1 Introduction to Software Engineering, Software Process Models

1.1 : Software Engineering

Q.1 Define a software engineering.

☞[SPPU : May-11, 14, Marks 2]

Ans. : Software engineering is a discipline in which theories, methods and tools are applied to develop professional software product.

1.2 : Nature of Software

Q.2 What are various categories of software ?

☞[SPPU : May-14, Marks 8]

Ans. : Software can be classified in following categories :

- **System software** - Typical programs in this category are compiler, editors, and assemblers. The purpose of the system software is to establish a communication with the hardware.
- **Application software** - It consists of standalone programs that are developed for specific business need.
- **Engineering/scientific software** - This software category has a wide range of programs from astronomy to volcanology, from automative stress analysis to space shuttle orbital dynamics.
- **Embedded software** - This category consists of program that can reside within a product or system.
- **Web applications** - Web application software consists of various web pages that can be retrieved by a browser.
- **Artificial intelligence software** - This kind of software is based on knowledge based expert systems. Typically, this software is useful in robotics, expert systems, image and voice recognition.

1.3 : Software Engineering Principles

Q.3 Write and explain the software engineering principles.

Ans. : 1. **Reason for Existence :** The intention of software system is to provide value to its users.

2. **Keep It Simple Stupid(KISS) :** Software design must be simple to implement and the resultant software system must be more maintainable and less error-prone.
3. **Vision :** The goal and objective of the software must be defined.
4. **The produce is getting consumed :** While developing the software system, the documentation must be maintained so that the maintenances of the system becomes easy.
5. **Be Open to future :** If some changes need to be incorporated in the system, then those must be accommodated easily.
6. **Plan for Reuse :** The software system components can be made reusable so that the time and efforts can be saved. The programming techniques like Object oriented programming can be used for it.
7. **Think :** Before performing every software engineering activity think and understand it.

1.4 : The Software Process

Q.4 Define software process.

Ans. : Software process can be defined as the structured set of activities that are required to develop the software system. The fundamental activities are -

1. Specification
2. Design and Implementation
3. Validation
4. Evolution

1.5 : Software Myths

Q.5 What are different software myths that the customers and practitioners believe in and what are their corresponding realities ?

☞ [SPPU : April-15 (In-Sem), Marks 4]

Ans. : • **Myth :** Using a collection of standards and procedures one can build software. (Management Myth)
Reality : Eventhough we have all standards and procedures with us for helping the developer to build software, it is not possible for software professionals to build desired product. This is because - the collection which we have should be complete, it should reflect modern techniques and more importantly it should be adaptable.

- **Myth :** Add more people to meet deadline of the project. (Management Myth)
  Reality : Adding more people in order to catch the schedule will cause the reverse effect on the software project i.e. software project will get delayed. Because, we have to spend more time on educating people or informing them about the project.
- **Myth :** If a project is outsourced to a third party then all the worries of software building are over. (Management Myth)
  Reality : Sometimes, the outsourced projects require proper support for development.
- **Myth :** Even if the software requirements are changing continuously it is possible to accommodate these changes in the software. (Customer Myth)
  Reality : There are chances of introducing more and more errors in the software. Similarly, the additional resources and more design modifications may be required.
- **Myth :** We can start writing the program by using general problem statements only. Later on using problem description we can add up the required functionalities in the program. (Customer Myth)
  Reality : It is not possible each time to have comprehensive problem statement.
- **Myth :** Once the program is running then its over! (Practitioner's Myth)
  Reality : Even though we obtain that the program is running major part of work is after delivering it to customer.
- **Myth :** Working program is the only work product for the software project. (Practitioner's Myth)
  Reality : Other elements need to be present in the software project such as documentation of software, guideline for software support.
- **Myth :** There is no need of documenting the software project; it unnecessarily slows down the development process. (Practitioner's Myth)

Reality : Documenting the software project helps in establishing ease in use of software.

Q.6 Briefly explain the myths and reality associated with practitioner and management in software project.

☞[SPPU : April-17 (In-Sem), Marks 5]

Ans. : Practitioner's Myths and Reality - Refer Q.5.

- **Myth :** Using a collection of standards and procedures one can build software. (Management Myth)
  Reality : Eventhough we have all standards and procedures with us for helping the developer to build software, it is not possible for software professionals to build desired product. This is because - the collection which we have should be complete, it should reflect modern techniques and more importantly it should be adaptable. It should also help the software professional to bring quality in the product.
- **Myth :** Add more people to meet deadline of the project. (Management Myth)
  Reality : Adding more people in order to catch the schedule will cause the reverse effect on the software project i.e. software project will get delayed. Because, we have to spend more time on educating people or informing them about the project.
- **Myth :** If a project is outsourced to a third party then all the worries of software building are over. (Management Myth)
  Reality : When a company needs to outsource the project then it simply indicates that the company does not know how to manage the projects. Sometimes, the outsourced projects require proper support for development.

1.6 : Generic Process Model

Q.7 Explain the layered approach in software engineering.

☞ [SPPU : May-13, Marks 6]

Ans. : • Various layers on which the technology is based are **quality focus layer, process layer, methods layer, tools layer.**

- A disciplined **quality management** is a backbone of software engineering technology.

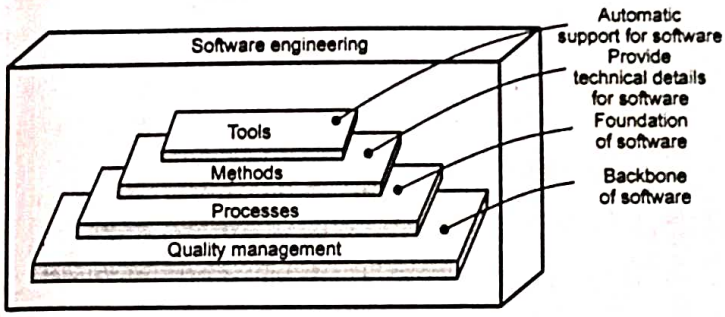


Fig. Q.7.1 Software engineering : A layered approach

- **Process layer** is a foundation of software engineering. Basically, process defines the framework for timely delivery of software.
- In **method layer** the actual method of implementation is carried out with the help of requirement analysis, designing, coding using desired programming constructs and testing.
- Software **tools** are used to bring automation in software development process.

Q.8 What is software process framework ? Explain in detail.

☞ [SPPU : May-15, End Sem Marks 7]

Ans. : The process framework is required for representing the common process activities.

As shown in Fig. Q.8.1, the software process is characterized by process framework activities, task sets and umbrella activities.

Process framework activities

- Communication
  - By communicating customer requirement gathering is done.
- Planning - Establishes engineering work plan, describes technical risks, lists resource requirements, work products produced and defines work schedule.
- Modeling - The software model is prepared by :
  - Analysis of requirements
  - Design

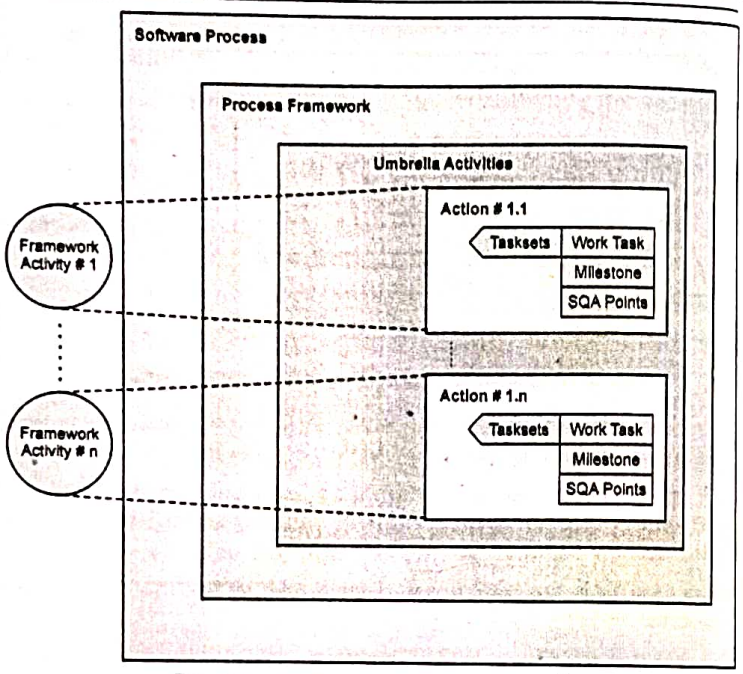


Fig. Q.8.1 Software process framework

- Construction - The software design is mapped into a code by :
  - Code generation
  - Testing
- Deployment - The software delivered for customer evaluation and feedback is obtained.

Task sets - The task set defines the actual work done in order to achieve the software objective. The task set is used to adopt the framework activities and project team requirements using :

- Collection of software engineering work tasks
- Project milestones
- Software quality assurance points

Q.9 What are the various umbrella activities applied throughout a software project ? ☞ **[SPPU : April-15, In Sem, Marks 6]**

Ans. : The umbrella activities occur **throughout the process**. They focus on project management, tracking and control. The umbrella activities are

1. **Software project tracking and control** - This is an activity in which software team can **assess progress** and take corrective action to **maintain schedule.**
2. **Risk management** - The **risks** that may affect project outcomes or quality can be **analyzed.**
3. **Software quality assurance** - These are activities required to **maintain** software **quality.**
4. **Formal technical reviews** - It is required to **assess engineering work products** to uncover and **remove errors** before they propagate to next activity.
5. **Software configuration management** - Managing of configuration process when any **change** in the software occurs.
6. **Work product preparation and production** - The activities to create **models, documents, logs, forms** and **lists** are carried out.
7. **Reusability management** - It defines criteria for **work product reuse.**
8. **Measurement** - In this activity, the process can be defined and collected. Also **project and product measures** are used to assist the software team in delivering the required software.

1.7 : Perspective Process Model

Q.10 What is software process model ?

