**Question bank**

**SOFTWARE ENGINEERING**

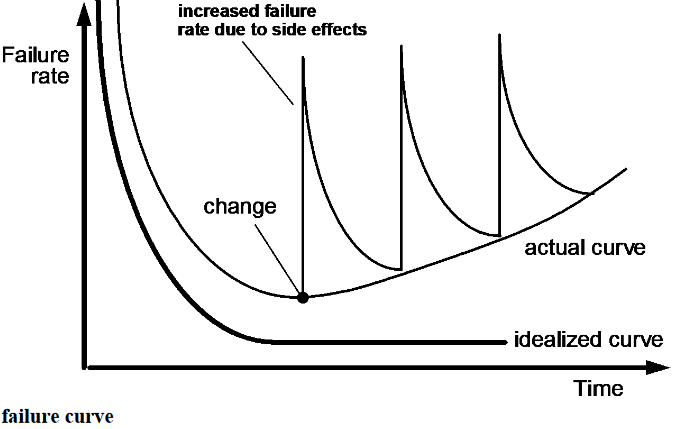
**Chapter 1**

**1.Define Software Engineering.**

ANS: Application of Systematic, disciplined, quantifiable, approach to the development, operation, and maintenance of software as well as study of these approaches.

**2.Draw failure Curve of Software.**

ANS:



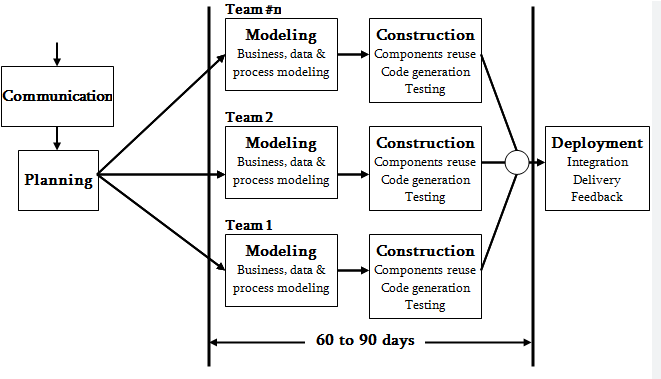
3.Difference between Waterfall model and Incremental model.

ANS:

|  |  |  |
| --- | --- | --- |
| Sr. No. | Waterfall Model | Incremental Model |
|  | There is a high amount of risk in the waterfall model. | There is a low amount of risk in the incremental model. |
|  | There is a long waiting time for running software in the waterfall model. | There is a short waiting time for running software in the incremental model. |
|  | Flexibility to change in the waterfall model is Difficult. | Flexibility to change in incremental model is Easy. |
|  | Testing is done in the waterfall model after the completion of the coding phase. | Testing is done in the incremental model after every iteration of the phase. |
|  | Returning to the previous stage/phase in the waterfall model is not possible. | Returning to the previous stage/phase in the incremental model is possible. |
|  | In the waterfall model overlapping of phases is not possible. | In incremental model overlapping of phases is possible. |
|  | There is only one cycle in the waterfall model. | Multiple development cycles take place in the incremental model. |
|  | The customer is involved only at the beginning of development. | In incremental model, there is a continuous customer involvement . |
|  | In waterfall model the linear framework type is used. | In Incremental model linear with iterative framework type is used. |
|  | Reusability is the least possible. | Reusability is possible to some extent. |

4.Draw Neat diagram of RAD model and Explain it’s all the stages of Operation.

Ans : Diagram of RAD model



**Fig : RAD model**

Rapid application development (RAD) is an incremental software development process model that emphasizes an extremely short development cycle.

The RAD model is a “high-speed” adaptation of the linear sequential model in which rapid development is achieved by using component-based construction.

If requirements are well understood and project scope is constrained, the RAD process enables a development team to create a “fully functional system” within

very short time periods (e.g., 60 to 90 days). Phases in RAD MODEL are as given below:

**Communication :**

Communication between the developer and customer takes place so as to explain the need statement and specifications. Requirements of customer are understood clearly before starting the work.

**Planning :**

Work is planned in the terms of resource requirements , process and tasks requirements

,etc.

Reuse of components and processes and schedule is planned in this phase.

1. **Business modelling** : Some questions like where this product will have business? Why is it being prepared? Who will use it? What information is generated? Who processes it? Where does the information go ; are answered .
2. **Data modeling**. The information flow defined as part of the business modeling phase is refined into a set of data objects that are needed to support the business. The characteristics (called attributes) of each object are identified and the relationships between these objects defined.
3. **Process modeling**. The data objects defined in the data modeling phase are transformed to achieve the information flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.
4. **Application generation.** RAD assumes the use of fourth generation techniques. Rather than creating software using conventional third generation programming languages the RAD process works to reuse existing program components (when possible) or create reusable components (when necessary). In all cases, automated tools are used to facilitate construction of the software.
5. **Testing and turnover**. Since the RAD process emphasizes reuse, many of the program components have already been tested. This reduces overall testing time. However, new components must be tested and all interfaces must be fully exercised.
6. **Deployment :** the ready software is handed over to the customer. Customer feedback, evaluation and support are also the parts of this phase.

