SEPM CAT 1 ANSWERS

A) Explain common process framework for software engineering in detail.

① Software project tracking and control :→
To access the progress of software development.

② Risk management :→
To analyse /access the list during the software development

③ Software Quality Assurance :→
To ensure Software quality

④ Technical review :→
To remove the technical issues (connection issues, errors and etc..).

⑤ Measurement :→
To analyse the process, product bazzels, needs of customer.

⑥ Software configuration management :→
It manages the effect of configuration change.

⑦ Reusability management :→
It defines the product's reuse (platform independent)

⑧ Work product preparation :→
It deals with creating models, documents, list, forms etc.

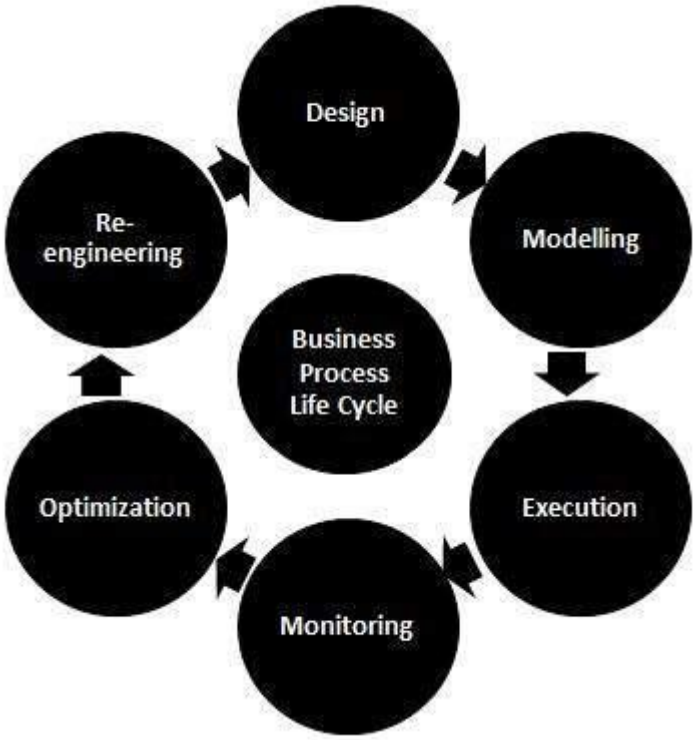
B) Explain Business Process Engineering in detail.

A business process is a collection of activities or tasks that produce a specific service or product for end users. It is usually represented as a flowchart as a sequence of activities that points to a Process Matrix.

Business process Modelling is carried out by Process owners or product owners to enable the test team to test efficiently. It aims to improve business performance by optimising the efficiency of the associated activities of a product or service.

Business Process Life Cycle:

Business Process Life cycle has various phases as listed below:



Business Process Testing [BPT]:

It is a tool used for an automated and manual testing for designing tests, maintaining tests and executing tests. The reusable tests are usually designed by Business Analysts for improving test efficiency.

Benefits and Features of BPT :

- Allows non-technical subject matter expertise to quickly build a reusable test workflow.
- It reduces the effort required for test maintenance.
- It converts the manual tests to manual test components.
- It provides a framework to build User Acceptance Tests to meet the requirements.

A)Discuss " Software Engineering - A Layered Technology".

Software engineering is a fully layered technology, to develop software we need to go from one layer to another. All the layers are connected and each layer demands the fulfillment of the previous layer.

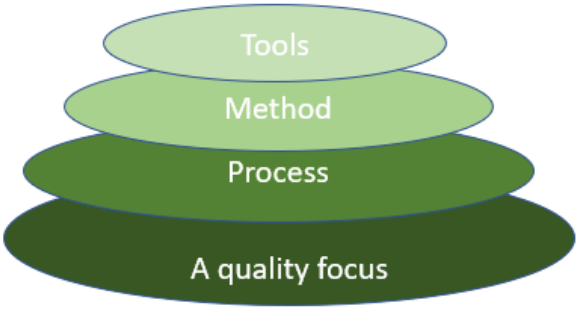
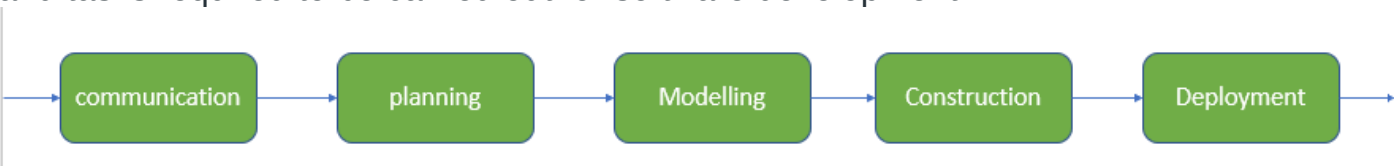


Fig: The diagram shows the layers of software development

Layered technology is divided into four parts:

1. A quality focus: It defines the continuous process improvement principles of software. It provides integrity that means providing security to the software so that data can be accessed by only an authorized person, no outsider can access the data. It also focuses on maintainability and usability.

2. Process: It is the foundation or base layer of software engineering. It is key that binds all the layers together which enables the development of software before the deadline or on time. Process defines a framework that must be established for the effective delivery of software engineering technology. The software process covers all the activities, actions, and tasks required to be carried out for software development.



Process activities are listed below:-

- **Communication:** It is the first and foremost thing for the development of software. Communication is necessary to know the actual demand of the client.
- **Planning:** It basically means drawing a map for reduced the complication of development.
- **Modeling:** In this process, a model is created according to the client for better understanding.
- **Construction:** It includes the coding and testing of the problem.
- **Deployment:-** It includes the delivery of software to the client for evaluation and feedback.

3. Method: During the process of software development the answers to all "how-to-do" questions are given by method. It has the information of all the tasks which includes communication, requirement analysis, design modeling, program construction, testing, and support.