☞ **[SPPU : May-14, Marks 2]**

Ans. : The process model can be defined as the abstract representation of process. The appropriate process model can be chosen based on abstract representation of process.The software process model is also known as Software Development Life Cycle (SDLC) Model or software paradigm. These models are called prescriptive process models because they are following some rules for correct usage.

Q.11 Give the need of different process models in software development. ☞ **[SPPU : Dec.-13, Marks 4]**

Ans. : Following are some reasons to use the process model in software development

i) The software product development can then be done systematically.

ii) Each team member will understand - what is the next activity and how to do it. Thus process model will bring the definiteness and discipline in overall development process.

iii) Every process model consists of definite entry and exit criteria for each phase.

iv) If the process model is not followed for software development then any team member can perform any software development activity, this will ultimately cause a chaos and software project will definitely fail.

Q.12 Explain the waterfall model with its advantages and disadvantages. ☞ **[SPPU : Dec.-12, Marks 6, May-13, Marks 8]**

Ans. : The waterfall model is also called as 'linear-sequential model' or 'classic life cycle model'.

The software development starts with requirements gathering phase. Then progresses through analysis, design, coding, testing and maintenance. Following figure illustrates waterfall model.

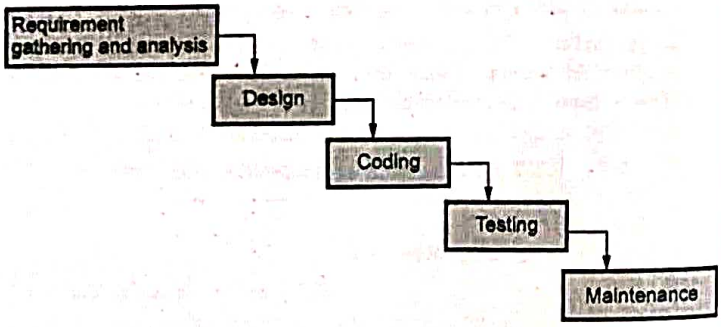


Fig. Q.12.1 Waterfall model

Requirement gathering and analysis phase : In this phase the basic requirements of the system must be understood by software engineer.All these requirements are then well documented and discussed further with the customer, for reviewing.

Design : Design focuses on program attributes such as -

i) Data structure ii) Software architecture

iii) Interface representation iv) Algorithmic details.

Design must be created in such a way that the requirements can be translated into coding easily.

Coding : Coding is a step in which design is translated into machine-readable form. If design is done in sufficient detail then coding can be done effectively. Programs are created in this phase.

Testing : The testing ensures execution of all the paths, functional behaviours. The purpose of testing is to uncover errors, fix the bugs and meet the customer requirements.

Maintenance : When the system is installed and put in practical use then error may get introduced, correcting such errors and putting it in use is the major purpose of maintenance activity.

Advantages

1. The waterfall model is simple to implement.
2. For implementation of small systems waterfall model is useful.

Disadvantages

1. It is difficult to follow the sequential flow in software development process. If some changes are made at some phases then it may cause some confusion.
2. The requirement analysis is done initially and sometimes it is not possible to state all the requirements explicitly in the beginning.
3. The customer can see the working model of the project only at the end. After reviewing of the working model; if the customer gets dissatisfied then it causes serious problems.

Q.13 Explain the incremental process model.

☞ **[SPPU : May-14, Marks 6]**

Ans. : • This model is iterative in nature. The incremental model has following phases.

1. Analysis 2. Design 3. Code 4. Test

- The incremental model delivers series of releases to the customer. These releases are called increments.
- More and more functionality is associated with each increment.
- The first increment is called **core product.** In this release the basic requirements are implemented and then in subsequent increments new requirements are added.

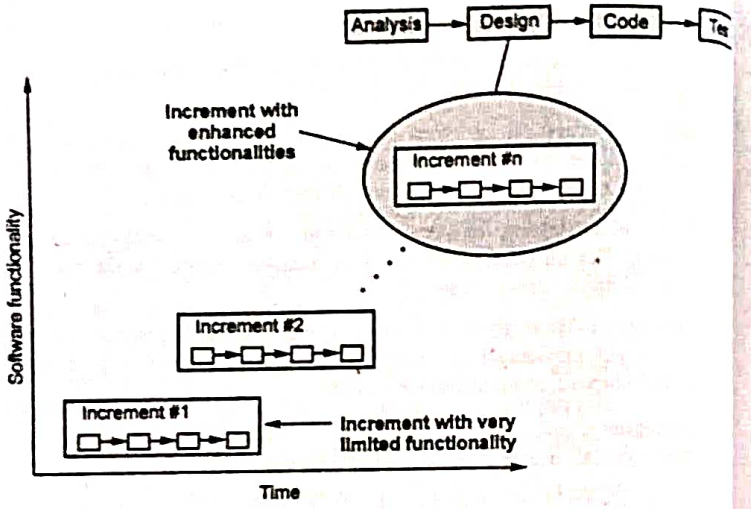


Fig. Q.13.1 The Incremental model

- **Example :** The word processing software package can be considered as an example of incremental model. In the first increment only the document processing facilities are available. In the second increment more sophisticated document producing and processing facilities, file management functionalities are given. In the next increment spelling and grammar checking facilities can be given. Thus in incremental model progressive functionalities are obtained with each release.
- **When to choose it ?**
  - When requirements are reasonably well-defined.
  - When overall scope of the development effort suggests a purely linear effort.
  - When limited set of software functionality needed quickly.

Q.14 Explain the prototyping model with its advantages and disadvantages. ☞ **[SPPU : Dec.-13, Marks 6]**

Ans. : • In prototyping model initially the requirement gathering is done.

• Developer and customer define overall objectives; identify areas needing more requirement gathering.

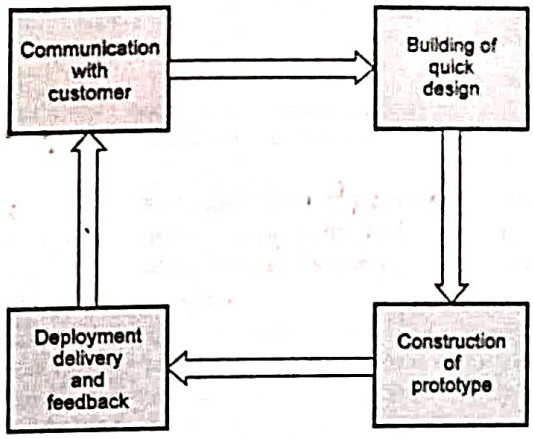


Fig. Q.14.1 Prototyping

- Then a quick design is prepared. This design represents what will be visible to user in input and output format.
- From the quick design a prototype is prepared. Customer or user evaluates the prototype in order to refine the requirements. Iteratively prototype is tuned for satisfying customer requirements. Thus prototype is important to identify the software requirements.
- When working prototype is built, developer use existing program fragments or program generators to throw away the prototype and rebuild the system to high quality.
- **When to choose it ?**
  - Software applications that involve Human-computer Interaction (HCI) the prototyping model is suggested.
  - When the developer is unsure of the efficiency of an algorithm or the adaptability of an operating system then prototype serves as a better choice.

Advantages

1. The software development can be fast and less expensive as at the early stage of development what the user really wants is determined.

2. Due to user involvement in each step of development better and m complete feedback is available in this model.

Disadvantages

1. In the first version itself, customer often wants "few fixes" rather th rebuilding of the system whereas rebuilding of new system maintai high level of quality.
2. The first version may have some compromises.

Q.15 What do you mean by evolutionary process models ? Explai spiral model as an evolutionary process model.

☞ **[SPPU : May-11, Marks 8]**

Ans. : • Evolutionary process model is an iterative process model which there is user involvement in each stage of software development.

- This model possess the iterative nature of prototyping model w controlled and systematic approaches of the linear sequential model.
- The sprial model is divided into a number of framework activitie These framework activities are denoted by task regions.
- The spiral model is as shown in Fig. Q.15.1. (See Fig. Q.15.1 on ne page.)
- In the **initial pass,** product specification is built and in subseque passes around the spiral the prototype gets developed and then mo improved versions of software gets developed.
- During **planning phase,** the cost and schedule of software can b planned and adjusted based on feedback obtained from custome evaluation.
- In spiral model, project **entry point axis** is defined. This axis represent starting point for different types of projects. For instance, concep development project will start at core of spiral and will continue alon the spiral path. If the concept has to be developed into actual projec then at entry point 2 the product development process starts. Hence entry point 2 is called product development project entry point.
- The **task regions** can be described as :

i) **Customer communication** - In this region, it is suggested to establish customer communication.

ii) **Planning** - All planning activities are carried out in order to define resources time line and other project related activities.

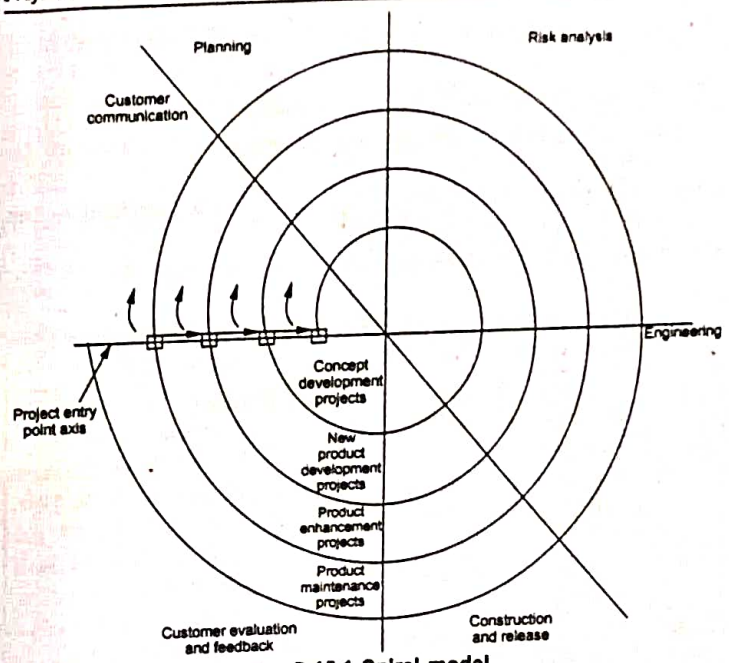


Fig. Q.15.1 Spiral model

iii) **Risk analysis** - The tasks required to calculate technical and management risks are carried out.

iv) **Engineering** - In this task region, tasks required to build one or more representations of applications are carried out.

v) **Construct and release** - All the necessary tasks required to construct, test, install the application are conducted.

vi) **Customer evaluation** - Customer's feedback is obtained and based on customer evaluation required tasks are performed and implemented at installation stage.