# Advantages of RAD Model

* The process of application development and delivery are fast.
* This model is flexible, if any changes are required.
* Reviews are taken from the clients at the staring of the development hence there are lesser chances to miss the requirements.

# Disadvantages of RAD Model

* The feedback from the user is required at every development phase.
* This model is not a good choice for long term and large projects.

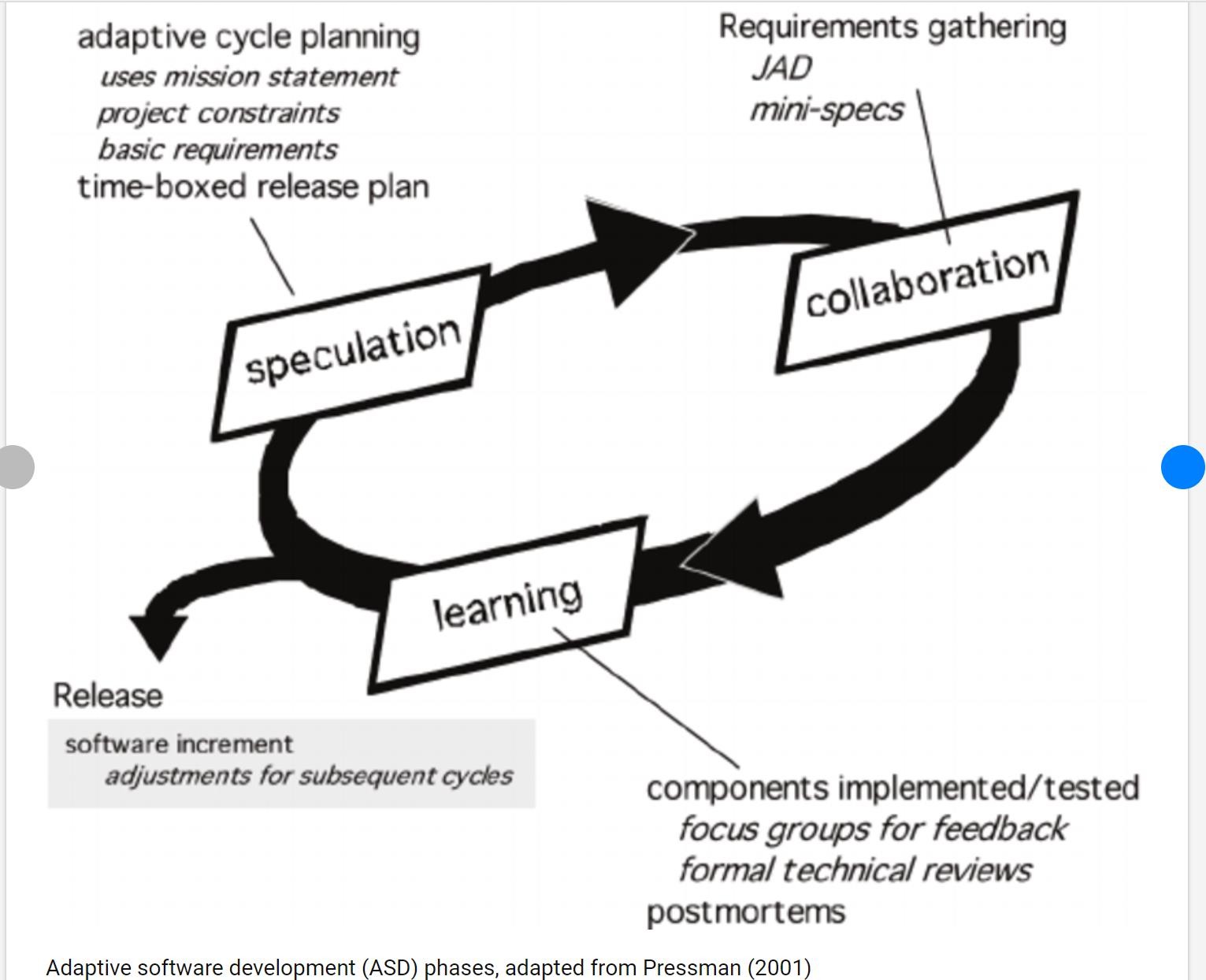
5. Draw Neat diagram of Adaptive software development model

and Explain it’s all the stages of Operation.

Ans :

**Adaptive Software Development (ASD)**

**Adaptive Software Development** is a method to build complex software and system. ASD focuses on human collaboration and self-organisation.



ASD **“life cycle”** incorporates three phases / stages namely:

1. Speculation
2. Collaboration
3. Learning

These are explained as following below.

# Speculation:

During this phase project is initiated and planning is conducted. The project plan uses project initiation information like project requirements, user needs, customer mission statement etc, to define set of release cycles that the project wants.

# Collaboration:

It is the difficult part of ASD as it needs the workers to be motivated. It collaborates communication and teamwork but emphasizes individualism as individual creativity plays a major role in creative thinking. People working together must trust each others to

* + Criticize without animosity,
  + Assist without resentment,
  + Work as hard as possible,
  + Possession of skill set,
  + Communicate problems to find effective solution.

