S.E. SEM4 Q&A

1. Explain the SDLC process

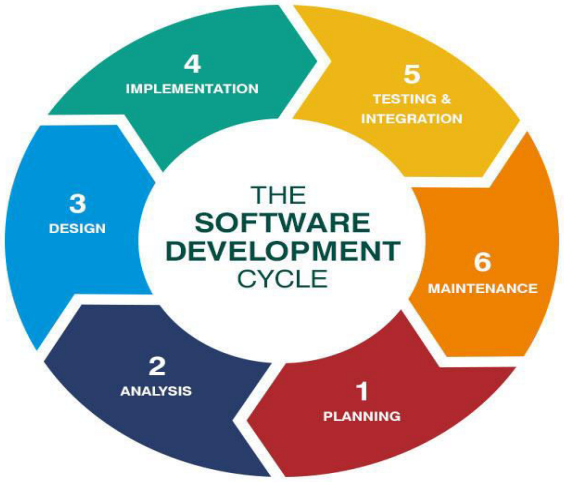
Software Development Life Cycle (SDLC)

- Software Development Life Cycle (SDLC) is a framework that defines the steps involved in the development of software at each phase. It covers the detailed plan for building, deploying and maintaining the software.
- SDLC defines the complete cycle of development i.e., all the tasks involved in planning, creating, testing, and deploying a Software Product. Every phase of the SDLC life Cycle has its own process and deliverables that feed into the next phase.

SDLC Phases

- Phase 1: Requirement collection and analysis
- Phase 2: Feasibility study
- Phase 3: Design

- Phase 4: Coding
- Phase 5: Testing
- Phase 6: Installation/Deployment
- Phase 7: Maintenance



Phase 1: Requirement collection and analysis

The requirement is the first stage in the SDLC process. It is conducted by the senior team members with inputs from all the stakeholders and domain experts in the industry. Planning for the quality assurance requirements and recognition of the risks involved is also done at this stage.

Phase 2: Feasibility study

Once the requirement analysis phase is completed the next

SDLC step is to define and document software needs. This process conducted with the help of 'Software Requirement Specification' document also known as 'SRS' document. It includes everything which should be designed and developed during the project life cycle.

Phase 3: Design

In this third phase, the system and software design documents are prepared as per the requirement specification document. This helps define overall system architecture. This design phase serves as input for the next phase of the model.

Phase 4: Coding

Once the system design phase is over, the next phase is coding. In this phase, developers start build the entire system by writing code using the chosen programming language. In the coding phase, tasks are divided into units or modules and assigned to the various developers. It is the longest phase of the Software Development Life Cycle process.

Phase 5: Testing

Once the software is complete, and it is deployed in the testing

environment. The testing team starts testing the functionality of the entire system. This is done to verify that the entire application works according to the customer requirement.

Phase 6: Installation/Deployment

Once the software testing phase is over and no bugs or errors left in the system then the final deployment process starts. Based on the feedback given by the project manager, the final software is released and checked for deployment issues if any.

Phase 7: Maintenance

Once the system is deployed, and customers start using the developed system, following 3 activities occur

- **Bug fixing**- bugs are reported because of some scenarios which are not tested at all
- **Upgrade**- Upgrading the application to the newer versions of the Software
- **Enhancement**- Adding some new features into the existing software

The main focus of this SDLC phase is to ensure that needs

continue to be met and that the system continues to perform as per the specification mentioned in the first phase

2. Describe the SCRUM process

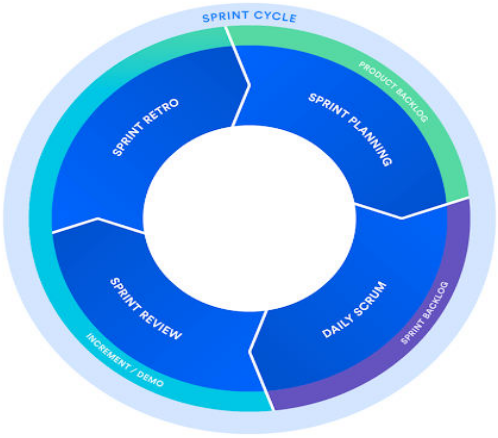
What is Scrum?