Q.16 Provide an overview of the evolutionary development approach and identify key areas of advantage over more traditional development approaches ☞ **[SPPU : May-17 (End Sem), Marks 5]**

Ans. : Evolutionary Development Approach - Refer Q.15.

Advantages of evolutionary development Approach over Traditional Development Approach -

1. All the end users are involved in all the phases of development.
2. Moderate Requirement changes can be made at any stage of development.
3. Due to iterative approach of development, required modifications can be done during any phase of development.

Q.17 Differentiate between waterfall model and spiral model.

Ans. :

Sr. No.	Waterfall model	Spiral model
1.	It requires well **understanding of requirements** and familiarzzz technology.	It is developed in **iterations.** Hence the requirements can be identified at new iterations.
2.	**Difficult** to accommodate **changes** after the process has started.	The required changes can be made at every stage of **new version.**
3.	Can accommodate iteration but indirectly.	It is iterative model.
4.	**Risks** can be identified **at the end** which may cause failure to the product.	**Risks** can be identified and **reduced** before they get problematic.
5.	The custom can see the **working model** of the project only **at the end.** After reviewing of the working model; if the customer gets dissatisfied then it causes serious problems.	The customer can see the working **product at certain stages** of iterations.
6.	**Customers prefer** this model.	Developers prefer this model.
7.	This model is good for **small systems.**	This model is good for **large systems.**
8.	It has **sequential** nature.	It has **evolutionary** nature.

Q.18 Explain the component development model with activities.

☞ [SPPU : Dec.-13, Marks 6]

Ans. : The component based development model makes use of various characteristics of spiral model. This model is evolutionary in nature.

Before beginning the modelling and construction activity of software development the candidate component must be searched and analyzed. The components can be simple functions or can be object oriented classes or methods.

Following steps are applied for component based development -

1. Identify the component based products and analyze them for fitting in the existing application domain.
2. Analyze the component integration issues.
3. Design the software architecture to accommodate the components.
4. Integrate the components into the software architecture.
5. Conduct comprehensive testing for the developed software.

Software reusability is the major advantage of component based development.

Q.19 The Concurrent process model defines a set of "states".Describe what these states represent in your own words, and then indicate how they come in to play within the concurrent process model. ☞[May-17 (End Sem), Marks 5]

Ans. : • The concurrent development model is also called as **concurrent engineering.**

- In this model, the framework activities or software development tasks are represented as **states.**
- The **modeling** or **designing** phase of software development can be in one of the states like *under development*, *waiting for modification*, *under revision* or *under review* and so on.
- Fig. Q.19.1 represents these states. (Refer Fig. Q.19.1 on next page)
- All the software development activities exist concurrently in this model but these activities can be in various **states.**
- These states make transitions. That is during **modeling,** the transition from **under development** state to **waiting for modification** state occurs.
- This model basically defines the **series of events** due to which the transition from one state to another state occurs. This is called **triggering.** These series of events occur for every software development activity, action or task.

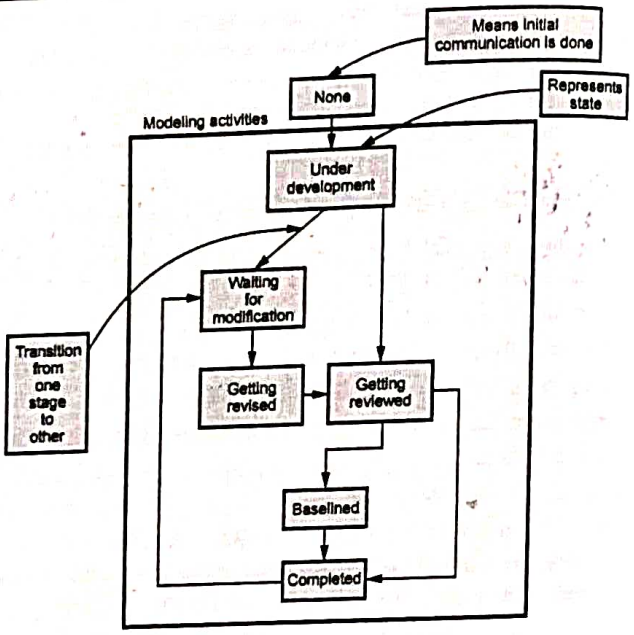


Fig. Q.19.1 Concurrent development model

- This model defines various activities that occur concurrently and network of activities is defined.

Q.20 What are characteristics of software ? What are elements of Rapid Application Development model ?

[SPPU : April-17 (In Sem), Marks 5]

Ans. : Software characteristics :

1. **Software is engineered, not manufactured :** The software produ is developed by systematically following the software development li cycle activities. It is not manufactured with the help of physic components.
2. **Software does not ware out :** Software does not get affected b environmental maladies. But due to undiscovered errors the failure ra is very high and drops down as soon as error get corrected. If an changes are made in some defects may get introduced. This als causes failure rate to be high. Following figure represents the hardwar and software failure curves.

3. **Most software is custom built rather than being assembled from components :** Using the reusable components the software is built. The library of reusable components is also maintained.

Rapid Application Development :

- Various phases in RAD are Requirements Gathering, Analysis and Planning, Design, Build or Construction and finally Deployment.
- Multiple teams work on developing the software system using RAD model parallely.
- In the **requirements gathering** phase the developers communicate with the users of the system and understand the business process and requirements of the software system.
- During **analysis and planning** phase, the analysis on the gathered requirements is made and a planning for various software development activities is done.
- During the **design** phase various models are created. Those models are Business model, data model and process model.

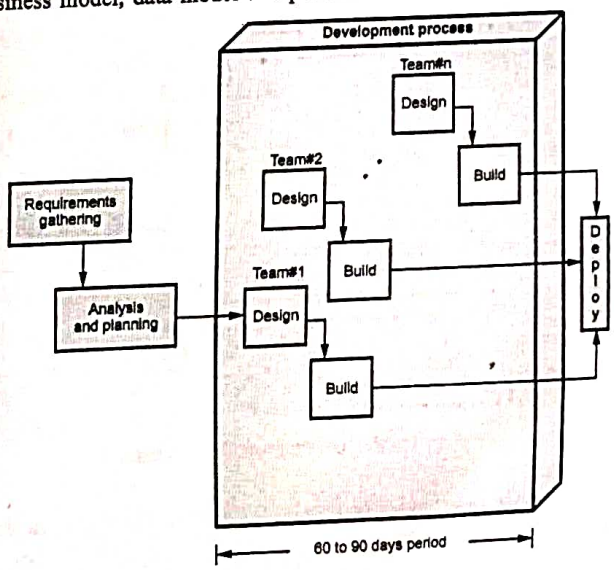


Fig. Q.20.1 Rapid application development

- The **build** is an activity in which using the existing software components and automatic code generation tool the implementation code is created for the software system. This code is well tested by its team. The functionalities developed by all the teams are integrated to form a whole.
- Finally the deployment of all the software components (created by various teams working on the project) is carried out.

Q.21 What is unified process ?

Ans. : The unified process is a framework for object oriented models. This models is also called as Rational Unified Process Model(RUP). This model is iterative and incremental in nature.

Q.22 Explain various phases of unified process.

Ans. : Various phases of unified process are -

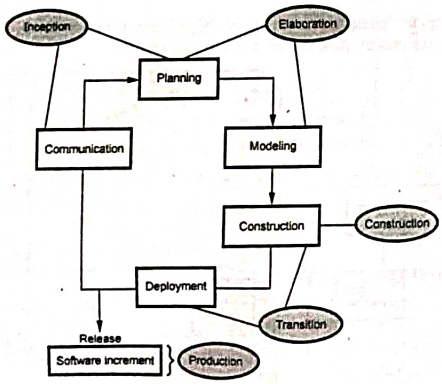


Fig. Q.22.1 Unified process model

1. Inception

- In this phase there are two major activities that are conducted : *Communication* and *planning*.
- By having customer communication business requirements can be identified

- Then a rough architecture of the system is proposed. Using this rough architecture it then becomes easy to make a plan for the project.

2. Elaboration

- Elaboration can be done using two activities : *Planning* and *Modelling*.
- In this phase the use cases are redefined. And an architectural representation is created using five models such as use-case model, analysis model, design model, implementation model and deployment model.

3. Construction

- The main activity in this phase is to make the use cases **operational**.
- The analysis and design activities that are started in elaboration phase are completed in this phase and a **source code** is developed which implements all desired functionalities.
- Then unit testing is conducted and acceptance testing is carried out on the use cases.

4. Transition

- In the transition phase all the activities that are required at the time of deployment of the software product are carried out.
- Beta **testing** is conducted when software is delivered to the end user.
- **User feedback** report is used to remove defects from the created system.
- Finally software team prepares user manuals, installation guides and trouble shooting procedures. This makes the software more usable at the time of release.

5. Production

- This is the final phase of this model. In this phase mainly the maintenance activities are conducted in order to support the user in operational environment.

1.8 : Agile Software Development

Q.23 What is Agility ? List the principles of Agility.

☞ **[SPPU : April-17 (In Sem), Marks 5]**

Ans. : • The agile processes are the **light-weight** methods are **people-based** rather than plan-based methods.