# Learning:

The workers may have a overestimate of their own understanding of the technology which may not lead to the desired result. Learning helps the workers to increase their level of understanding over the project.

Learning process is of 3 ways:

1. Focus groups
2. Technical reviews
3. Project postmortem

ASD’s overall emphasis on the dynamics of self-organizing teams, interpersonal collaboration, and individual and team learning yield software project teams that have a much higher likelihood of success.

6.Difference between Perspective and agile process model.

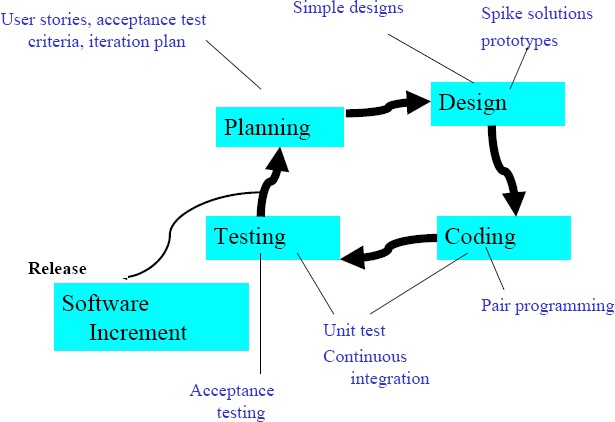
Ans : Differences between Perspective and agile process model are as follows :

|  |  |  |
| --- | --- | --- |
| Sr. no | Prescriptive model | Agile model |
| 1 | It describes “how to do” according to a specific software process system. | It describes the way of quick development processes which are light in weight and very easy. |
| 2 | It has predefined and ordered activities. | It promotes practical and quick approach rather than predefined  processes. |
| 3 | These are more rule based. | These are more principles based. |
| 4 | The generic framework activities are:  Communication Planning Modeling  Construction And deployment | The activities similar like prescriptive approach are used but not in the perfect manners. They are used iteratively. |
| 5 | It facilitates planning, improves  the quality by very formal methods. | It facilitates the informal methods of  development of quick, easy, adaptive processes. |
| 6 | Involvement of customer is very less | Customer is completely involved in the development of processes. |

7.Explain Extreme programming model with the help of neat diagram.

Ans :

# Extreme Programming (XP)



**Fig : Extreme Programming**

Extreme Programming (XP) is the most widely used Agile Process model. XP uses an object oriented approach as its development paradigm. XP encompasses a set of rules and practices that occur within the context of four framework activities : planning, design , coding and testing.

# Planning

Begins with a set of stories (scenarios).

* Each story written by the customer is assigned a value depending on its priority.
* The members of the XP team assess each story and assigned a cost measured in development weeks.
* If a story has more that three weeks to develop the customer is asked to split it.
* New stories can add any time.
* Customers and XP team work together to decide how to group stories for next increment.
* AS development work proceeds, the customers can add stories, split stories and eliminate them.
* The XP team then reconsiders all remaining releases and modify its plan accordingly.

# Design

A simple design is preferred.

* Design only consider the given stories.
* Extra functionality discouraged.
* Identify the object oriented classes that are relavant to the current system.
* The output of the design process is a set of CRC ( Class Responsibility Collaborator) cards.

# Coding

XP recommends developing a series of unit tests for each of the story.

* Once the code is complete, units should be unit tested.
* Pair programming – two people work together at one computer.

# Testing

The unit tests that has been created in the coding stage should be implemented using a framework that can be implemented.

* This enables regression testing.
* Integration and validation can occur on a daily basis.
* This provides the XP team with a continual indication of the progress and also raise flags early if things are going wrong.
* Acceptance tests are derived from user stories that have been implemented as parts of the software release.

8.Define:-Software .List any two characteristics of software

Ans :

**Software is :**

* + - * A set of instructions to acquire the inputs and manipulate them to produce the desired output in terms of functions and performance.
      * Data structures that enable the programs to adequately manipulate information
      * Includes set of documents that will describe its operation and use. Eg. System manual, installation and implementation manual etc.

Characteristics of Software are:

* + - 1. **Software is developed or engineered but it is not manufactured in the classical sense:** -Although some similarities exist between software development and hardware manufacture, the two activities are fundamentally different.

-In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce quality problems that are nonexistent (or easily corrected) for software.

-Both activities are dependent on people, but the relationship between people applied and work accomplished is entirely different.

-Both activities require the construction of a "product" but the approaches are different.

-Software costs are concentrated in engineering. This means that software projects cannot be managed as if they were manufacturing projects.