4. Tools: Software engineering tools provide a self-operating system for processes and methods. Tools are integrated which means information created by one tool can be used by another.

B)What is SRS? Explain in detail.

A software requirements specification (SRS) is **a document that describes what the software will do and how it will be expected to perform**. It also describes the functionality the product needs to fulfill all stakeholders (business, users) needs.

Qualities of SRS:

- Correct
- Unambiguous
- Complete
- Consistent
- Ranked for importance and/or stability
- Verifiable
- Modifiable
- Traceable

Types of Requirements:

The below diagram depicts the various types of requirements that are captured during SRS.

A)What is an Agile process? What are different methods of Agile?State its principles.

Agile is **an iterative approach to project management and software development that helps teams deliver value to their customers faster and with fewer headaches**. Instead of betting everything on a "big bang" launch, an agile team delivers work in small, but consumable, increments.

The most widely used Agile methods include:

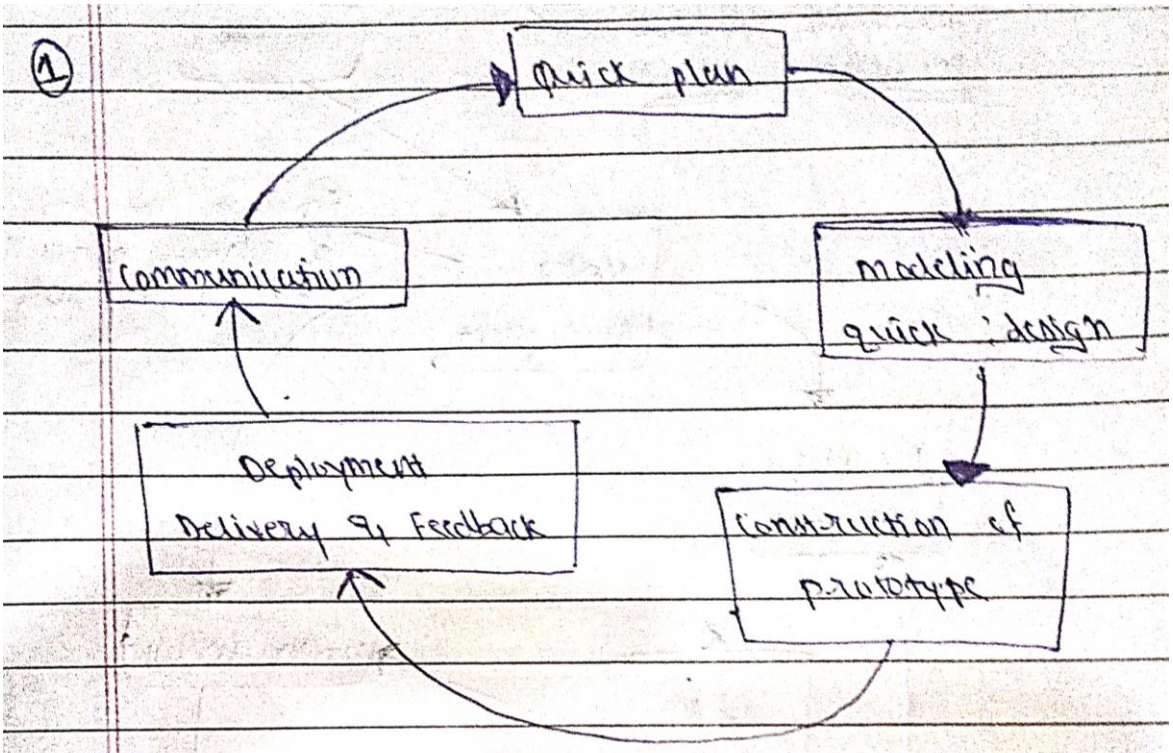
- Scrum
- Lean software development
- Extreme programming
- Crystal
- Kanban
- Dynamic systems development method
- Feature-driven development

Principles:

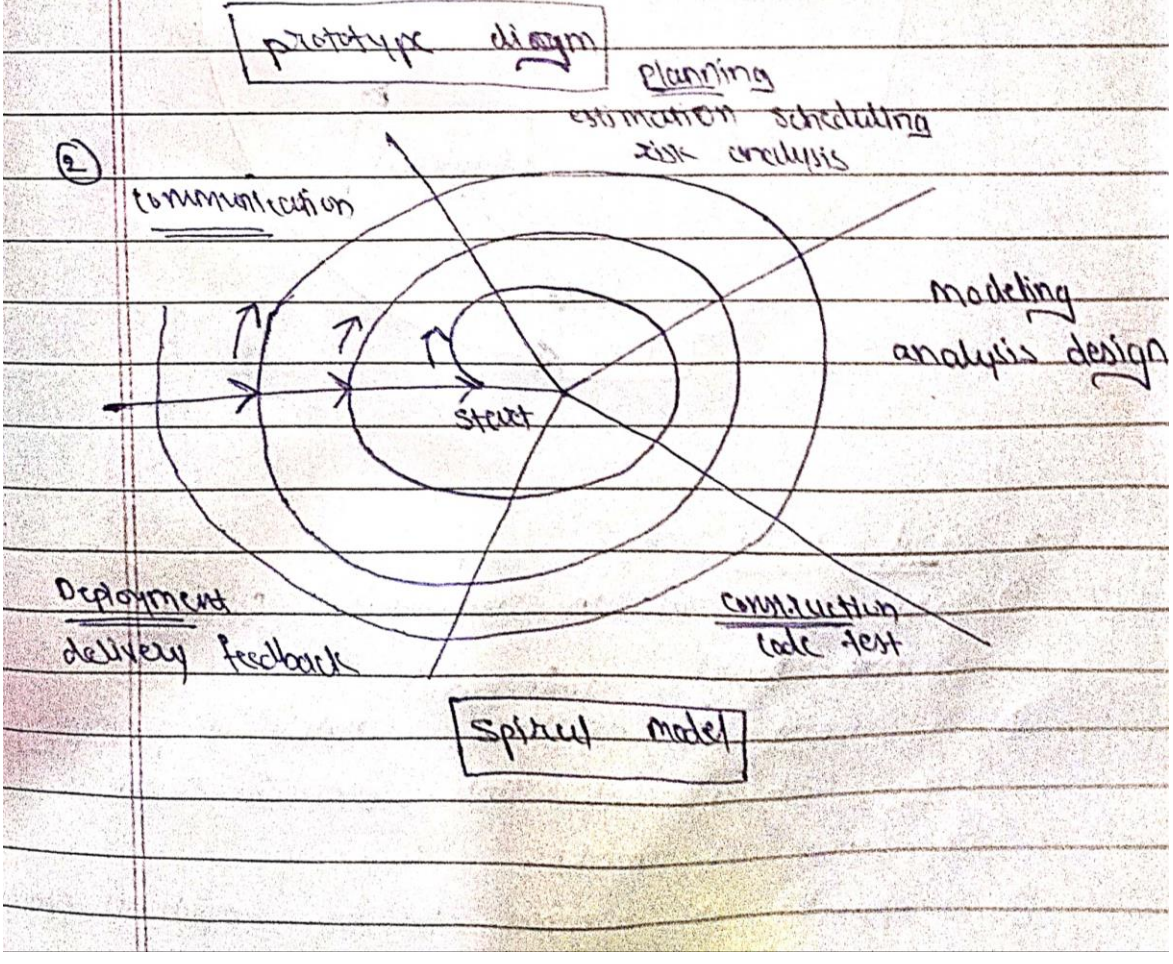
1. Highest priority is to satisfy the customer through early and continuous delivery of valuable software.
2. It welcomes changing requirements, even late in development.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shortest timescale.
4. Build projects around motivated individuals. Give them the environment and the support they need, and trust them to get the job done.
5. Working software is the primary measure of progress.
6. Simplicity the art of maximizing the amount of work not done is essential.
7. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

B)Explain System Engineering hierarchy in detail.

A)Explain spiral model as a combination of life cycle model and prototyping model.



prototype diagm



Spiral model

* Spiral model :→

(step by step)

(1) Spiral model is an evolutionary software process model.

(logic based)

(2) It consist of iterative nature of prototyping along with a control and systematic aspect of waterfall model.

(3) It provides the potential for the rapid development

of increasingly more complex version of the Software.

ie. used to make heavy software (may consist of more mathematics.)

(4) Using the spiral model the SW is developed in the series of evolutionary releases.

(5) During the early stage the release might be a model or prototype.

(6) During the later stage increasingly more complex versions of engineered system are produced.

(7) The spiral model is divided into set of framework activities defined by SW engineering team.

(8) As the evolutionary process begins the SW team performs an activity implied by the circle, Spiral in the clockwise direction beginning at the center.

(9) The first circuit around the spiral might presented in development of product Specification, Subsitituents passes around the Spiral might be used to develop a prototype, the more (updated) Suffisticated version of this software

(10) The pass through the planning region result in adjustment to the project plan

(11) cost and schedule are adjusted based upon the the feedback from the customer.

(12) The project manager adjust the planned number of iteration required to complete the software

(13) The spiral model can be adapted to apply throughout the life of computer s/w

B)What is FAST?Explain it in detail.

Facilitated Application Specification Technique ("FAST")
- It is a technique for requirements elicitation for software development.
- The objective is to close the gap between what the developers intend and what users expect.
- It is a team-oriented approach for gathering requirements.

It's objective is to bridge the expectation gap – difference between what the developers think they are supposed to build and what customers think they are going to get.

Basic guidelines
1. Meetings are conducted at a neutral site attended by both developers and users.
2. The group establishes rules for preparation and participation.
3. An agenda is suggested that with enough formality to cover all important points but informal enough to encourage the free flow of ideas.
4. A facilitator controls the meeting.
5. A definition mechanism is used.

- The main goal is to identify the problem, propose solutions, negotiate different approaches, and specify a preliminary set of software requirements in an atmosphere that is conductive to accomplish the goal.

- After initial meeting, user and developer should write a one or two product request form. Before the next meeting it is distributed to all other attendees. Each attendee is asked to make the following lists:

1. List of objects
2. List of services
3. List of constraints
4. performance criteria

Representatives of FAST
1. Marketing person
2. Software and hardware engineer
3. Representative from manufacturing
4. An outside faciitator