- The agile process forces the development team to **focus** on **software** itself rather than design and documentation.
- The agile process believes in **iterative method.** The aim of agile process is to **deliver** the working model of software **quickly** to the customer.
- **For example : Extreme programming** is the best known of agile process.

Principles of agility

1. **Satisfy the customer** by early and **continuous delivery** of valuable software.
2. The **changes** in the requirements must be **accommodated**. Even though the changes occur late in the software development process, the agile process should help to accommodate them.
3. **Deliver working** software quite often. Within the **shorter time** span deliver the working unit.
4. Business people and developers must **work together** throughput the project.
5. Motivate the people who are building the projects. **Provide** the **environment** and support to the development team and trust them for the job to be done.
6. The **working software** is the primary measure of the progress of the software development.
7. The agile software development approach promote the constant project development. The **constant speed** for the development of the product must be maintained.
8. To enhance the agility there should be **continuous technical excellence**.
9. The proper attention to be given to **technical excellence** and good **design**.

Q.24 Explain the influence of software development approach on agile process.

Ans. : • If the incremental delivery is combined with agile practices such as continuous unit testing and pair programming then the cost of changes can be controlled.

- The following graph represents how the software development approach has a strong influence on the development cost .

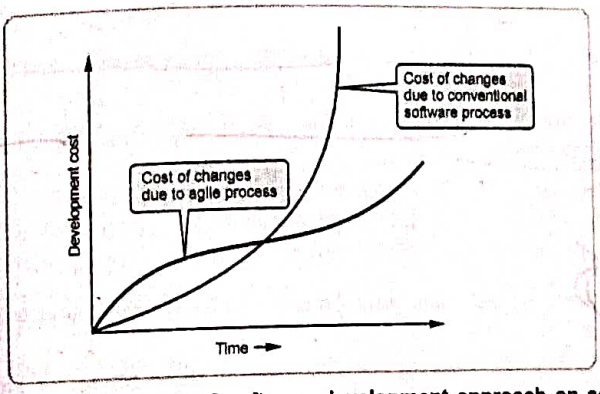


Fig. Q.24.1 Influence of software development approach on agile process

Q.25 Explain the concept of software evolution. Explain the merits and demerits of agile process model

[SPPU : April-15, In Sem, Marks 4]

Ans. : Software evolution : Software evolution is a process of developing a software initially and repeatedly updating it for various reasons. Software change occurs because of following reasons.

1. New requirements emerge when the software is used.
2. The business environment changes.
3. Errors needs to be repaired.
4. New equipment must be accommodated.
5. The performance or reliability may have to be improved.

Agile process model

Merits :

1) Customer satisfaction can be attained by rapid and continuous delivery of useful software.
2) Customer, developer and tester interact with each other during software development process.
3) Continuous attention can be given for excellent technical design and software quality.
4) Even late changes in requirements can be accommodated.

Demerits :

1) There is lack of emphasis on necessary designing and documentation during software development process.
2) The project can easily get off the track if customer is not clear about his requirements.

1.9 : Agile Methods

Q.26 Write a short note on - Adaptive Software Development(ASD) method.

Ans. : • The life cycle of ASD consists of three phases of software development and those are -

1. Speculation 2. Collaboration 3. Learning.

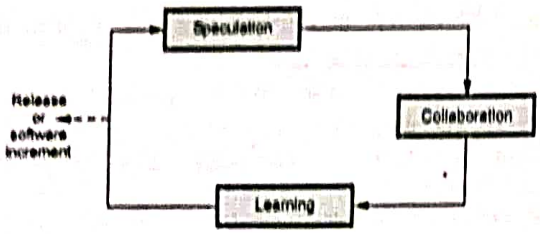


Fig. Q.26.1 Adaptive software development life cycle

1. **Speculation :** This is an initial phase of the adaptive software development process. In this phase the **adaptive cycle planning** is conducted. In this cycle planning mainly three types of information is used such as - Customer's mission statement, project constraints (delivery date, user description, budgets and so on) and basic requirements of the project.
2. **Collaboration :** The motivated people work in collaboration to develop the desired software product. In this phase **collaboration among the members** of development team is a key factor. For successful collaboration and coordination it is necessary to have following qualities in every individual -
   - Assist each other without resentment
   - Work hard.

   - Posses the required skill set.
   - Communicate problems and help each other to accomplish the given task.
   - Criticize without any hate.
3. **Learning :** As the team members go on developing the components, the emphasize is on learning new skills and techniques. There are three ways by which the team members learn -
   - **Focus groups :** The feedback from the end-users is obtained about the software component being developed. Thus direct feedback about the developed component can be obtained.
   - **Formal technical review :** This review for software components is conducted for better quality.
   - **Postmortems :** The team analyses its own performance and makes appropriate improvements.

Q.27 What is SCRUM ?

Ans. : • SCRUM is an **agile process model** which is used for developing the complex software systems.

• It is a **lightweight process framework** that can be used to manage and control the software development using **iterative and incremental** approach.

Q.28 Explain in detail the development activities in SCRUM.

Ans. : In SCRUM emphasize is on **software process pattern.** The software process pattern defines a set of development activities. Refer Fig. Q.28.1.

Various development activities in SCRUM are -

1. **Backlog :** It is basically **a list** of project requirements or features that must be provided to the customer. The items can be included in the backlog list at any time. The product manager analyses this list and updates the priorities as per the requirements.
2. **Sprint :** These are the **work units** that are needed to achieve the requirements mentioned in the backlogs. Typically the sprints have **fixed duration or time-box** (typically of 2 to 4 weeks). Thus sprints allow the team members to work in stable and short-term environment.

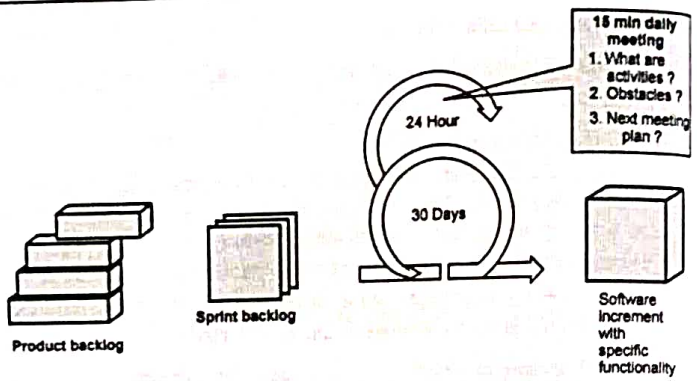


Fig. Q.28.1 SCRUM workflow activities

Meetings : These are 15 minutes daily meetings to report the completed activities, obstacles and plan for next activities. Following are three questions that are mainly discussed during the meetings

i) What are the tasks done since last meeting ?

ii) What are the issues (obstacles) that team is facing ?

iii) What are the next activities that are planned ?

4. **Demo :** During this phase, the software increment is delivered to the customer. The implemented functionality which is demonstrated to the customer. Note that demo focuses on only implemented functionalities and not all the planned functionalities (and yet to get implemented) of the software product.

Q.29 What is Feature Driven Development(FDD) ?

Ans. : • In FDD, the feature means **client valued function.** It is an iterative and incremental software development process

In FDD, the collaborative activities are carried out. These activities are called as **process** as shown in Fig. Q.29.1.

• Various phases in the FDD life cycle are

1. **Develop overall model :** In this phase the high-level walkthrough of scope and detailed domain walkthroughs are conducted. Later on peer reviews and discussions are carried out on these walkthroughs and domain area models are created. These domain area models are then merged into the overall models.

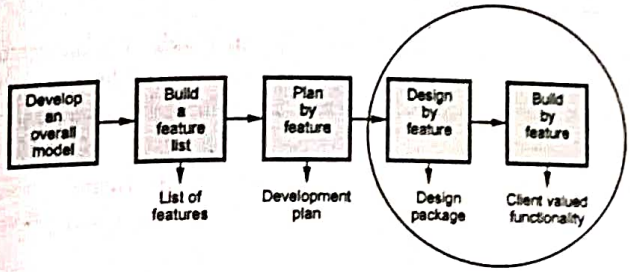


Fig. Q.29.1 Feature driven development life cycle

2. **Build features list :** Initially the list of **features** is created. The domain is functionally decomposed into various **subject areas.** These subject areas contain the **business activities.** The steps within business activity forms the categorized feature list. Features are basically the **client valued functions** and can be expressed in the form.

<action> <result> <by|for|of|to> <object>

For example

"Display product-specifications of the product"

<Action> <result> <of> <object>

3. **Plan by feature :** After completing the building of feature list the development plan is created. The features are assigned as **classes** and are chief programmer or the class owner is assigned with appropriate classes.

4. **Design by feature :** A **design package** was produced for each feature. A chief programmer selects a small group of features and these features are to be developed within two weeks. For each feature the **sequence diagram** is created.

5. **Build by feature :** Finally a complete client valued function is developed for each feature. The class owners develop the actual code for their classes and this code is promoted to the main build.

1.10 : Plan Driven and Agile Development

Q.30 Explain the plan driven and agile development approach.

Ans. : • **Agile Development :**

- In agile approach, the **design and implementation** are the central activities and other activities such as requirement elicitation and analysis, testing and so on are integrated with design and implementation.
- The agile approach is not completely code focused, some necessary documents such as design specification can be produced in this approach.

• **Plan Driven Development :**

- On the hand, in plan driven development approach each activity is performed in separated stage and output of each stage is used for **planning** the next stage activity.
- In plan driven approach **the iterations** occur within the activities.
- The formal documents used to communicate between stages of software development process.
- The incremental delivery is possible in plan driven development.