# Software doesn't wear out

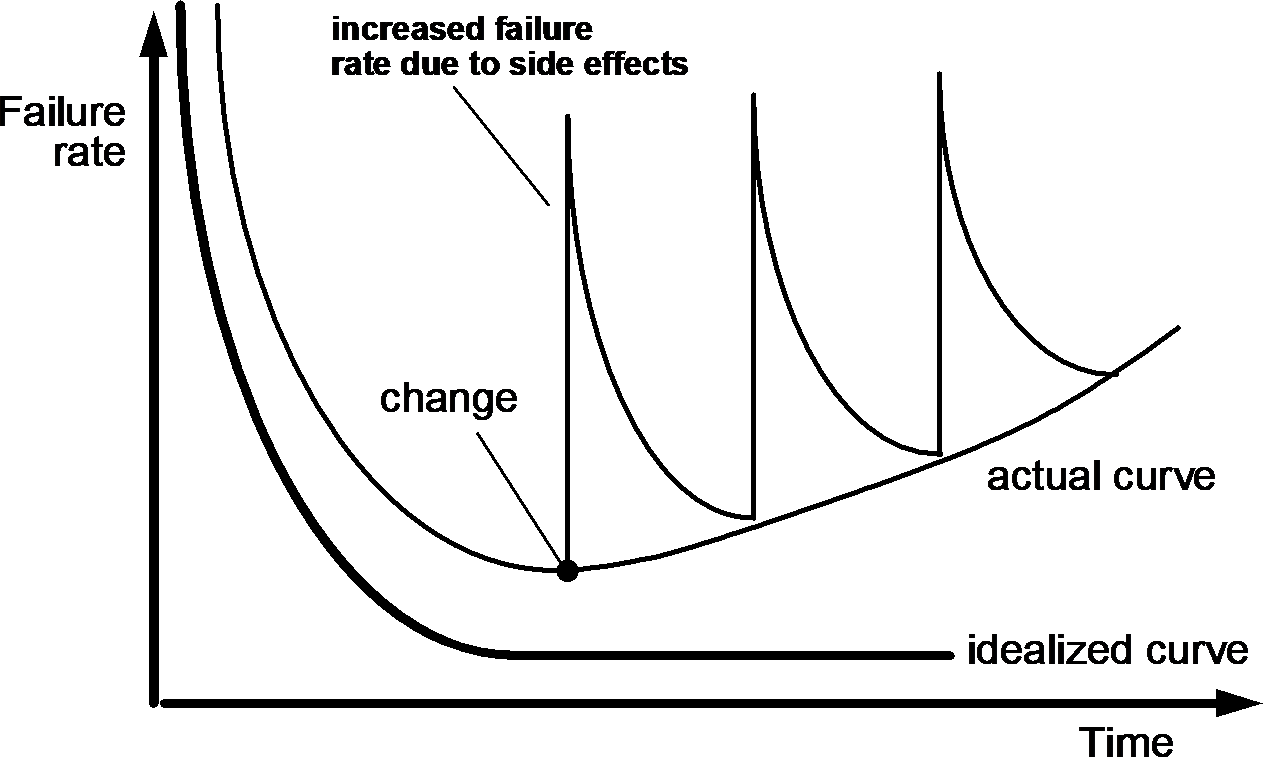
- Hardware bathtub curve compared to the software ascending spiked curve.

Although the industry is moving toward component-based construction, most software continues to be custom built (it is still complex to build).

- Software is not susceptible to the environmental maladies that cause hardware to wear out.

-In theory, therefore, the failure rate curve for software should take the form of the "idealized curve" .

-Undiscovered defects will cause high failure rates early in the life of a program. However, these are corrected (ideally, without introducing other errors) and the curve flattens as shown.

The idealized curve is a gross over-simplification of actual failure models for software. However, the implication is clear, software doesn’t wear out. But it does deteriorate.

# Fig 1.1 failure curve

* + - 1. **Most of the softwares are custom built :**

As an engineering discipline evolves, a collection of standard design components is created. Standard screws and off-the-shelf integrated circuits are only two of thou-sands of standard components that are used by mechanical and electrical engineers as they design new systems. The reusable components have been created so that the engineer can concentrate on the truly innovative elements of a

design, that is, the A software component should be designed and implemented so that it can be reused in many different programs.

These subroutine libraries reused well-defined algorithms in an effective manner but had a limited domain of application. Today, we have extended our view of reuse to encompass not only algorithms but also data structure. Modern reusable components encapsulate both data and the processing applied to the data, enabling the software engineer to create new

applications from reusable parts. For example, today's graphical user interfaces are built using reusable components that enable the creation of graphics windows, pull-down menus, and a wide variety of interaction mechanisms. The data structure and processing detail required to build the interface are contained with a library of reusable components for interface construction.

9. Draw Neat diagram of SPIRAL model and Explain it’s all the stages of Operation.

ANS :

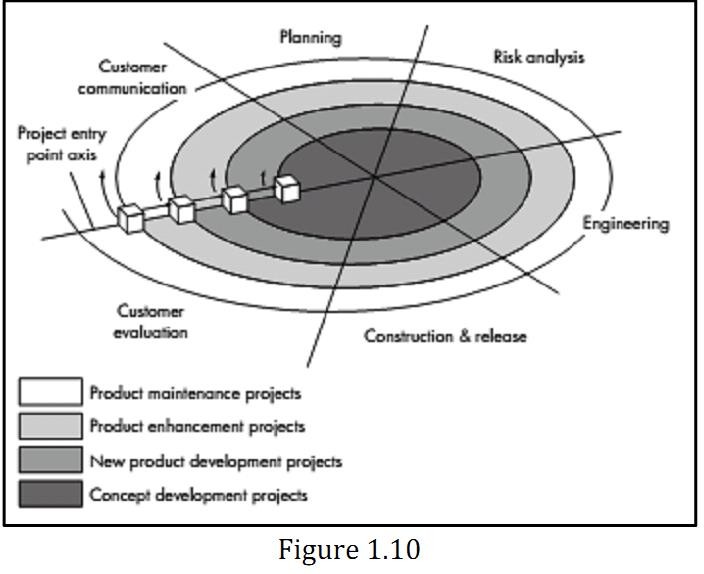
Spiral model:

A spiral model is divided into a number of framework activities, also called task regions.