- Scrum is a management framework that teams use to self-organize and work towards a common goal.
- It describes a set of meetings, tools, and roles for efficient project delivery.
- Software teams use Scrum to solve complex problems cost effectively and sustainably.

How does Scrum work?

- Scrum is a framework that is easy to learn but difficult to become an expert in.
- The essence of Scrum is a self-organizing team delivering customer value in a time-boxed period called a Sprint.
- Scrum defines artifacts, roles, and events associated with

each Sprint.



<u>Scrum events include the following:</u>

Sprint Planning

- In this event, the team estimates the work to be completed in the next Sprint.

Sprint

- A Sprint is the actual time period when the Scrum Team works together to finish an Increment.

Daily Scrum or stand-up

- A Daily Scrum is a short meeting in which team members check in and plan for the day.

Sprint Review

- At the end of the Sprint, the team gets together for an informal session to review the work completed and showcase it to stakeholders.

3. Define software. What are the fundamental activities of the software process?

What is Software?

- Software is a set of instructions, data or programs used to operate computers and execute specific tasks.
- It is the opposite of hardware, which describes the physical aspects of a computer.
- Software is a generic term used to refer to applications, scripts and programs that run on a device.
- It can be thought of as the variable part of a computer, while hardware is the invariable part.
- The two main categories of software are application software and system software.

- An application is software that fulfills a specific need or performs tasks.
- System software is designed to run a computer's hardware and provides a platform for applications to run on top of.

Fundamental activities

The 4 basic process activities:

1. Specification
2. Development
3. Validation
4. Evolution

Software Specification:

- The process of understanding and defining what services what services are required from the system and identifying the constraints on the system operation and development.

Software design and implementation:

- A software design is a description of the structure of the software to be implemented, interfaces between system

components and sometimes algorithm used.

Software Validation:

- It is intended to show that a system both conforms to its specifications and meets the user expectations.

Software evolution:

- The maintenance of the system.

4. Explain Requirement Validation. What are its techniques

What is Requirement Validation?

- Requirements validation is the process of checking that the defined requirements are for development, and defining the system that the customer really wants.
- Requirements validation helps us detect errors at an early stage of product development so that it does not result in excessive rework when detected later in the system development life cycle.

Validation Techniques:

There are various techniques that can be used to validate the requirements. They include:

Checks-

- While checking the requirements, we proofread the requirements documents to ensure that no elicitation notes are missed out.
- During these checks, we also check the traceability level between all the requirements.
- For this, the creation of a traceability matrix is required.
- This matrix ensures that all the requirements are being properly considered and everything that is specified is justified.
- We also check the format of requirements during these checks.
- We see if the requirements are clear and well-written or not.

Prototyping-

- This is a way of building a model or simulation of the system that is to be built by the developers.
- This is a very popular technique for requirements validation among stakeholders and users as it helps them to easily identify the problems, detect missing requirements and understand how technology can help them.
- We can just reach out to the users and stakeholders and get their feedback.

Test Design-

- During test designing, we follow a small procedure where the testing team build a few testing scenarios.
- Tests have to be derived from the requirements specification The aim of this process is to figure out the errors in the specification or the details that are missed out leading to difficulties in the definition of the test scenarios.

Requirements Review-

- During requirement review, a group of knowledgeable people analyze the requirements in a structured and detailed manner and identify potential problems.
- After that, they gather up to discuss the issues and figure out a way to address the issues.
- A checklist is prepared that the reviewers fill up to provide a formal output of the review.
- After that, a final approval sign-off is done.

5. Explain the terms Safety and Security

Safety:

Safety critical systems are systems where it is essential that system operation is always safe. That is the system should never damage people or the system's environment even if the system fails.

Examples of safety critical system are control and monitoring systems in aircraft, process control process control systems in chemical and pharmaceutical plants and automobile control

systems.

Safety critical software are 2 types:

Primary safety-critical systems

Embedded software systems whose failure can cause hardware malfunction which results inhuman injury or environmental damage.

Secondary safety-critical systems

Systems whose failure indirectly results in injury. Eg - Medical Database holding details of drugs.