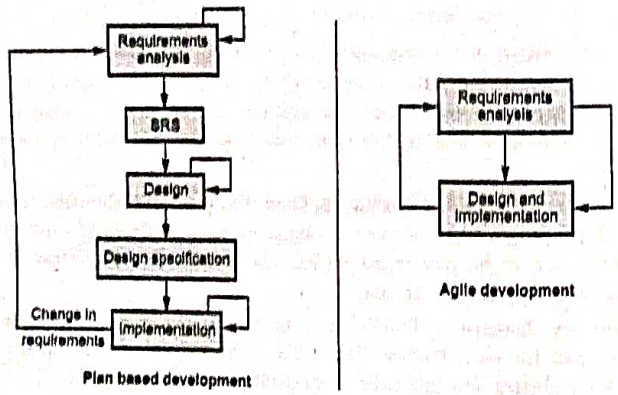


Fig. Q.30.1 Plan driven development and Agile development

1.11 : Extreme Programming Practices

Q.31 What is extreme programming ? Explain the extreme programming process.

[SPPU : Dec.-15, End Sem-15, Marks 7]

Ans. : Extreme Programming (XP) is one of the best known agile process. The extreme programming process is explained as follows -

- Customer specifies and priorities the system requirements. Customer becomes one of the important members of development team. The developer and customer together prepare a **story-card** in which customer needs are mentioned.
- The developer team then aims to implement the scenarios in the story-card.
- After developing the story-card the development team breaks down the total work in **small tasks.** The efforts and the estimated resources required for these tasks are estimated.
- The customer priorities the stories for implementation. If the requirement changes then sometimes unimplemented stories have to be discarded. Then release the complete software in **small** and frequent **releases.**
- For accommodating new changes, **new story-card** must be developed.
- Evaluate, the system along with the customer.

This process is demonstrated by the following Fig. Q.31.1.

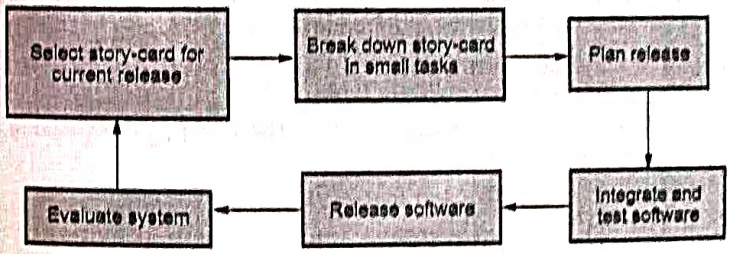


Fig. Q.31.1 Extreme programming release cycle

1.12 : Testing in XP

Q.32 State and explain various features of testing in XP.

Ans. : Following are the features of testing in XP

1. Test First Development :

- Test first development is an approach in which the tests are written before the code.
- One can run the test while executing the code and can discover the bugs during development.
- In this approach it is necessary to understand the specification to write the test cases for the code.

2. Incremental test development from scenarios :

- In XP user requirements are represented in the form of scenarios or stories.
- Each the story card can be created for each important task.
- The one or more unit tests can be prepared for each task.

3. Involving user in testing :

- User involvement in testing is for the acceptance testing for the stories.
- Acceptance testing is a kind of testing in which user is involved in testing process to check whether the system is created according to the requirements.

4. Use of automated test tools :

- In test automation the tests are written as executable components before writing the code. These testing components are standalone units.
- Junit is a widely used test tool in automation testing.

1.13 : Pair Programming

...programming technique ? Specify its advantages.

Ans. : • In any pair programming technique one person writes the code while the other person reviews it constantly while getting typed. When errors or defects are noticed, they are immediately removed by the pair.

• While developing the code data structures, algorithms, the coding strategies are discussed by the pair. The role rotates frequently and makes both the partners to participate actively.

1.14 : Introduction to Agile Tools

Q.34 Write a short note on - Agile tool : JIRA.

Ans. : • JIRA is an agile tool developed by a company named Atlassian.

• This tool is used for **bug tracking, issue tracking** and project management.

• The name of this tool is inherited from the Japanese word "Gojira"

• JIRA is written in **Java.**

• The **main features** of Jira for agile software development are the functionality to plan development iterations, the iteration reports and the bug tracking functionality.

• The tool is basically used for project management activities. It is also used to track issues bugs related to the software and mobile app.

Q.35 Write a short note on - Agile tool : Kanban.

Ans. : • Kanban is a new technique for managing an agile software development process in a highly efficient way.

• It makes use of the **Just In Time(JIT)** approach.

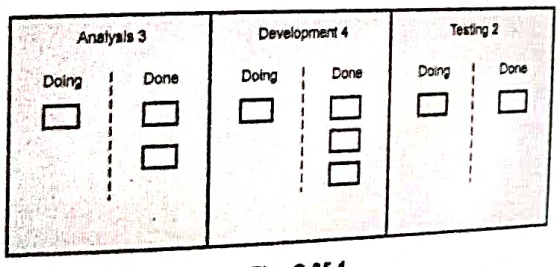
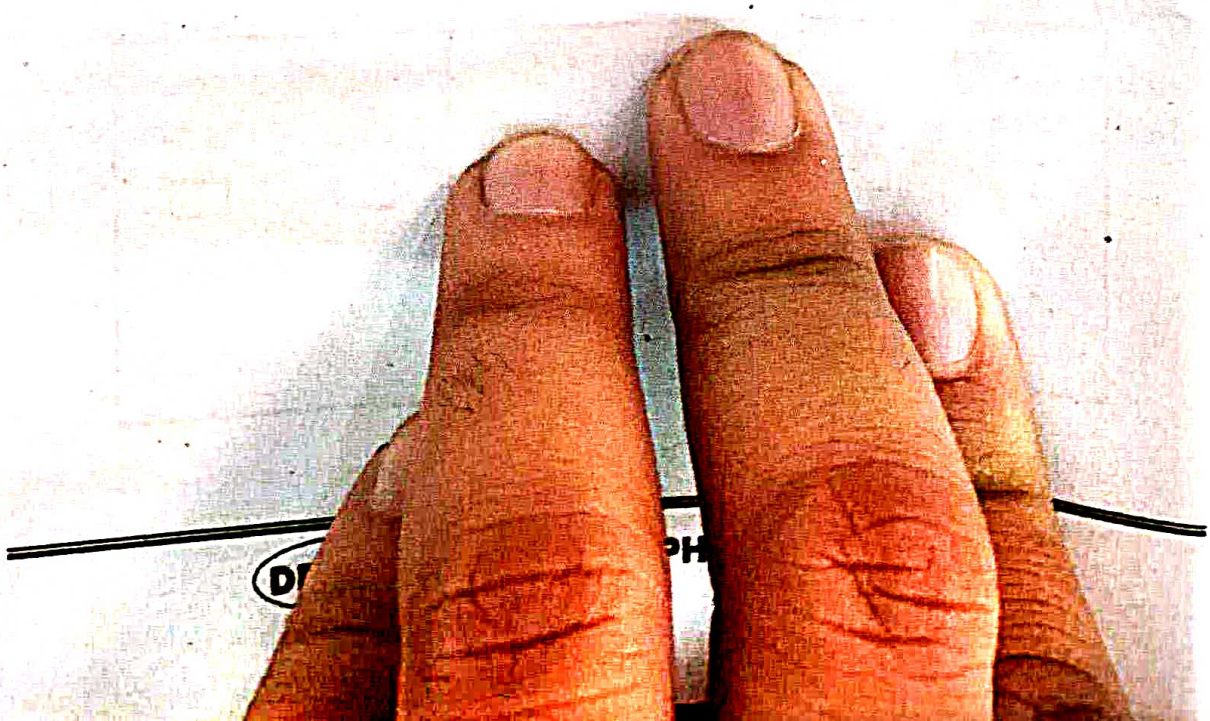


Fig. Q.35.1

- In Kanban technique, the "Work picture" can be created by using **visual information.**
- The board above shows a situation where the developers and analysts are tracking the agile software development process. The software development phases are represented in which the number in each column indicate the number of items to be analysed. Each phase is then sub divided into two more columns - **Doing** and **Done**. The tasks that are under **Done** columns will be then promoted to the next phase.

END...



2 Software Requirements Engineering and Analysis

2.1 : Requirement Engineering

Q.1 What is requirements engineering process ?

Ans. : Requirement engineering is the process of i) Establishing the services that the customer requires from a system and ii) The constraints under which it operates and is developed.

Q.2 Why it is necessary to have the stable and correct requirements in software engineering ?

Ans. : Following are those reasons which specify that there is a need for the requirements to be stable and correct -

i) As requirements are identified and analysis model is created the software team and other stakeholders negotiate the priority, availability and relative cost of the requirement. During this negotiation there are chances that the requirements may get changed.

ii) During requirement analysis, each requirement is validated against the needs of the customer. If the requirements are stable, correct and unambiguous then they remain unchanged throughout the requirement analysis process. And finally the working model that satisfied customers need can be created effectively.

Q.3 What are the characteristics that requirement must meet ?

☞ [SPPU : May-14, Marks 6]

Ans. : 1. The requirements must be unambiguous

2. It should be complete.
3. The requirement must be correct.
4. The requirements must be consistent and should not get changed as the software evolves.
5. It should be cost effective.
6. It should be traceable.

Q.4 What tasks are to be carried out in software requirement engineering ? Explain in detail.

☞ [SPPU : May-15, End Sem-15, Marks 7, April-17, In Sem, Marks 5]

OR Explain the elements of requirements model.