Typically, there are between three and six task regions. Figure 1.10 depicts a spiral model

that contains six task regions:

1. Customer communication—tasks required to establish effective communication between developer and customer are carried out in this phase . Specifications of the product are understood by the developer from the customer clearly. In this model there is a continuous communication with customer.
2. Planning—tasks required to define resources, timelines, and other project related Information are carried out. Estimation of cost is done in this phase
3. Risk analysis—tasks required to assess both technical and management risks are done. The risks are identified and the steps to be taken if the risks occur are found.
4. Engineering—tasks required to build one or more representations of the Application are carried out. Model of the software is prepared. Approach of designing and development is finalized in this step.
5. Construction and release—tasks required to construct (coding of processes) , test, install, and provide user support (e.g., documentation and training). Testing is carried out at the end of each process development .
6. Customer Evaluation : The customer uses/evaluates the software installed on his site and gives feedback about its operation/work to the developer. If any changes are needed in the software, those will be communicated to the team of developers by the customer.



1. Advantages:
   1. Risk reduction mechanisms are in place
   2. Supports iteration and reflects real-world practices
   3. Systematic approach
2. Disadvantages:
   1. Requires expertise in risk evaluation and reduction
   2. Complex, relatively difficult to follow strictly
   3. Applicable only to large systems
3. Applicability:
   1. Internal development of large systems

UNIT 2.

1.Define:-Requirement .State any two tasks of requirement engineering.

Ans: Requirement: A condition that must be possessed by a system to

satisfy a contract specification, standard or other formally imposed document.

Tasks of requirement engineering :

Requirement engineering consists of seven different tasks as follow:

1. Inception

2. Elicitation

3. Elaboration

4. Negotiation

5. Specification

6. Validation

7. Requirement management

1. Inception : Inception is a task where the requirement engineering asks a set of questions to establish a software process.

2. Elicitation : Elicitation means to find the requirements from anybody.

The requirements are difficult because the following problems occur

in elicitation.

3. Elaboration : In this task, the information taken from user during

inception and elaboration and are expanded and refined in

elaboration.

· Its main task is developing pure model of software using

functions, feature and constraints of a software.

4. Negotiation : In negotiation task, a software engineer decides the

how will the project be achieved with limited business resources.

5. Specification : In this task, the requirement engineer constructs a final

work product.

· The work product is in the form of software requirement

specification.

6. Validation : The work product is built as an output of the requirement

engineering and that is accessed for the quality through a validation

step.

7. Requirement management : It is a set of activities that help the project team to identify, control and track the requirements and changes

can be made to the requirements at any time of the ongoing

project.

2.Define :-Requirement engineering and state any two characteristics of good requirement.

Ans : The procedure to collect the s/w requirements from customer, analyze and document them is called as Requirement engineering.

Characteristics of Effective/good Requirements

● Complete

1. fully describe the functionality to be delivered.

2. must contain all the information necessary for the developer to

design and implement

● Correct

1. accurately describe the functionality to be built.

2. software requirement that conflicts with its parent system

requirement is not correct.

● Feasible

1. must be possible to implement each requirement within the known

capabilities and limitations of the system and its operating

environment.

● Unambiguous

All readers of a requirement statement should arrive at a

single, consistent interpretation of it, but natural language

is highly prone to ambiguity.

● Consistent

Consistent software requirements don’t conflict with other

requirements of the same type or with higher-level

business, system, or user requirements.

Characteristics of Effective/good Requirements

● Traceable

A traceable requirement can be linked backwards to its origin

and forward to the design elements and source code that

implement it and to the test cases that verify the implementation

as correct.

● Testable

A testable requirement is one which can check whether the

requirement can be checked against the feature.

3.Explain Planning Principles of software engineering

Ans : Planning Principles of software engineering

1. Understand the scope of the project

2. Involve the customer in the planning activity

3. Recognize that planning is iterative

4. Estimate based on what you know

5. Consider the risk as you define the plan

6. Be realistic

7. Adjust granularity as you define the plan

8. Define how you intend to ensure quality (EULA)

9. Describe how you intend to accommodate change

10.Always keep track of the plan and make changes as required.

Explanation :

1)      **Understand the scope of project**: – if team unaware about destination then it is impossible to use roadmap. Scope provides the software team with destination as scope represent the range within which development take place.

2)       **Involve the customer in the planning activity**: – the customer defines priority and establish project constraints. While direct interacting with software team ,customer can negotiate on order of delivery ,timeline and other project related issue so customer involvement is important

3)      **Recognize that planning is iterative**: – project plan is suppose to change or the personal reason so plan must be adjust to accommodate these changes. Incremental model dictate replanting that is  after delivery of each software increment to user feedback is receive and change are made.

