demonstrate with real time project

[10] CO4 L2

When you begin planning and drafting your project schedule, you want to include all project activities. At first, the project management scheduling process may feel a bit foreign to you—that's normal! By following these steps, you can get more comfortable creating your own process for project schedule development, and use it every time you plan a project.

Define your project goals. Write down key milestones or deliverables that will make this project successful in the end.

Identify all stakeholders. Make a list of every person that needs to interact with the project team, even if their role is a simple sign-off.

Determine your final deadline. Decide when you need to be completely finished with the project. Be sure to give yourself enough time to account for conflicts or changes that might come up later during schedule management.

List each step or task.Take those milestones and deliverables you defined in the first step and break them down into smaller tasks and subtasks to be sure all bases are covered.

Assign a team member responsible for each task. Decide who will take on each task and subtask, and be transparent with deadlines. Remember that your colleagues likely have other projects going on at the same time. Be mindful of their workload so they don't feel overloaded.

Work backward to set due dates for each task. Figure out how long each task will take to complete (its start and end date), knowing that delays are inevitable. Sequencing is important to consider as well since certain tasks will need to be finished before another can start.

You've successfully built your project plan and now it's important to organize it in a way that everyone involved can see and work from it. Finding a tool that can help you do both will be critical to your success.

Part-V

9	What is risk management? Briefly explain the risk management activities?	[6]	CO1	L1

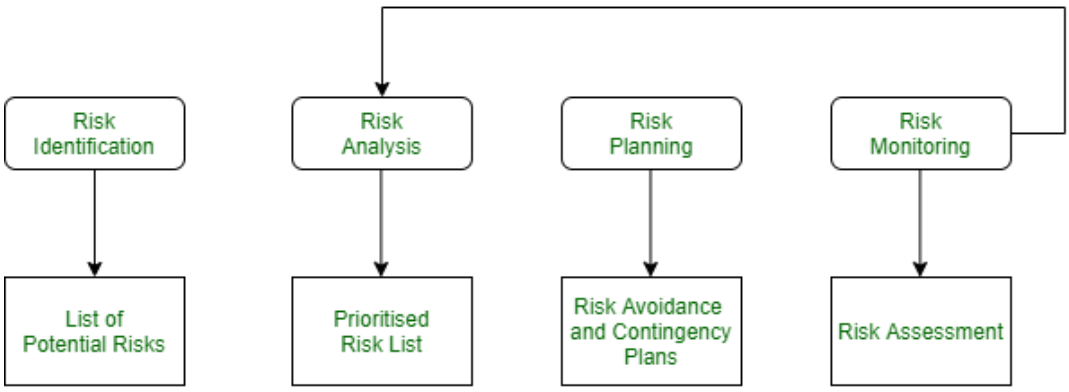
Risk management is the area which tries to ensure that the impact of risks on cost, quality and schedule is minimized. The main purpose of risk management is to identify and manage the risks associated with a software project and solve the problem. Estimating the risks that can affect the project schedule or the quality of the software being developed and taking action to avoid the risk is the important task of a project manager. Identifying and preparing plans to reduce their impact on the project is called risk management. The basic motivation of risk management is to avoid disaster or heavy losses. The risk can be categorised as follows.

1. **Project Risks :** These are the risks which affect the project schedule or resources.
2. **Product Risks :** These are the risks which affect the quality or performance of the being developed.
3. **Business Risks :** These are the risks which affect the organization developing or procuring the software.

This classification is not a special classification. If an experienced programmer leaves a project then it is a project risk because the delivery of the system may be delayed, the product may be a risk because the replacement may not be a seasoned one and therefore may be mistakes and business. Risk management is very important for software projects due to the inherent uncertainties that most

[4] CO2 L2

projects face. The process of risk management is shown in fig.