☞ [SPPU : May-13, Marks 8]

Ans. : Requirement engineering process performs following seven distinct functions -

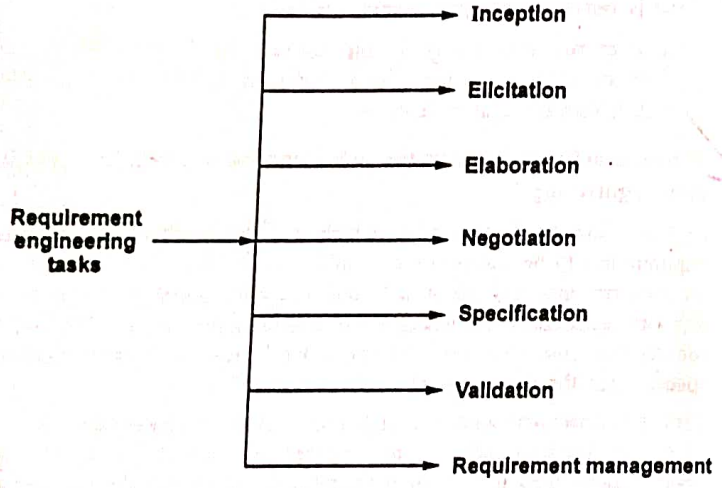


Fig. Q.4.1 Requirement engineering tasks

1. **Inception :** The inception means specifying the beginning of the software project. Most of the software projects get started due to business requirements. There may be potential demand from the market for a particular product and then the specific software project needs to be developed.
2. **Elicitation :** Before the requirements can be analyzed and modelled they must undergo through the process of elicitation process. Requirements elicitation means requirements discovery.
3. **Elaboration :** Elaboration is an activity in which the information about the requirements is expanded and refined. This information is gained during inception and elicitation. The goal of elaboration activity is to prepare a technical model of software functions, features and constraints.
4. **Negotiation :** Sometimes customer may demand for more than that is achieved or there are certain situations in which customer demands for something which cannot be achieved in limited business resources. To handle such situations requirement engineers must convince the customers or end users by solving various conflicts.
5. **Specification :** A specification can be a written document, mathematical or graphical model, collection of use case scenarios or may be the prototypes.
6. **Validation :** Requirement Validation is an activity in which requirement specification is analyzed in order to ensure that the requirements are specified unambiguously.
7. **Requirement Management :** Requirements management is the process of managing changing requirements during the requirements engineering process and system development.

2.2 : User and System Requirements

Q.5 What are user and system requirements ?

Ans. : • **User requirements :** The user requirements should describe functional and non functional requirements in such a way that they are understandable by system users who don't have detailed technical knowledge.

• User requirements are defined using natural language, tables and diagrams because these are the representations that can be understood by all users.

• **System requirements :** System requirements are more detailed specifications of system functions, services and constraints than user requirements.

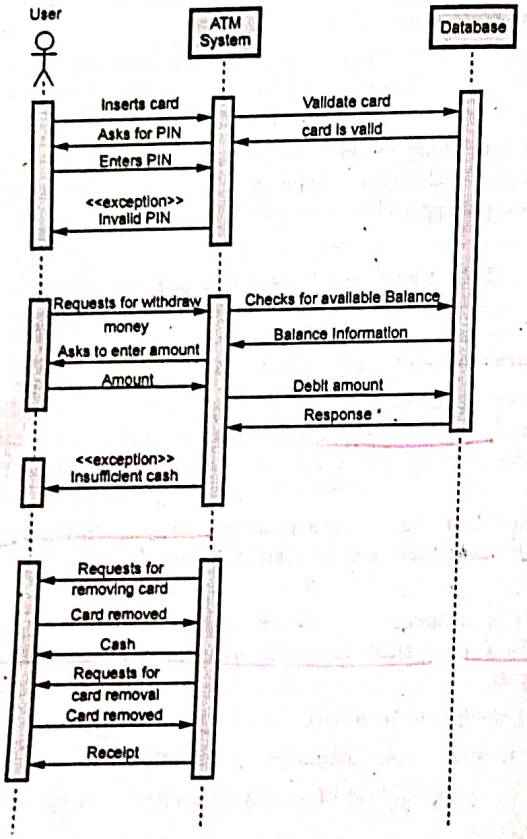
• They are intended to be a basis for designing the system.

Q.6 What is Structured Language Specification ?

Ans. : • One of the method of writing requirements is by using structured natural language.

• All the requirements should be written in a **standard way** while using structured language specification.

- The **advantage** of specifying requirements using this method is that requirement become **understandable** and **expressive.**
- The only necessary thing while writing requirements using natural language is that some degree of **uniformity must be maintained.**
- Extra information can be added when the requirements are written using natural language. This information can be represented using tables or graphical models.
- One way of using graphical model is use of **sequence diagram.**



- The sequence diagram represents the sequence of actions that user performs while interacting the system.
- Example : Fig. Q.6.1 shows a sequence diagram for withdrawal of cash from ATM.

2.3 : Functional and Non Functional Requirements

Q.7 What are functional and non functional requirements of software ? ☞ **[SPPU : May-15, In Sem-15, Marks 4]**

Ans. : Functional requirements : Functional requirements should describe all the required functionality of system services.

Functional user requirements are the high-level statements of what the system should do but functional system requirements should describe the system services in detail.

For example : Consider a library system in which there is a single interface provided to multiple databases. These databases are collection of articles from different libraries. A user can search for, download and print these articles for a personal study.

From this example we can obtain functional requirements as,

1. The user shall be able to search either all of the initial set of databases or select a subset from it.
2. The system shall provide appropriate viewers for the user to read documents in the document store.

Non functional requirements : The non functional requirements define system properties and constraints. Various properties of a system can be : Reliability, response time, storage requirements. And constraints of the system can be : Input and output device capability, system representations etc.

- Non functional requirements are more critical than functional requirements. If the non functional requirements do not meet then the complete system is of no use.

Q.8 Enlist various functional and non functional requirements for the Bank ATM system.

Ans. : Functional requirements

1. There should be the facility for the customer to insert a card.

2. The system should first validate card and PIN.
3. The system should allow the customer to deposit amount in the bank.
4. The system should dispense the cash on withdrawal.
5. The system should provide the printout for the transaction.
6. The system should make the record of the transactions made by particular customer.
7. On invalid PIN entry for three times the card should be retained by the system.
8. The cash withdrawal is allowed in multiple of 100.
9. The cash deposition is allowed in multiple of 100.
10. The customer is allowed to transfer amount between the two accounts.
11. The customer is allowed to know the balance enquiry.
12. The customer is allowed to get the printout for desired transaction.
13. The system should be efficient.

Non functional requirements

1. Each of the transaction should be made within 60 seconds. If the time limit is exceeded, then cancel the transaction automatically.
2. If there is no response from the bank computer after request is made within the minutes then the card is rejected with error message.
3. The bank dispenses money only after the processing of withdrawal from the bank. That means if sufficient fund is available in user's account then only the withdrawal request is processed.
4. Each bank should process the transactions from several ATM centers at the same time.
5. The machine should be loaded with sufficient fund in it.

Q.9 Differentiate between functional and non functional requirements

Ans. :

Functional requirements	Non functional requirements
The functional requirements specify the features of the software system.	The non functional requirements specify the properties of the software system.

Functional requirements describe what the product must do.	Non functional requirements describe how the product should perform.
The functional requirements specify the actions with which the work is concerned.	The non functional requirements specify the experience of the user while using the system.
Example : For a library management system, allowing user to read the article online is a functional requirement.	Example : For a library management system, for a user who wishes to read the article online must be authenticated first.

2.4 : Spiral View of Requirement Engineering Process

Q.10 Explain the requirement engineering process along with its spiral view.

Ans. :

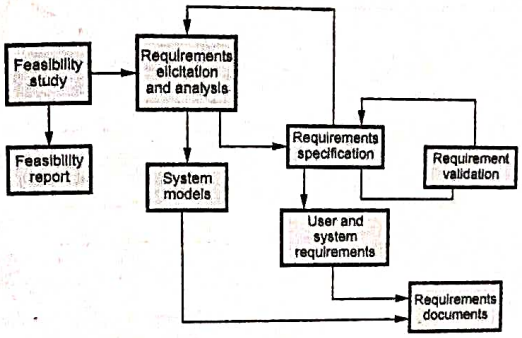


Fig. Q.10.1 Requirement engineering process

- A Requirement Engineering is a process in which various activities such as **discovery, analysis** and **validation** of system requirements are done.
- It begins with feasibility study of the system and ends up with requirement validation. Finally the requirement document has to be prepared.

- This process is a three stage activity where the activities are arranged in the iterative manner. In the early stage of this process most of the time is spent on understanding the system by understanding the high-level non functional requirements and user requirements. The spiral model of requirement engineering process is as shown below.
- The generic activities that are common to all processes are
  1. Requirements elicitation
  2. Requirements analysis
  3. Requirements validation
  4. Requirements management

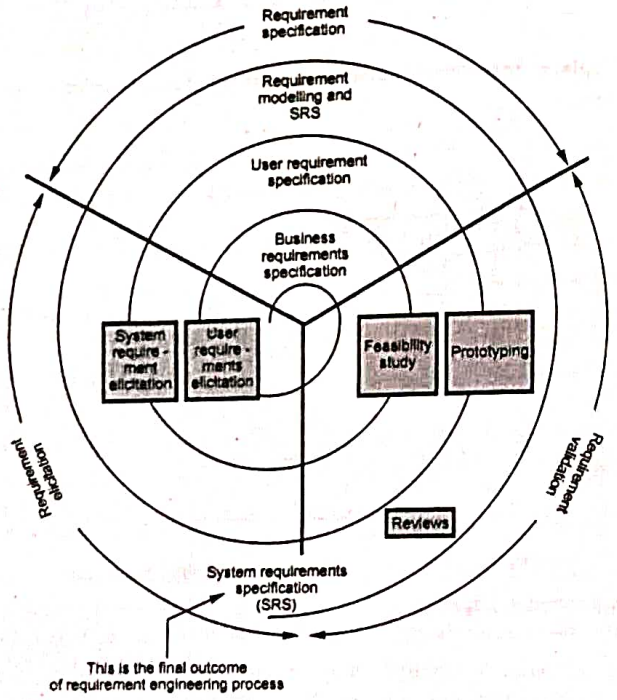


Fig. Q.10.2 Spiral model of requirements engineering process

- **Requirements engineering process can also be viewed as structured analysis method** in which the system is analysed fully some **system models** are prepared. Particularly use cases are developed which help in exposing the functionalities of the systems.
- Along with creation of system models some **additional information** is also provided in the requirement engineering process.