4)      **Estimate based on what you know**: – Team works effectively .if efforts, cost and task estimates are reliable so estimation provides reliability and conveniences to software team if the information is reliable.

5)      **Consider risk as you define the plan**:-Risk is event when cause happens some unwanted outcomes because risk may have high impact and high probability on project which will destroy it. so proper management of risk is a required

6)      **Be realistic**: – project should be a realistic .all the practical aspect has to be considered .plan should be dynamic enough to include all reason like delay, mistake, and wrong decision lagging due to technical reason due to personal reason of team.

7)      **Adjust granularity as you define the plan** :- Granularity means level of detail for particular specification .some activity are repeated and require longer time required detail plan hence granularity must be adjust in the project plan. So that is why Adjust granularity as you defines the plan.

8)      **Define how you are going to achieve quality**: – the most effective mechanism applied during project development to assure the quality. Formal technical review is meeting conducted by technical staff to find out defects in any stage of development and quality is improved by removing defects that is why it mention very clear to achieve quality.

9)      **Describe how you aim to accommodate change**: – Good quality software must be well planned to accommodate changes that are demanded by customer in any stage of software development process. Changes in project affect the cost and the schedule of the project so it is important to think about change. So that why Describe how you aim to accommodate change.

10)   **Always keep track of the plan and make changes as required**: – track the progress of project development on daily basis all team members should participate in planning activity so that they will accept to recover the lagging schedule.

4. Explain Communication Principles of software engineering

Ans : Communication Principles of software engineering :

1. Listen carefully

i. To collect lots of data from the client, the developer team has to listen

carefully.

ii. Maximum information with respect to requirement and the specifications

should be collected before the implementation and the designing of the

software.

2. Prepare before you communicate

i. A proper agenda or the guidelines for the meetings should be prepared

before the start of the meeting.

ii. Complete detail and the description about the clients and their work area should be gathered to deliver the software up to the best expectation.

1. Have a facilitator for any communication meeting

i. The requirement gathering and the specification are important for any

software development, hence the communication should continue till the

requirement gathering is over.

4. Face-to-face communication is best

i. It is always better to sit across the table and have discussion on the

requirement on the software development by the client and the developer.

ii. Distant communication does not help gathering data properly.

5. Take notes and document decisions

i. The important points discussed should also be recorded.

ii. Proper notes and the documentation is important for the successful

completion and deployment of the project.

6. Strive for collaboration

i. Collaboration in terms of teamwork is required for the successful

completion of the software.

ii. The collective knowledge of the team members should be implemented in the development.

7. Stay focused and modularize your discussion

i. As the development is the working of many team members, so the

possibility of the discussion going from one topic to the other topic is quite possible.

ii. As a good software developer it is required that the discussion remains focused on the specified area.

8. Draw a picture if something is unclear

i. Drawing flowcharts, E-R diagrams and other supporting graphical representations give clarity to the discussion and the documentation.

9. Move on once you agree, move on when you can't agree, move on if

something unclear can't be clarified at the moment

CHAPTER 2 FOR COMPUTER ENGINEERING BY VAISHALI RANE Page 4

i. Healthy discussion leads to the final conclusion of successful implementation of the

software.

ii. Once reached to final statement recorded should move to the next step.

iii. If no conclusion is reached than that point should be left and move ahead with new implementation which is cost effective.

10. Negotiation is not a contest or game

i. Negotiation should be mutual not to put someone down or make them

feel to be the loser.

5. Explain Deployment Principles of software engineering.

Ans : Deployment Principles of software engineering :

i. Manage customer’s expectations or requirements

• In most cases customer wants more than he/she has stated earlier as his requirements or expectations.

• In many cases customer is disappointed, even after getting all his/her requirements satisfied.

• Hence, at the time of software delivery, developer must have skills to manage the customer’s expectations and requirements.

ii. Record-keeping mechanism must be established for customer support

• ‘Customer Support’ is essential and important factor in deployment phase.

• The support should be well planned and with proper record keeping mechanism.

iii. Provide essential instructions, documentations and manual

• Actual software project delivery includes all documentations, help files and guidance for handling the software by user.

iv. Assemble and test complete delivery package

• The customer side must get all supporting and essential help from developer’s side.

• For this reason CD with complete assembled and tested delivery package with the following support should be delivered:

(i) Necessary supported operational features and help.

(ii) Essential manuals required for software troubleshooting.

(iii) All executable file of developed software.

(iv) Necessary installation procedures.

v. Do not deliver any defective or buggy software to the customer.

• The software should be tested before deployment to the customer.

The specification should be with the requirement to the customer.

6.List any four Core Principles of software engineering.

Ans:

Core Principles of software engineering are:

1. The reason it all exists. Provide value to the user

i. The software system exists to provide value for the user.

ii. Before specifying the problem the requirement and the specifications have

to be laid down.

iii. The hardware and the software platform to be decided for implementation.

2. Keep it simple stupid

i. The terms and the design used for development of the project should be

kept simple and easily understandable.

ii. All the terms used should be easy to facilitate the basic concept of the

project.