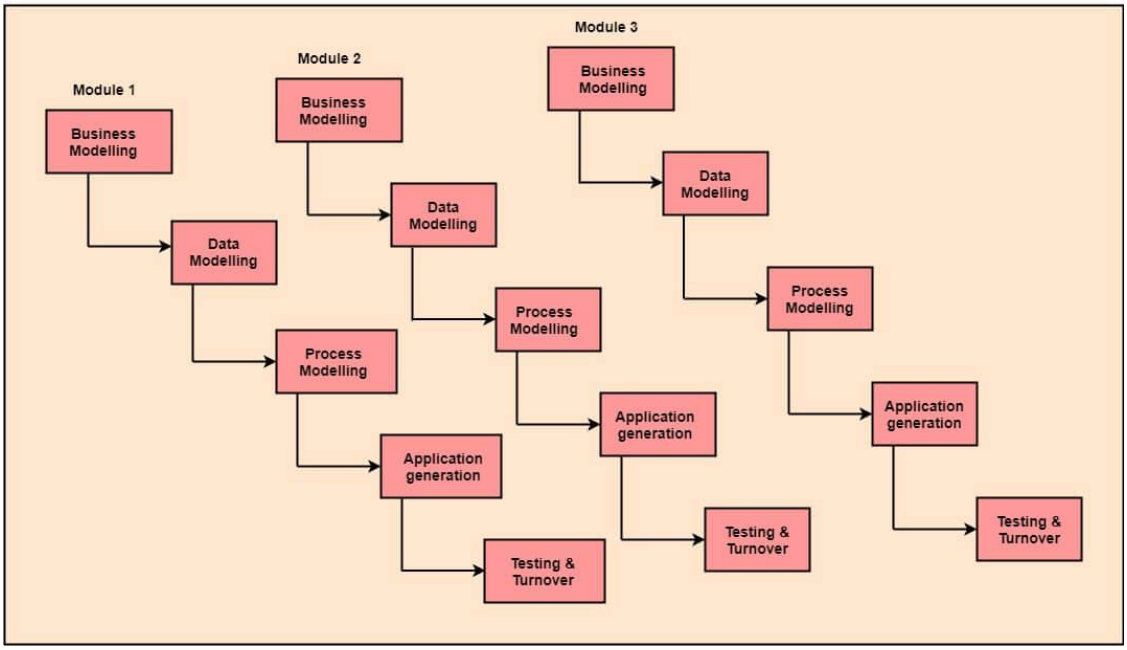
A)Write a short note on:RAD Model.

RAD is a linear sequential software development process model that emphasizes a concise development cycle using an element based construction approach. If the requirements are well understood and described, and the project scope is a constraint, the RAD process enables a development team to create a fully functional system within a concise time period.

RAD (Rapid Application Development) is a concept that products can be developed faster and of higher quality through:

- Gathering requirements using workshops or focus groups
- Prototyping and early, reiterative user testing of designs
- The re-use of software components
- A rigidly paced schedule that refers design improvements to the next product version
- Less formality in reviews and other team communication

Fig: RAD Model



1.Business Modelling: The information flow among business functions is defined by answering questions like what data drives the business process, what data is generated, who generates it, where does the information go, who process it and so on.

2. Data Modelling: The data collected from business modeling is refined into a set of data objects (entities) that are needed to support the business. The attributes (character of each entity) are identified, and the relation between these data objects (entities) is defined.

3. Process Modelling: The information object defined in the data modeling phase are transformed to achieve the data flow necessary to implement a business function. Processing descriptions are created for adding, modifying, deleting, or retrieving a data object.

4. Application Generation: Automated tools are used to facilitate construction of the software; even they use the 4th GL techniques.

5. Testing & Turnover: Many of the programming components have already been tested since RAD emphasis reuse. This reduces the overall testing time. But the new part must be tested, and all interfaces must be fully exercised.

Advantage of RAD Model

- This model is flexible for change.
- In this model, changes are adoptable.
- Each phase in RAD brings highest priority functionality to the customer.
- It reduced development time.
- It increases the reusability of features.

Disadvantage of RAD Model

- It required highly skilled designers.
- All application is not compatible with RAD.
- For smaller projects, we cannot use the RAD model.
- On the high technical risk, it's not suitable.
- Required user involvement.

B)Explain Requirement Engineering process in detail.

Requirement engineering is a process that is performed in the initial stages of any software development. It includes analyzing the customer's requirement and various tasks such as:

- At first, identify the user's requirement, what does it want the operational system to perform?
- Analyze the requirements gathered from the customer.
- Evaluate the feasibility of the operational system.
- Propose some unambiguous solutions.
- Validate the requirement provided by the customer.
- Manage the requirements as far as they are modelled into an operational system.

Importance of Requirement Engineering

Requirement engineering is an important stage in any software development. It is also believed if the requirement engineering is done appropriately then we can expect that the final software developed doesn't lag behind in terms of design or functionality leading to successful and profitable software.

1. The requirement engineering provides a vision of the final software i.e. what the software would do? This creates a sense of mutual understanding between the customer and the software developer.
2. Requirement engineering also helps in defining the scope of the software i.e. what will be the functionalities of the final software.
3. It also helps in perceiving the cost of the final software.
4. It also helps in perceiving the schedule up to which the software will be delivered to the customer.

Qualities of Requirements

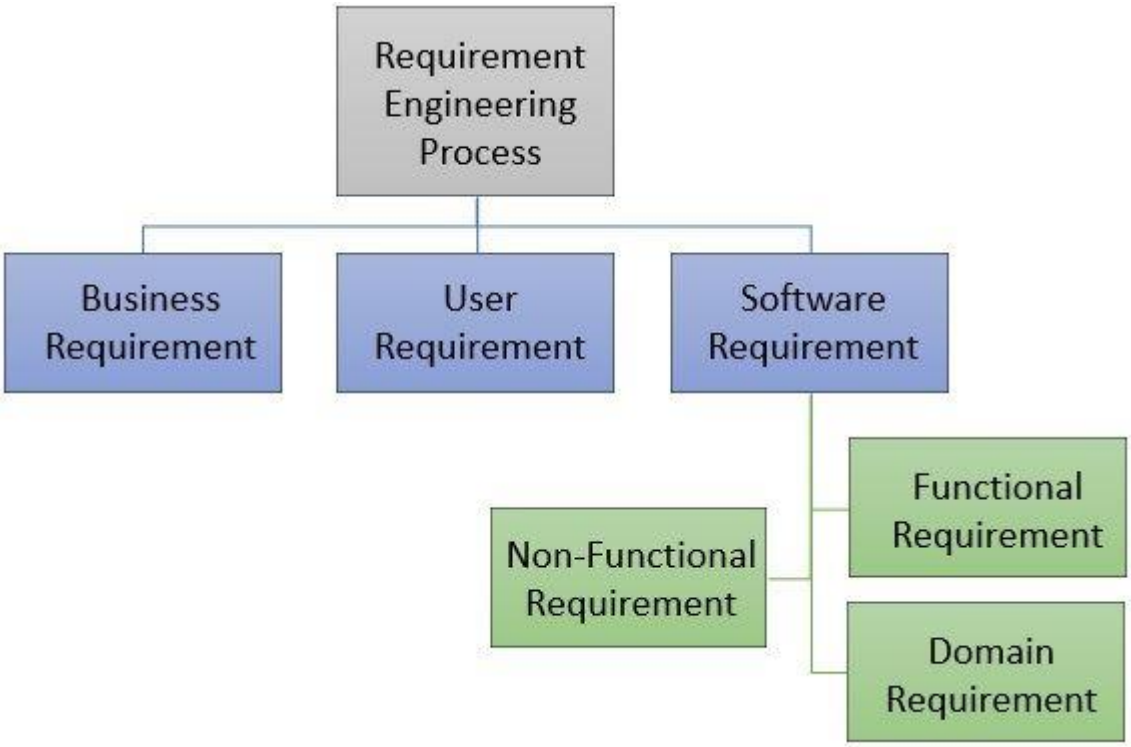
The requirements collected from the customer must be precise and must convey the customer need to the software developer. Quality requirements possess the following features:

- **Complete:** The specified requirements must be complete, there must be nothing missing that is important for the development of the software.
- **Consistent:** If the software requirements are provided by more than one customer. Then the software engineer must ensure that the requirements provided by an individual must not contradict the requirements provided by another customer.
- **Unambiguous:** Each specified requirement must be unambiguous i.e. it must specify a single meaning.
- **Functions:** The requirement must specify what functions or computations must the software perform and not how they must be implemented.

- **Concise:** Each requirement must be specified only once and there should not be duplication of any requirement statement.
- **Minimum:** The specified requirements must not carry unnecessary details.
- **Understandable:** The specified requirements must be understandable by both customer and software developer.
- **Achievable:** The requirement must be technically feasible.
- **Testable:** It can be verified that the requirements are collected completely.

Types of Requirements

Requirements can be classified into the following ways:



1. Business Requirements

Business requirements specify the software's business demand. The business requirement identifies why the software is required, who will be the end-users of the software, how the software will benefit its end users.

The business requirement does specify the technicality of the software i.e. how it should be implemented it focuses only on what software must do for them.

2. Users Requirements

The user requirements specify the need and expectations of the end-users of the software. Although the user's requirement is sometimes incorporated with the business requirements. But sometimes requirements of software with more complex functionality or more complex user interface must be documented separately.

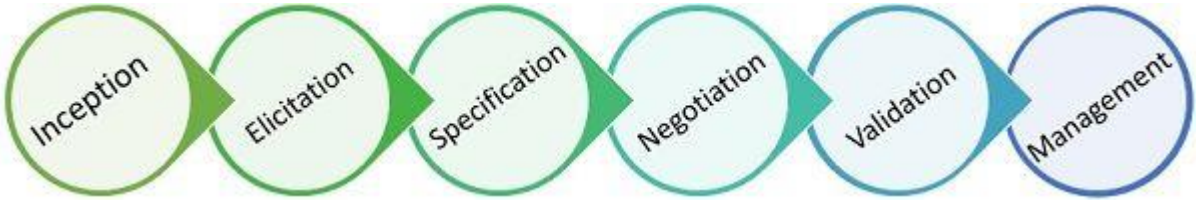
3. Software Requirements

The software requirements specify what the software must do and it define how the software is expected to perform. Type of software requirement are as follow:

- **Functional**: The functional requirements describe what the software must deliver and what it must not. How the software must respond to incorrect inputs. Thus, the functional requirement describes the behaviour of the software.
- **Non-Functional**: The non-functional requirements describe the non-behavioural aspects of the system such as its scalability, reliability, performance, security, its portability, reusability and flexibility.
- **Domain Requirements**: The domain requirement describes the realm, area, group for which the software is to be developed. Such as for college, office, military, hospital, students, teachers, patients etc.

Requirement Engineering Process

The requirement engineering process involver certain steps that must be followed to collect the entire specifications about the project.



1. Inception

In the inception phase we how the concept of the software has evolved. Does there was any catalyst event that triggered the need for the software or its need has evolved over time? Although there is no precise answer to this question the conversation starts with these basic questions.

In the inception phase, the business requirements are identified where first the business need is identified, the scope of the software in the market is analysed, a rough feasibility analysis is done and the functioning of the software is discussed. Though all these requirements are subject to change they are enough to start a conversation between business analysts or customers or stakeholders and the software developers.

2. Requirement Elicitation

If the inception report is positive to undertake the project the next step is to gather the requirement from the clients. This phase is the requirement elicitation phase where the system analyst or the software developers communicate with the clients or the end-users of the system to elicit more information.

There are many things that the software analyst come across while eliciting the requirements such as:

- The problem statement defined by the client or the end-user may have ill-defined boundaries or the technical details provided by them might confuse the analyst or the developer regarding the objective of the software to be developed.
- Even the clients or the end-user of the software are not sure about what they want. They are unaware of the potential of their computing environment and even find it difficult to convey the problem statement to the software engineer.
- In some cases, the client or end-user misses the most important information to convey to the software engineer or provide ambiguous requirements.
- Sometimes the requirements of the clients or users change with time which raises confusion among the software developers.

3. Requirement Specification

In terms of the software, the requirement specification can be a written document, a mathematical model, graphical models, some usage scenarios, some sort of sample model, some sort of code etc.

4. Negotiating Requirements

Now each client associated with the same project might have different versions of requirements and they think that their aspect is more useful to develop software. Observing the conflicting requirements, the software engineer has to reconcile and negotiate the facts with the clients and end-users.

Negotiation is an iterative process where each client is asked to rank the requirement in terms of priority. Parallelly the software engineer also assesses the cost associated with the project, risk factors.