2.5 : Software Requirements Specification(SRS)

Q.11 Explain four desirable components of a good software requirements specification document(SRS)

☞ **[SPPU : April-15, In Sem-15, Marks 6]**

Ans. : The software requirements document is the specification of the system.The standard template for writing SRS is as given below.

1. Introduction

1.1 Purpose of this document : Describes the purpose of the document.

1.2 Scope of this document : Describes the scope of this requirements definition effort. This section also details any constraints that were placed upon the requirements elicitation process, such as schedules, costs.

1.3 Overview : Provides a brief overview of the product defined as a result of the requirements elicitation process.

2. General Description

Describes the general functionality of the product such as similar system information, user characteristics, user objective, general constraints placed on design team.

Describes the features of the user community, including their expected expertise with software systems and the application domain.

3. Functional Requirements

This section lists the functional requirements in ranked order. A functional requirement describes the possible effects of a software system, in other words, what the system must accomplish. Each functional requirement should be specified in following manner

Short, imperative sentence stating highest ranked functional requirement

1. **Description :** A full description of the requirement.
2. **Criticality :** Describes how essential this requirement is to the overall system.
3. **Technical issues :** Describes any design or implementation issues involved in satisfying this requirement.
4. **Cost and schedule :** Describes the relative or absolute costs of the system.
5. **Risks :** Describes the circumstances under which this requirement might not able to be satisfied.
6. **Dependencies with other requirements :** Describes interactions with other requirements.
7. Any other appropriate

4. Interface Requirements

This section describes how the software interfaces with other software products or users for input or output. Examples of such interfaces include library routines, token streams, shared memory, data streams and so forth.

4.1 User Interfaces : Describes how this product interfaces with the user.

4.1.1 GUI : Describes the graphical user interface if present. This section should include a set of screen dumps to illustrate user interface features.

4.1.2 CLI : Describes the command-line interface if present. For each command, a description of all arguments and example values and invocations should be provided.

4.1.3 API : Describes the application programming interface, if present.

4.2 Hardware Interfaces : Describes interfaces to hardware devices.

4.3 Communications Interfaces : Describes network interfaces.

4.4 Software Interfaces : Describes any remaining software interfaces not included above.

5. Performance Requirements

Specifies speed and memory requirements.

6. Design Constraints

Specifies any constraints for the design team such as software or hardware limitations.

7. Other Non Functional Attributes

Specifies any other particular non functional attributes required by the system. Such as :

7.1 Security

7.2 Binary Compatibility

7.3 Reliability

7.4 Maintainability

7.5 Portability

7.6 Extensibility

7.7 Reusability

7.8 Application Compatibility

7.9 Resource Utilization

7.10 Serviceability

... others as appropriate

8. Operational Scenarios

This section should describe a set of scenarios that illustrate, from the user's perspective, what will be experienced when utilizing the system under various situations.

9. Preliminary Schedule

This section provides an initial version of the project plan, including the major tasks to be accomplished, their interdependencies, and their tentative start/stop dates.

10. Preliminary Budget

This section provides an initial budget for the project.

11. Appendices

11.1 Definitions, Acronyms, Abbreviations : Provides definitions terms, and acronyms, can be provided.

11.2 References : Provides complete citations to all documents and meetings referenced.

Q.12 Explain the desirable characteristics of SRS. — Software Requirements Specification

Ans. : Following are some characteristics of a good SRS -

1. **Correct :** The SRS must be correct. That means all the requirements must be correctly mentioned, or the requirements must be realistic by nature.
2. **Complete :** To make the SRS complete, it should be specified what a software designer wants to create a software. The SRS is said to be complete only in following situations -
   1) When SRS consists of all the requirements related to functionality, performance, attributes, design constraints or external interfaces.
   2) When labels and corresponding references are mentioned for all the figures, diagrams and tables in the SRS.
   3) When expected responses to the input data is mentioned by considering validity and invalidity of an input.
3. **Unambiguous :** When requirements are understood correctly then only unambiguous SRS can be written. Unambiguous specification means only one interpretation can be made from the specified requirements.
4. **Consistent :** If there are not conflicts in the specified requirements then SRS is said to be consistent.
5. **Stability :** In SRS, it is not possible to specify all the requirements. The SRS must contain all the essential requirements. Each requirement must be clear and explicit.
6. **Verifiable :** The SRS should be written in such a manner that the requirements that are specified within it must be satisfied by the software. Testable, Ranked, Valid, Modifiable
7. **Traceable :** If origin of requirement is properly given or references of the requirements are correctly mentioned then such a requirement is called as traceable requirement.

Q.13 Give the structured and Tabular SRS for Insulin Pump.

Ans. : Structured SRS

1.1 Purpose :

Compute the insulin dose to maintain the safe sugar level.

1.2 Problem Description :

When current measured sugar level is in safe zone then compute how much insulin is required to deliver.

1.3 Input :

Three inputs are considered, previous two sugar level readings(L1,L2) and the current sugar level in(L3).

1.4 Output :

The appropriate dose of the insulin to be delivered(say d).

1.5 Source :

Current sugar reading from sensor. Other readings from memory.

1.6 Destination :

Main control loop.

1.7 Action :

i) If sugar level is stable or falling then d i.e. the dose of the insulin is 0.

ii) If sugar level is increasing and rate of increase is decreasing then the dose of insulin is 0.

iii) If sugar level is increasing and rate of increase is stable or it is increasing then the insulin dose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result.
If the result, is rounded to zero then this dose is set to the minimum dose that can be delivered.

1.8 Requirements :

Two previous readings which help to compute the rate of change in the sugar level.

1.9 Pre-Condition

The maximum allowed single dose of insulin must be present in the insulin reservoir.

1.10 Post-Condition

The sugar level reading L1 is replaced by level L2 and then level L2 is replaced by level L3

1.11 Side Effects

None

Tabular SRS

Status	Condition	Action
Sugar level falling	L3 < L2	d = 0
Sugar level stable	L3 = L2	d = 0
Sugar level increasing and rate of increase is decreasing	(L3 - L2) < (L2 - L1)	d = 0
Sugar level is increasing and rate of increase is stable or increasing	(L3 - L2) >= (L2-L1)	d = round((L3 - L2)/4) If rounded result is 0 then d=min_dose

2.6 : Requirements Elicitation and Analysis

Q.14 Explain the requirement elicitation and Analysis process.

Ans. : The spiral model as given below depicts the requirement elicitation and analysis process.

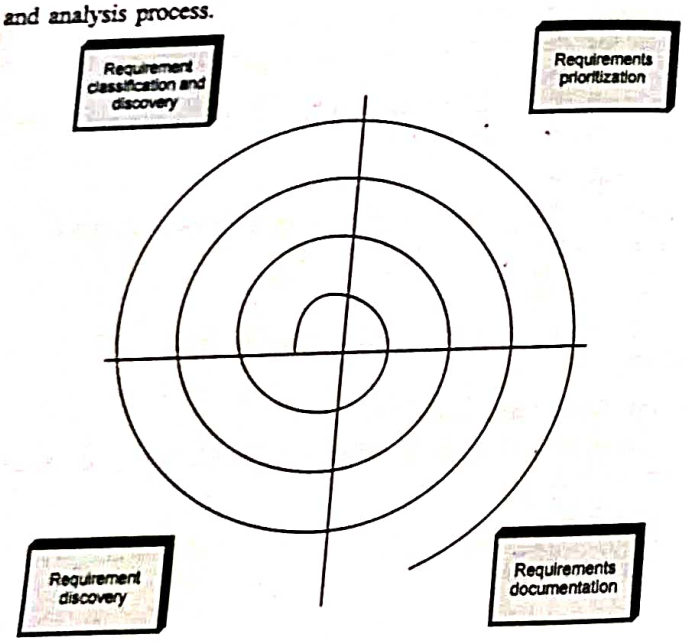


Fig. Q.14.1 Requirements elicitation and analysis process

The process activities are -

1. **Requirement discovery :** By having effective communication with the customers the requirements can be identified.
2. **Requirements classification and discovery :** All the unstructured requirements can be categorised systematically depending upon their nature. And they are arranged in groups.
3. **Requirement prioritization :** There are some conflicting requirements. Hence the requirements are prioritized first. If there are some unrealistic requirements then negotiations are made and only realistic prioritized requirements are collected. If any conflict occurs then it its resolved by requirement engineers.
4. **Requirement documentation :** This is the specification of all the requirements. And an important requirement document is created.

Q.15 Explain the techniques of requirement discovery.

Ans. : Requirement discovery means finding all relevant information about the system. The **sources of information** for requirement gathering are : documentation, system stakeholders and specification some other system which is of similar kind. Various **methods** of requirement discovery are

1. Interviewing

- This is an effective method of requirement gathering. The requirement engineering team communicates the stakeholders by asking them various questions about the system and its use. From the answers the requirements can be identified.

2. View point

- Viewpoint provides the perspective to the system and using these perspectives the requirements can be discovered.
- The viewpoint is also useful in classifying the stakeholders.

3. Use cases

- Use cases are the **fundamental units** of modelling language, in which functionalities are distinctly presented.
- The use case is a scenario based technique.

- Use cases help to identify individual interactions with the system. Use-cases are extensively used for requirements elicitation.
- By designing the proper use cases for different scenarios major requirements of the system can be identified.

4. Ethography

- Ethnography is a technique of observation which is used to understand social and organizational requirements.
- The system analyst imagines himself as a part of the working environment in which the system will be used. He then observes the day-to-day activities and notes down the needs and tasks required by the organization.
- The analysts thus finds the implicit requirements of the system.

Q.16 State and explain the principle stages of VORD.

Ans. : The Viewpoint Oriented Requirement Definition (VORD) is a service oriented definition for supporting requirement analysis and elicitation.