Ways to achieve Safety

i. **Hazard avoidance**
ii. **Hazard detection and removal**
iii. **Damage limitation**

Security:

- Security is a system property that reflects the ability to protect itself from accidental or deliberate external attack.
- Security is becoming increasingly important as systems are

networked so that external access to the system through the Internet is possible.

- Security is an essential pre-requisite for availability, reliability and safety

Example: Viruses, unauthorised use of service/data modification.

Damage from insecurity

i. **Denial of service**

ii. **Corruption of programs or data**

iii. **Disclosure of confidential information**

6. Explain Requirement Engineering process

Requirements engineering is the process of identifying, eliciting, analyzing, specifying, validating, and managing the needs and expectations of stakeholders for a software system.

- The requirements engineering process is an iterative process that involves several steps, including:

Requirements Elicitation:

- This is the process of gathering information about the needs and expectations of stakeholders for the software system.
- This step involves interviews, surveys, focus groups, and other techniques to gather information from stakeholders.

Requirements Analysis:

- This step involves analyzing the information gathered in the requirements elicitation step to identify the high-level goals and objectives of the software system.
- It also involves identifying any constraints or limitations that may affect the development of the software system.

Requirements Specification:

- This step involves documenting the requirements identified in the analysis step in a clear, consistent, and unambiguous manner.
- This step also involves prioritizing and grouping the requirements into manageable chunks.

Requirements Validation:

- This step involves checking that the requirements are complete, consistent, and accurate.
- It also involves checking that the requirements are testable and that they meet the needs and expectations of stakeholders.

Requirements Management:

- This step involves managing the requirements throughout the software development life cycle, including tracking and controlling changes, and ensuring that the requirements are still valid and relevant.

The Requirements Engineering process is a critical step in the software development life cycle as it helps to ensure that the software system being developed meets the needs and expectations of stakeholders, and that it is developed on time, within budget, and to the required quality.

7. Explain User Interface Design process

User interface is the front-end application view to which user

interacts in order to use the software.

The software becomes more popular if its user interface is:

- Attractive
- Simple to use
- Responsive in short time
- Clear to understand
- Consistent on all interface screens

There are two types of User Interface:

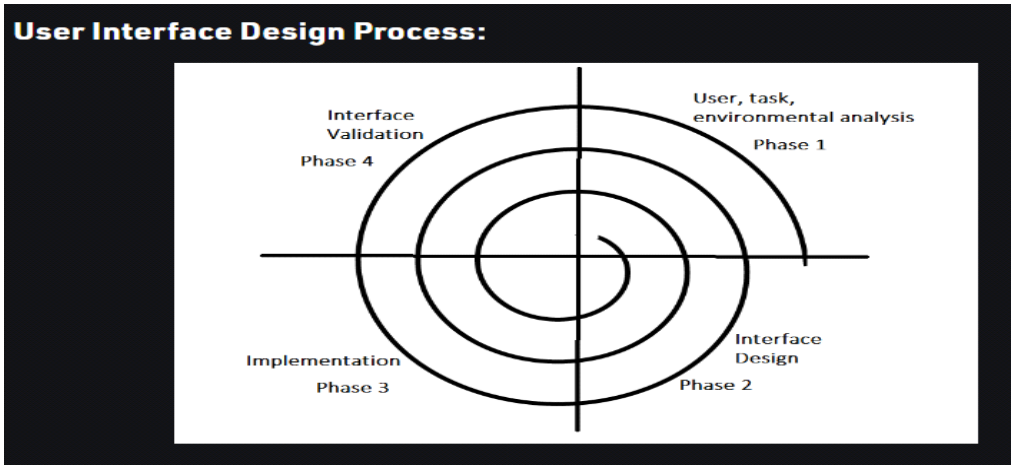
i. **Command Line Interface**: Command Line Interface provides a command prompt, where the user types the command and feeds to the system. The user needs to remember the syntax of the command and its use.

ii. **Graphical User Interface**: Graphical User Interface provides the simple interactive interface to interact with the system. GUI can be a combination of both hardware and software. Using GUI, user interprets the software.

The analysis and design process of a user interface is iterative

and can be represented by a spiral model.