While addressing the conflicting requirements they may eliminate, add, combine or modify the conflicting requirements so that each client sense some sort of satisfaction that their requirements are essential for the development of the software.

5. Requirement Validation

Whatever is gathered, understood during the process of the requirement of engineering must be validated to ensure that all the software requirements are specified. The software developer or engineer must also ensure that the requirements are unambiguous, correct, consistent, essential.

The requirement validation phase requires the presence of all like software developers, clients, end-users and stakeholders. The specified requirements are checked thoroughly for any missing, redundant, inconsistent information.

6. Requirement Management

Requirements of any software keep on changing and the business and technical change throughout the life of the software. Requirement management is subject to keep track of the requirements and the changes that occur during the development of the software.

So, these are the steps or tasks that must be performed in the requirement engineering process. The efficient requirement engineering promises that developed software satisfies each requirement specified and up to the standards.

OR

Requirement Engineering is the process of defining, documenting and maintaining the requirements. It is a process of gathering and defining service provided by the system. Requirements Engineering Process consists of the following main activities:

- Requirements elicitation
- Requirements specification
- Requirements verification and validation
- Requirements management
-

Requirements Elicitation:
It is related to the various ways used to gain knowledge about the project domain and requirements. The various sources of domain knowledge include customers, business manuals, the existing software of same type, standards and other stakeholders of the project.
The techniques used for requirements elicitation include interviews, brainstorming, task analysis, Delphi technique, prototyping, etc. Some of these are discussed here. Elicitation does not produce formal models of the requirements understood. Instead, it widens the domain knowledge of the analyst and thus helps in providing input to the next stage.

Requirements specification:
This activity is used to produce formal software requirement models. All the requirements including the functional as well as the non-functional requirements and the constraints are specified by these models in totality. During specification, more knowledge about the problem may be required which can again trigger the elicitation process.
The models used at this stage include ER diagrams, data flow diagrams(DFDs), function decomposition diagrams(FDDs), data dictionaries, etc.

Requirements verification and validation:
Verification: It refers to the set of tasks that ensures that the software correctly implements a specific function.
Validation: It refers to a different set of tasks that ensures that the software that has been built is traceable to customer requirements.

If requirements are not validated, errors in the requirement definitions would propagate to the successive stages resulting in a lot of modification and rework. The main steps for this process include:

- The requirements should be consistent with all the other requirements i.e no two requirements should conflict with each other.
- The requirements should be complete in every sense.
- The requirements should be practically achievable.

Reviews, buddy checks, making test cases, etc. are some of the methods used for this.

Requirements management:

Requirement management is the process of analyzing, documenting, tracking, prioritizing and agreeing on the requirement and controlling the communication to relevant stakeholders. This stage takes care of the changing nature of requirements. It should be ensured that the SRS is as modifiable as possible so as to incorporate changes in requirements specified by the end users at later stages too. Being able to modify the software as per requirements in a systematic and controlled manner is an extremely important part of the requirements engineering process.

A) Explain generic view of Software Engineering.

The process of a software development has three Generic views which are:

1. **Definition Phase** - It is the base of Definition phase. The experts get the knowledge about "What".

    - Information needed for processing.
    - Which functions are required.
    - Expectations about the capacity.
    - Interface which is established.
    - Area of the validation.

    This phase defines all the expectations depending on the standard of the software Engineering. It contains three steps.

    - Analysis of system
    - Planning of project
    - Requirement Analysis

2. **Development phase** - Focus point of development phase is "How". After the explanation of "What" it turn to "How". Various type of question raised in developer mind that how to design the data structure and Architecture of software, Procedural detail how to implemented and how design convert in a programming language and testing of software how to perform. Three special steps always taken in this phase which are

    - Design of software
    - Coding
    - testing of software system

3. **Maintenance phase** - The main focus of maintenance phase is change which cause is correction of errors, adaption of new idea, According to the needs of software after change in customer mood.

B) Discuss Communication practices in detail.

Before customer requirements can be analyzed, modeled, or specified they must be gathered through a communication (also called requirements elicitation) activity. A customer has a problem that may be amenable to a computer- based solution. A developer responds to the customer's request for help. Communication has begun. But the road from communication to understanding is often full of potholes.

Effective communication (among technical peers, with the customer and other stakeholders, and with project managers) is among the most challenging activities that confront software engineer. In this context, we discuss communication principles that apply equally to all forms of communication that occur within a software project.

Principle #1: Listen.
Principle #2: Prepare before you communicate.
Principle #3: Someone should facilitate the activity.
Principle #4: Face–to-face communication is best.
Principle #5: Take notes and documentation decisions:
Principle #6: Stay focused, modularize your discussion.
Principle #7: If something is unclear, draw a picture.
Principle #8: (a) Once you agree to something, move on; (b) If you can't agree to something, move on; (c) If a feature or function is unclear and cannot be clarified at the moment move on.

Principle #9: Negotiation is not a contest or a game. It works best when both parties win.

A)Explain Unified process model in detail.

B)Discuss Planning practices in detail.

The planning activity defines a set of management and technical practices that enable software team to define road map for travel to word it is strategic goal and objective. Planning provides guideline for software team to progress very fast in the project development.

Principle #1: Understand the scope of the project. It's impossible to use a road map if you don't know where you're going. Scope provides the software.

Principle #2: Involve the customer in planning activity. The customer defines priorities and establishes the project constraints.

Principle #3: Recognize that planning is iterative. As work begins, it is very likely that things will change. As a consequence, the plan must be adjusted to accommodate these changes. In addition, iterative and incremental process models dictate re-planning based on feedback received from users.

Principle #4: Estimate based on what you know. The intent of estimation is to provide an indication of effort, cost, and task duration, based on the team's current understanding of the work to be done.

Principle #5: Consider risk as you define the plan. If the team has defined risks that have high impact and high probability, contingency planning is necessary.