The principle stages of VORD are -

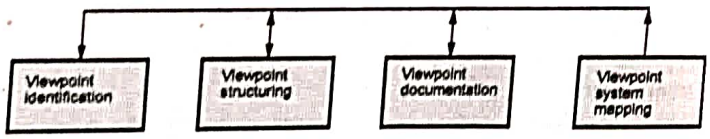


Fig. Q.16.1

Viewpoint identification : Viewpoint identification is a stage in which viewpoints to which the system services are received are identified. Similarly the services received to these viewpoints must be identified.

Viewpoint structuring : All the related viewpoints are grouped into the hierarchy. Common services are denoted by the top level of this hierarchy.

Viewpoint documentation : The identified viewpoints and services are described in detail.

Viewpoint system mapping : The transformation of system analysis is done into the object oriented design.

Q.17 Design the use case for ATM system.

Ans. :

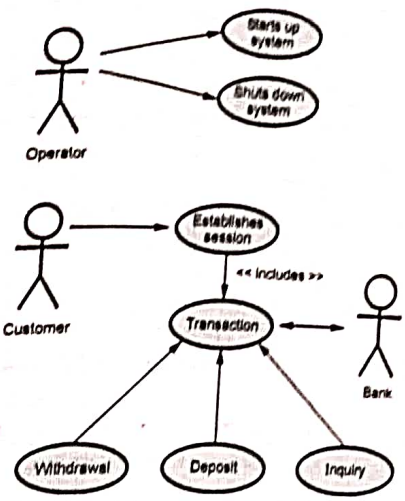


Fig. Q.17.1 Use cases for ATM system

Q.18 How are cases can be used to model the requirements ? Write an usecase for 'login' with a template and diagram.

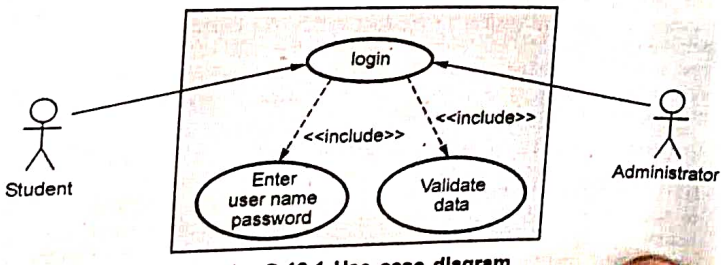
☞ **[SPPU : May-11, Marks 8]**

Ans. : • Use cases are fundamental units of modelling language in which functionalities are distinctly represented.

- Use-case depicts the software or system from end-user's point of view.
- The first step in writing use cases is to identify actors. Actors are entities that use the system or product within the context of function and behaviour of the system. Actors represent the role of the people as the system gets operated. Actor is anything that communicates with the system or product. It is external to the system. Every actor has one or more goals when using the system.
- Requirement elicitation is an evolutionary activity. It is not possible to identify all the actors in the first iteration itself. Hence primary actors are identified in the first iteration. These actors interact with the system to achieve required function. In next subsequent iterations, the secondary actors can be identified. The secondary actors support the system in [illegible] actors can perform their task.

- After identifying actors next step is to develop the use cases.
- The basic use case represents the high level scenario that describes the interaction between the actor and the system.
- Example- Use case for login with template and diagram

Use case template for login	
Use case	Course registration system.
Primary Actor	Student.
Goal in Context	To monitor all the functionalities required to register for the course.
Preconditions	None
Scenario / Basic Flow	1. The sytem requests student to enter his name and password. 2. The student enters his/her name and password. 3. The system validates the entered name and password and logs the student into the system.
Exceptions	1. The control panel is not ready. 2. User name is invalid. 3. Password is incorrect.
Priority	Essential and must be implemented in course registration system.
Postcondition	If user logs in successfully then course information must be displayed to him / her.
Secondary Actor	Administrator.



Q.19 What is ethnography ? How ethnography is effective in discovering types of requirements.

Ans. : Ethnography - Refer Q.15.

Effectiveness of Ethnography in discovering requirements - Two types of requirements can be discovered by ethnography.

i) Requirements derived from the way in which **people work** - For example in Microsoft word, spell check facility should be embedded in editor, so that while typing user will come to know his spelling mistakes.

ii) Requirements that are derived from **co-operation and awareness of other** people's activities.

- The technique of ethnography can be combined with prototype.

1. Requirements obtained from working style of people

These are the requirements that can be identified from the sequence of actions that a user is performing. For example in ATM system when the user enters the PIN, the card validity action takes place. Unless and until the card gets validated there should not be any transaction processing. That means, it is required that a valid card, valid PIN must be entered for getting the money from ATM.

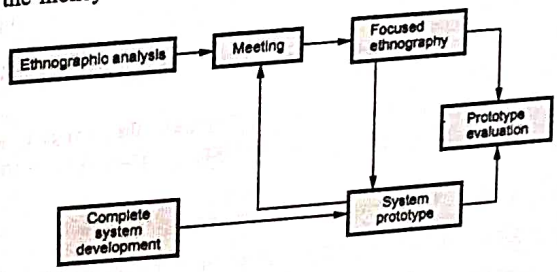


Fig. Q.19.1 Ethnographic process

2. Requirements obtained from inter-activities performed by the people

Sometimes for finding the social requirements the other people's activities should be known. For example in ATM system the operator can not shutdown the system if some transaction is in processing.

2.7 : Requirements Validation

Q.20 What do you understand with validating requirements ?

☞[SPPU : Dec.-16 (End Sem), Marks 5]

Ans. : *Requirement validation is a process in which it is checked that whether the gathered requirements represent the same system that customer really wants.*

- In requirement validation the requirement errors are fixed. Requirement error costs are high so validation is very important.

Requirements validation techniques

1. **Requirements reviews :** Requirement review is a systematic manual analysis of the requirements.
   - The requirement review should be taken only after formulation of requirement definition. And both the customer and contractor staff should be involved in reviews.
   - Reviews may be formal (with completed documents) or informal.
   - Good communications should take place between developers, customers and users. Such a healthy communication helps to resolve problems at an early stage.
2. **Prototyping :** The requirements can be checked using executable model of system.
3. **Test-case generation :** In this technique, the various tests are developed for requirements. The requirement check can be carried out with
   - **Verifiability :** Is the requirement realistically testable ?
   - **Comprehensibility :** Is the requirement properly understood ?
   - **Traceability :** Is the origin of the requirement clearly stated ?
   - **Adaptability :** Can the requirement be changed without a large impact on other requirements ?

2.8 : Requirement Management

Q.21 What is requirement management ?

Ans. : Requirement management is a process of managing the changing the requirements during requirements engineering process and system development.

Q.22 What are enduring and volatile Requirements ?

Ans. : Eduring requirements : These are the stable requirements that are derived from the core activity of the organization. These requirements are dependent upon the application domain of the software. For example : For banking system, transfer of money from one account to another is the enduring requirement.

Volatile requirements : For certain requirements if there is a possibility that those requirements may get changed during the development stage or after the system becomes operational, then such requirements are called volatile requirements.

Q.23 Draw and explain the traceability table for requirement management.

☞ [SPPU : May-14, Marks 6]

Ans. : Traceability is concerned with relationship between requirements their sources and the system design.

Traceability information is typically represented by a data structure traceability matrix. If one requirement is dependent upon the other requirement then in that row-column cell 'D' is mentioned and if there is a weak relationship between the requirements then corresponding entry can be denoted by 'R'.

For example

Requirement ID	A	B	C	D	E	F
A		D			R	
B			D			
C				R		R
D			D			
E						
F	R			D		

For mentioning the traceability of small systems usually the traceability

Q.24 Explain about requirements change management in detail.

Ans. : The requirement change management is a technique that can be applied to the processes in which requirements may get changed.

The requirement change management process can be applied in three stages.

1. Problem analysis and change specification
2. Change analysis and costing
3. Change implementation

Fig. Q.24.1 shows the stages of requirement change management process.

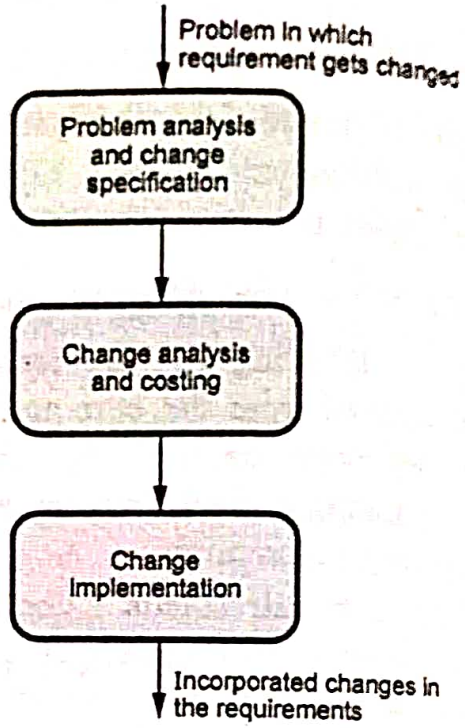


Fig. Q.24.1 Requirement change management

1. Problem analysis and change specification : When requirement change request is made for some particular problem then the problem with older requirement is mentioned or sometimes simply change specification is given. Then first of all, problem analysis or change specification is analyzed in order to **validate** the required change. If necessary the feedback of this analysis is given to the person who is demanding such change.

2. Change analysis and costing : Following actions are carried out in this stage :

i) The effect of change is assessed using traceability information.

ii) The cost of such change is estimated.

iii) After getting the cost of changes the decision is made on whether to go for implementation of these changes or not.

3. Change implementation : Once it is decided to implement the proposed changes in the requirement, the requirement document has to be modified. The requirement document has to either re-written or re-organised. This can be achieved by making the modularity in the requirement specifications, so that it becomes easy to change individual section without affecting other part of requirement document.

END...