The analysis and design process of user interface consists of four framework activities.



i. The goal of this phase is to define the set of interface objects and actions i.e. Control mechanisms that enable the user to perform desired tasks. Indicate how these control mechanisms affect the system.

ii. Specify the action sequence of tasks and subtasks, also called a user scenario.

iii. Indicate the state of the system when the user performs a particular task.

8. Describe the steps in Project Scheduling

What is a project schedule?

- A project schedule provides a general overview of your project, including the timeline, project tasks, dependencies, and assigned team members.
- Essentially, a project schedule should be able to tell you everything you need to know about your project at first glance.

7 steps to create a project schedule

- **Define your project goals.**

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

- **Identify all stakeholders.**

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

- **Determine your final deadline.**

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

- **List each step or task.**

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

- **Assign a team member responsible for each task.**

Decide who will take on each task and subtask, and be transparent with deadlines. Remember that your colleagues likely have other projects going on at the same time. Be mindful of their workload so they don ţ feel overloaded.

- **Work backward to set due dates for each task.**

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

- **Organize your project schedule in one tool, and share it with your team.**

You've successfully built your project plan and now it's important to organize it in a way that everyone involved can

see and work from it. Finding a tool that can help you do both will be critical to your success.

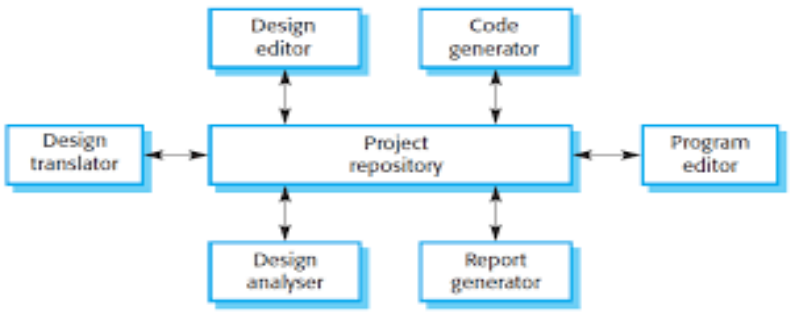
9. Write a note on Repository Model and Client Server Model

A repository model

- It is a system that will allow interfacing sub-systems to share the same data. Sub-system must exchange data so that they can work together effectively.
- This may be done in two ways:

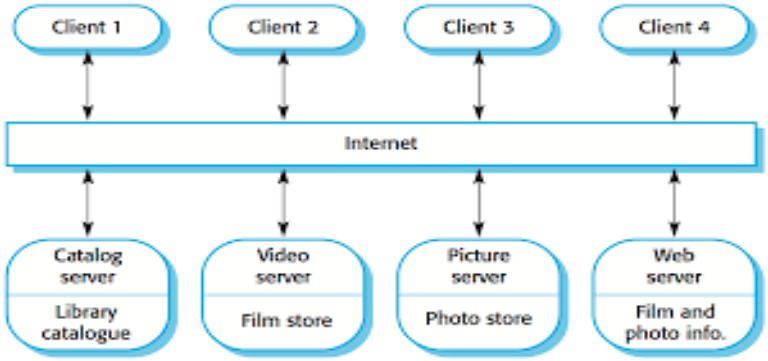
i. All shared data is held in a central database that can be accessed by all subsystems. It is called repository model.

ii. Each sub-system maintains its own database.