3. Maintain the vision

i. A clear vision is important for the development of a software.

ii. Compromising the architectural vision of the project weakens the

development of the software.

iii. The developer should hold the vision and ensure the successful

development and deployment of the software.

4. What you reproduce, someone else will have to consume. (implement

knowing someone else will have to understand what you are doing)

i. Always specify, design and implement knowing that someone else is going to understand what is being developed.

ii. Customers for the product development is very large.

iii. Design the data structure and the implementation keeping

implementation in mind and the end user.

iv. Code with the concern that the product has to be implemented and

maintained by the end user.

5. Be open to the future

i. The system designed today should be adaptable to the development and

changes in the future at a low cost.

ii. There should not be much changes to the software to adopt to the new

changes in the future development.

6. Plan ahead for reuse

i. The design and specifications should be developed in such a way that they can be reused for other implementations.

ii. The code and the design should be well documented for the use in future.

7. Think!

i. Before designing and implementation a proper thought should be to the

end result.

ii. Proper data structure and the design and implementation strategy should be developed if the software needs modification in the future.

7.What are the four stages of essence of practice

Ans : The practice involves problem solving, modeling, designing, code generation,

testing and quality assurance listed below in four steps.

i. Understand the problem (Communication and analysis)

a) Who are the stake holders of solution in the problem?

b) What are the unknowns?

c) Can the problems be compartmentalized?

d) Can the problem be represented graphically?

ii. Plan a solution (modeling and software design)

a) Have you seen similar problems before?

b) Has a similar problem been solved?

c) Can sub problems be defined?

d) Can you represent a solution in a manner that leads to an effective implementation?

iii. Carry out the plan (code generation)

a) Does the solution conform?

b) Is each component part of the solution probably correct?

iv. Examine the result for accuracy (testing and quality assurance)

a) Is it possible to test each component part of the solution?

b) Does the solution produce results that conform to the data, function,

features, and behavior that are required?

8. List Barry Boehn’s Seven W5HH Principles of the software development.

ANS : Software modeling follows the Barry Boehm’s seven W5HH

principles for the software development.

i. Why is the system being developed?

ii. What will be done?

iii. When will it accomplished?

iv. Who is responsible for the function?

v. Where they are organizationally located?

vi. How will the job be done technically and managerially?

Vii. How many resources will be required?

UNIT 3

1.Write neat diagram explain the translation of analysis model into design model. OR

2. Write neat diagram explain the translation of requirement model into design model.

Ans : Translating Requirement model into design model

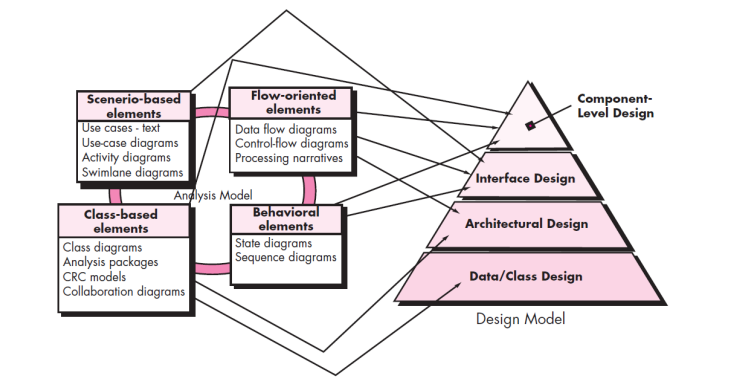
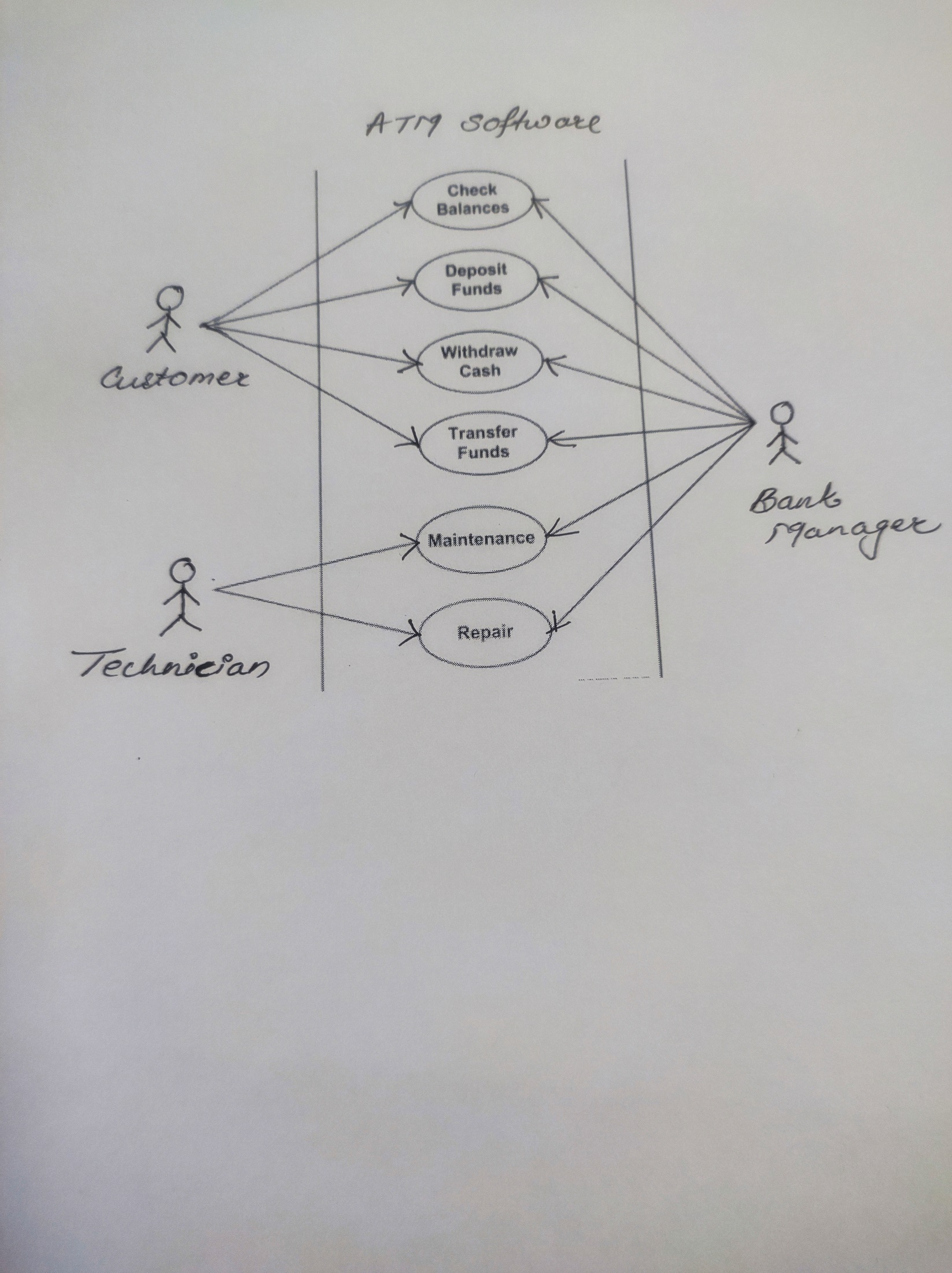