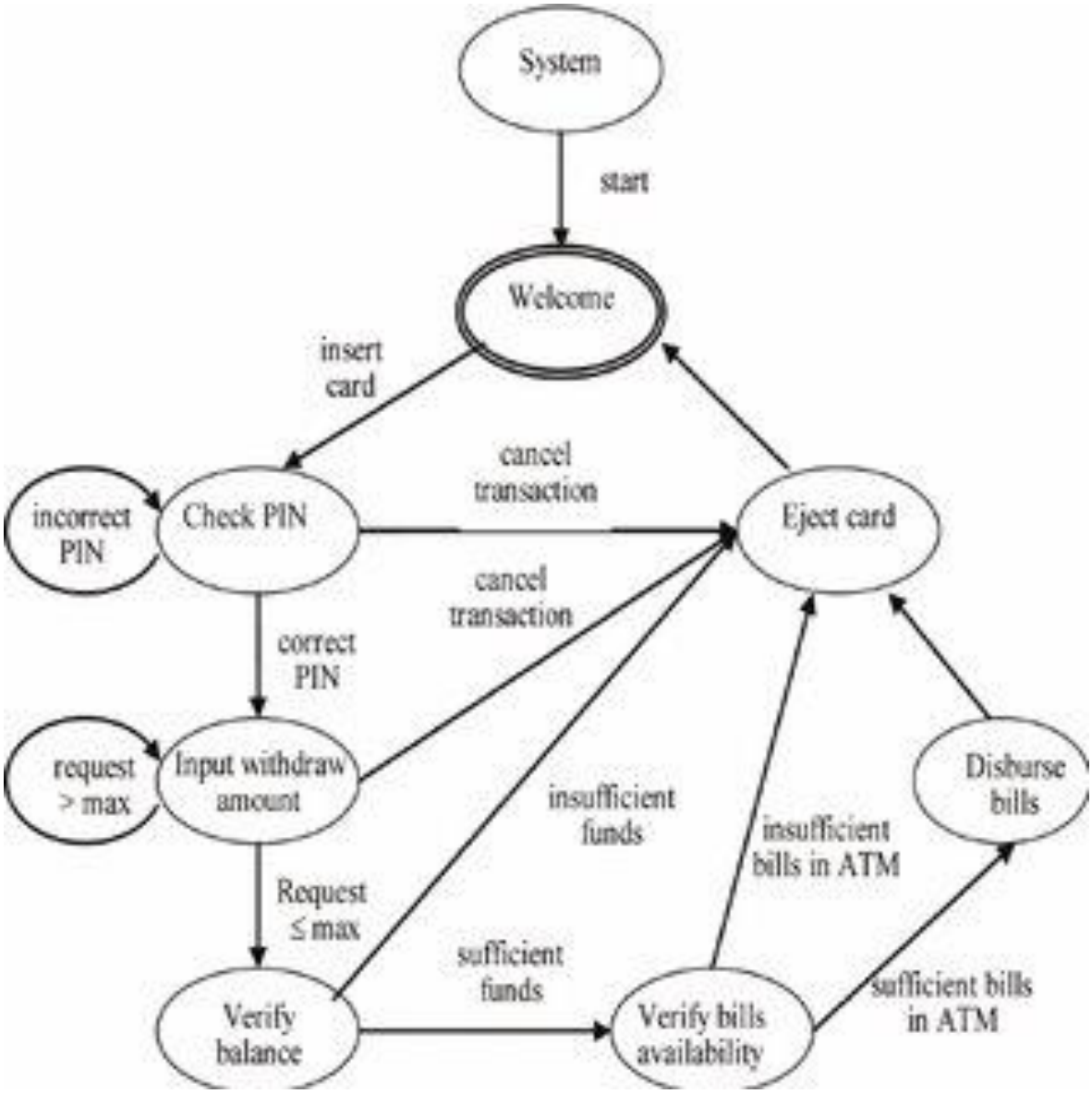
The Risk Management process

process of risk management involves several stages are as follows-

1. **Risk Identification :** In this stage, the possible project, product and business risks are identified.
2. **Risk Analysis :** In this stage or process, the likelihood and consequences of these risks assessed.
3. **Risk Planning :** In this stage, risk avoidance in either planned to affect the plan or mitigate its effects on the project.
4. **Risk Monitoring :** In this stage, risk assessment is done continuously and the risk reduction plan is revised as more information about risk is available.

Like all other project planning, the risk management process is an iterative process that continues throughout the project. Risk management process results should be documented in a risk management plan. This should include a discussion of the risks that the project faces, analyzing these risks and requiring plans to manage these risks.It may also include some results of the risk management. The risk management has to deal with identifying the undesirable events that may occur, the likelihood of them occurring and the losses that occur when undesirable events occur. knowing this, strategies can be devised to reduce the possibility of reducing the risk or impact of the content. Therefore, risk management revolves around risk assessment and risk control. These are top 10 item techniques for managing the

10	Design a system for ATM and list the function-oriented design components and explain.	10	CO5	L6


System
start
Welcome
insert
card
cancel
transaction
incorrect
PIN
Check PIN
Eject card
cancel
transaction
correct
PIN
request
> max
Input withdraw
amount
insufficient
funds
insufficient
bills in ATM
Disburse
bills
Request
≤ max
sufficient
funds
sufficient bills
in ATM
Verify
balance
Verify bills
availability