Principle #6: Be realistic. People don't work 100 percent every day. Noise always enters into any human communication. Omission and ambiguity are facts of life. Change will occur. Even the best software engineers make mistakes. These and other realities should be considered as a project plan is established.

Principle #7: Adjust granularity as you define the plan. Granularity refers to the level of detail that is introduced as a project plan is developed. A "fine granularity" plan provides significant work detail that is planned over relatively short time increments.

Principle #8: Define how you intend to ensure quality. The plan should identify how the software team intends to ensure quality. If formal technical reviews are to be conducted, they should be scheduled.

Principle #9: Describe how you intend to accommodate change. Even the best planning can be obviated by uncontrolled change. The software team should identify how changes are to be accommodated as software engineering work proceeds.

Principle #10: Track the plan frequently and make adjustments are required. Software project falls behind schedule one day at a time. Therefore, it makes sense to track progress on a daily

basis, looking for a problem areas and situation in which scheduled work does not confirm to actual work conducted. When slippage is encountered, the plan is adjusted accordingly.

A)Explain Specialized process model in detail.

B)Discuss modeling practices in detail.

The models are created to gain better understanding of actual entity to be built. When the entity is a physical thing, we can build a model that is identical in form of shape but smaller in scale. However, when the entity is software, our model must take a different form. It must be capable of representing the information that software transforms, the architecture and functions that enable the transformation to occur, the features that user's desire, and the behavior of the system as the transformation is taking place.

Two classes of models are crated: Analysis models and design models. Analysis models represent the customer requirements by depicting the software in three different domains: the information domain, the functional domain, and the behavioral domain. Design models represent characteristics of the software that help practitioners to construct it effectively.

Analysis Modeling Principles

A large number of analysis modeling methods have been developed. Each analysis methods has unique point of view. However, all analysis methods are related by a set of operational principles.

Principle #1: The information domain of a problem must be represented and understood. The information domain compasses the data that flow into the system and the data stores that collect and organize persistent data objects.

Principle #2: The functions that the software performs must be defined. Software functions provide direct benefit to visible end-user. Some functions transform data that flow into the system; in other cases, functions effect some level of control over internal software processing or external system elements.

Principle #3: The behavior of the software must be represented. The behavior of computer software is driven by its interaction with the external environment. Input provided by end-users, control data provided by an external system, or monitoring data collected over a network all cause the software to behave in a specific way.

Principle #4: The models that depict information, function, and behavior must be partitioned in a manner that uncovers detail in a layered fashion. Analysis modeling is the first step in software engineering problem solving. It allows the practitioner to understand the problem better and establishes a basis for the solution (design). Complex problems are difficult to solve in their entirety. For this reason, we use a divide and conquer strategy. A large, complex problem is divided into sub-problems until each sub- problem is relatively easy to understand. This concept is called partitioning, and it is a key strategy in analysis modeling.

Principle #5: The analysis task should move from essential information toward implementation detail. Analysis modeling begins by describing the problem from the end-user's perspective. The "essence" of a problem is described without any consideration of how a solution will be implemented.

Design Modeling Principles

The design model created for software provides a variety of different views of system. There is no shortage of methods for deriving various elements of a software design. Some methods are data-driven, allowing the data structure to dictate the program architecture and the resultant processing component. Others are pattern-driven, using information about the problem domain (the analysis model) to develop architectural styles and processing patterns- a set of design principles that can be applied regardless of the method that is used.

Principle #1: Design should be traceable to the analysis model. The analysis model describes the information domain of the problem, uses visible functions, system behavior, and a set of analysis classes that package business objects with the methods that service them. The design model translates this information into an architecture, a set of subsystems that implement major functions, and a set of component- level designs that realize analysis classes.

Principle #2: Always consider the architecture of the system to be built. Software architecture is the skeleton of the system to be built. It affects interfaces, data structures, program control flow behavior, the manner in which testing can be conducted and the maintainability of resultant system.

Principle #3: Design of data is as important as design of processing functions. Data design is an essential element of architectural design. The manner in which data objects are realized within the design cannot be left to chance. A well-structured data design helps to simplify program flow, makes design and implementation of software components easier, and makes overall processing more efficient.

Principle #4: Interfaces (both internal and external) must be designed with care. The manner in which data flow between the components of a system has much to do with processing efficiency, error propagation, and design simplicity, A well designed interface integration easier and assists the tester in validating components functions.

Principle #5: User interface design should be tuned the needs of the end-user. However, in every case, it should be stress free and easy to use. The user interface is the visible manifestation of the software. A poor interface design often leads to the perception that the software is "bad".

Principle #6: Components should be functionally independent. Functional independence is a

measure of the “single- mindedness” of a software component. The functionally that is delivered by a component should be cohesive- that is, it should focus on one and only on function or sub-function.

Principle #7: Components should be loosely coupled to one another and to the external environment. Coupling is achieved in many ways-via component inter-face, by messaging and through global data. As the level of coupling increases, the likelihood of error propagation also increases and the overall maintainability of the software decreases. Therefore, component coupling should be kept as low as is reasonably possible.