Fig 1. Translation of analysis model/requirement model into design model

* Each of the elements of the requirements model provides information that is necessary to create the four design models required for a complete specification of design.
* The flow of information during software design is illustrated in Figure 1.1.
* The requirements model, demonstrated by scenario-based, class-based, flow-oriented, and behavioral elements, feed the design task.
* Using design notation and design methods, design produces a data/class design, an architectural design, an interface design, and a component design. The data/class design transforms class models into design class realizations and the requisite data structures required to implement the software. The objects and relationships defined in the CRC diagram and the detailed data content depicted by class attributes and other notation provide the basis for the data design action.
* Part of class design may occur in conjunction with the design of software architecture.
* More detailed class design occurs as each software component is designed.
* The architectural design defines the relationship between major structural elements of the software, the architectural styles and design patterns that can be used to achieve the requirements defined for the system, and the constraints that affect the way in which architecture can be implemented.
* The architectural design representation—the framework of a computer-based system—is derived from the requirements model.
* The interface design describes how the software communicates with systems that interoperate with it, and with humans who use it. An interface implies a flow of information (e.g., data and/or control) and a specific type of behavior.
* Therefore, usage scenarios and behavioral models provide much of the information required for interface design.
* The component-level design transforms structural elements of the software architecture into a procedural description of software components. Information obtained from the class-based models, flow models, and behavioral models serve as the basis for component design.
* The importance of software design can be stated with a single word—quality. Design is the place where quality is fostered in software engineering.
* Design provides with representations of software that can be assessed for quality.
* Design is the only way that you can accurately translate stakeholder‘s requirements into a finished software product or system. Software design serves as the foundation for all the software engineering and software support activities that follow. Without design, there is a risk building an unstable system which may cause failure if system or difficult to test.
* The **architectural design** defines the relationship between more structural elements of the software, the architectural styles and design patterns that can be used to achieve the requirements defined for the system, and the constraints that affect the way in which the
* architectural design can be implemented. The **architectural design** can be derived from the System Specs, the analysis model, and interaction of subsystems defined within the analysis model.
* The **interface design** describes how the software communicates with systems that interpolate with it, and with humans who use it.  An interface implies a flow of information (data, and or control) and a specific type of behaviour.
* The **component-level design** transforms structural elements of the software architecture into a procedural description of software components.
* **The Design Model Data Design** –Transforms information domain model into data structures required to implement software
* **Architectural Design** –Defines relationship among the major structural elements of a program
* **The Design Model Interface Design** –Describes how the software communicates with itself, to systems that interact with it and with humans.
* **Procedural Design –Also Component-Level** –Transforms structural elements of the architecture into a procedural description of software construction.

3.Draw Use Case Diagram for atm(automated teller machine)with four use cases and 2 actors.

Ans : Use Case Diagram for atm(automated teller machine)



4. Draw Use Case Diagram for library management with four use cases and 2 actors

ANS :