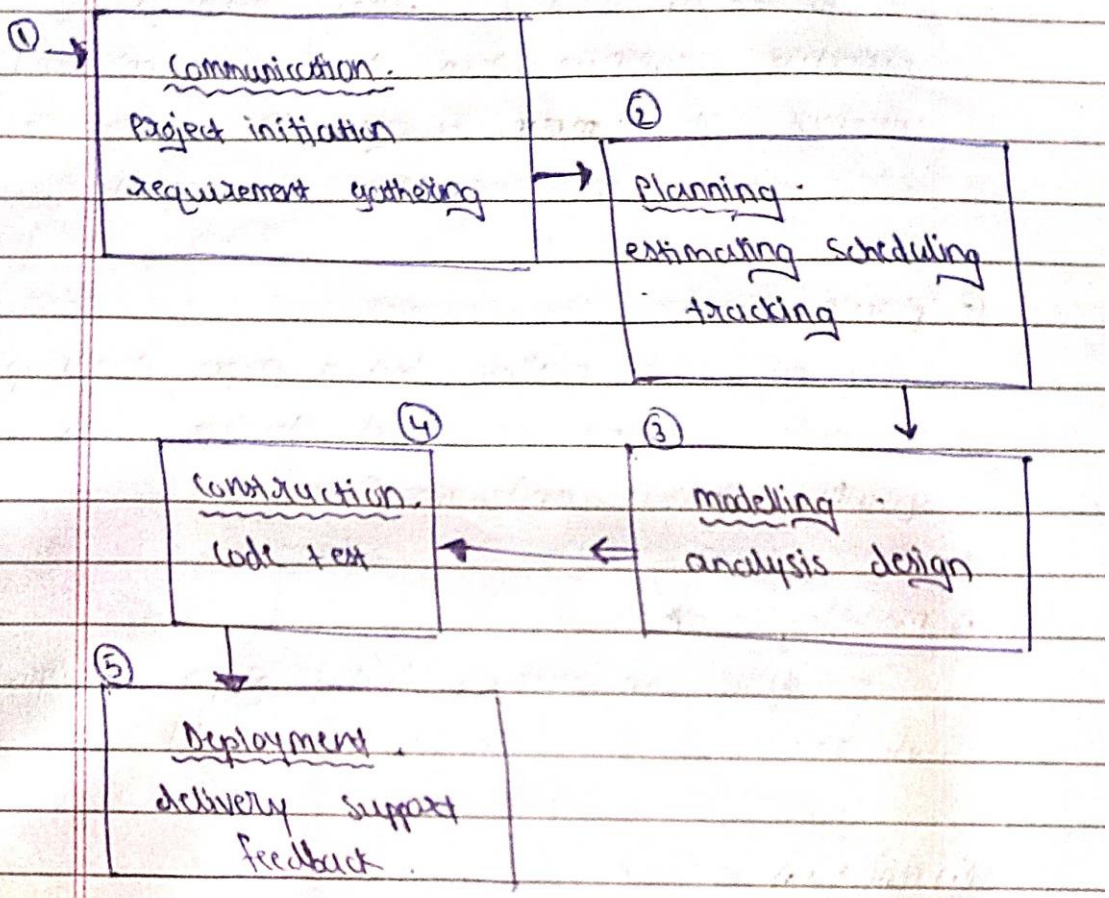
Principle #8: Design representation(model) should be easily understandable. The purpose of design is to communicate information to practitioners who will generate code, to those who will test the software, and others who may maintain the software in the future. If the design is difficult to understand, it will not serve as an effective communication medium.

Principle #9: The design should be developed iteratively. With each iteration, the designer should strive for greater simplicity. Like most of the creative activities, design occurs iteratively. The first iteration works to refine the design and correct errors, but later iterations should strive to make the design as simple as is possible.

A) Discuss Waterfall model with its limitations.

Imp

* The waterfall model :→



To develop a waterfall model, we have to follow certain steps :→

1. communication
2. planning
3. modeling
4. construction
5. Deployment.

Classic model = old model

DATE : ___/___/___ PAGE No : _____

(1) During the software development there is a situation when the requirement for problem are well understood when the work for communication to the deployment

(2) waterfall model is called classic life-cycle model
↳ paramparik (parivar se chalte aarahe)

(3) It suggest systematic sequential approach. It starts from customer's specification for requirement gathering, modeling, planning, construction, deployment.

(4) The variation in the above representation is called as 'V'-model. The model the relationship of assurance to the actions, communication, modeling and early construction activity. The waterfall model is oldest model :-

→

(A) ~~Disadvantage~~ The real projects based upon waterfall model may follow sequential flow, it may sometimes cause diversions as the project team needs.

(B) It may be difficult for customer to give all the requirements.

(C) The customer should also have patience, a working version of program will not be available till it is tested and made error free.

(D) Today, a software work is fast paced

it subject to never ending stream of changes.

(F) The waterfall model is often inappropriate as it supports linear flow of software development. It served as useful process model where the situation & requirement are fixed.

B) Define all requirement engineering tasks in detail.

- The process of collecting the software requirement from the client then understand, evaluate and document it is called as requirement engineering.
- Requirement engineering constructs a bridge for design and construction.

Requirement engineering consists of seven different tasks as follow:

1. Inception

- Inception is a task where the requirement engineering asks a set of questions to establish a software process.
- In this task, it understands the problem and evaluates with the proper solution.
- It collaborates with the relationship between the customer and the developer.
- The developer and customer decide the overall scope and the nature of the question.

2. Elicitation

Elicitation means to find the requirements from anybody.
The requirements are difficult because the **following problems occur in elicitation**.

Problem of scope: The customer give the unnecessary technical detail rather than clarity of the overall system objective.

Problem of understanding: Poor understanding between the customer and the developer regarding various aspect of the project like capability, limitation of the computing environment.

Problem of volatility: In this problem, the requirements change from time to time and it is difficult while developing the project.

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.
- To create rough guesses of development and access the impact of the requirement on the project cost and delivery time.

5. Specification

- In this task, the requirement engineer constructs a final work product.
- The work product is in the form of software requirement specification.
- In this task, formalize the requirement of the proposed software such as informative, functional and behavioral.
- The requirement are formalize in both graphical and textual formats.

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.
- The formal technical reviews from the software engineer, customer and other stakeholders helps for the primary requirements validation mechanism.

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.
- These tasks start with the identification and assign a unique identifier to each of the requirement.
- After finalizing the requirement traceability table is developed.
- The examples of traceability table are the features, sources, dependencies, subsystems and interface of the requirement.