Client-server model

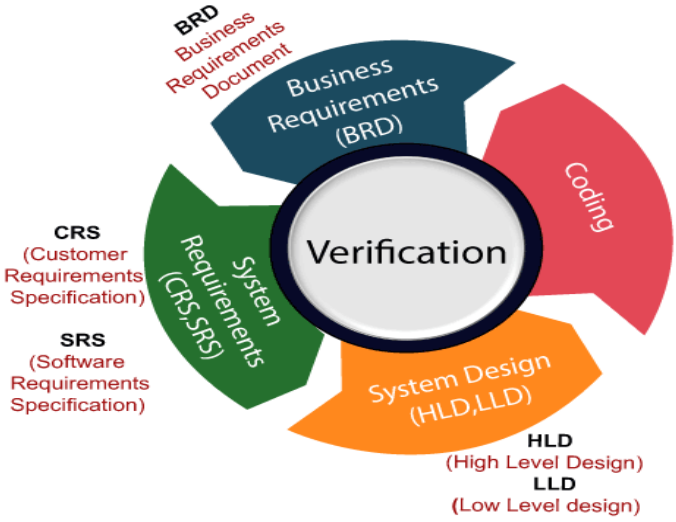
- It is a networking computing system design that illustrates a relationship between two or more computers, where the client computers request and receive services or resources from a powerful centralized server computer.
- It describes a specific way devices access the information you store in servers.
- It also allows multiple clients to open applications or retrieve files from an individual server, which helps maintain consistency across all devices.
- Many companies across various industries use servers to store and access information, offering more processing power and providing more extensive storage space.



10. Explain Verification and Validation

Verification testing

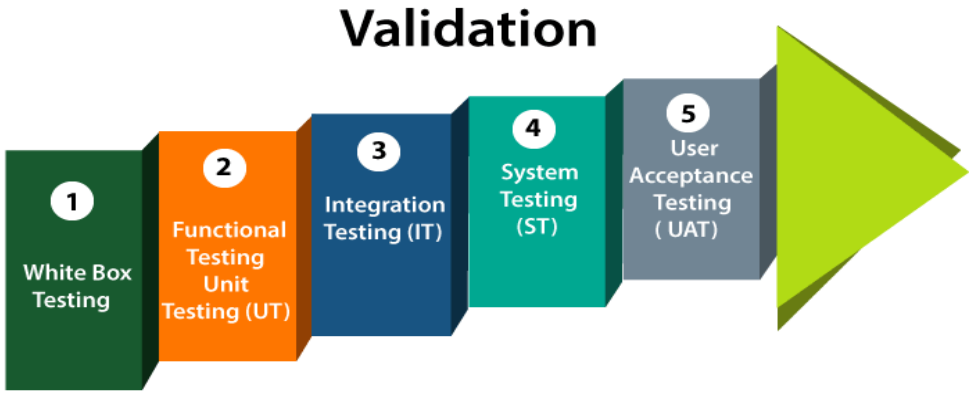
- Verification testing includes different activities such as business requirements, system requirements, design review, and code walkthrough while developing a product.
- It is also known as static testing, where we are ensuring that "we are developing the right product or not".
- And it also checks that the developed application fulfilling all the requirements given by the client.



Validation testing

- Validation testing is testing where tester performed functional and non-functional testing.

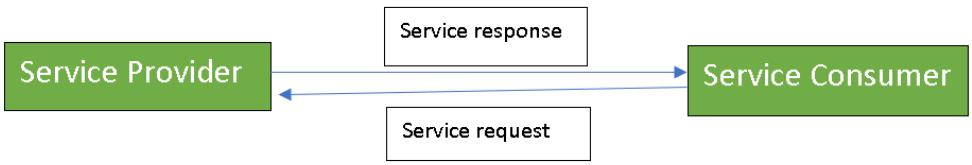
- Here functional testing includes Unit Testing (UT), Integration Testing (IT) and System Testing (ST), and non-functional testing includes User acceptance testing (UAT).
- Validation testing is also known as dynamic testing, where we are ensuring that "we have developed the product right."
- And it also checks that the software meets the business needs of the client.



11. Explain Service Oriented architecture

- Service-Oriented Architecture (SOA) is a stage in the evolution of application development and/or integration.

- It defines a way to make software components reusable using the interfaces.
- Formally, SOA is an architectural approach in which applications make use of services available in the network.
- In this architecture, services are provided to form applications, through a network call over the internet.
- It uses common communication standards to speed up and streamline the service integrations in applications.
- Each service in SOA is a complete business function in itself.
- The services are published in such a way that it makes it easy for the developers to assemble their apps using those services.
- Note that SOA is different from microservice architecture.



- SOA allows users to combine a large number of facilities

from existing services to form applications.

- SOA encompasses a set of design principles that structure system development and provide means for integrating components into a coherent and decentralized system.
- SOA-based computing packages functionalities into a set of interoperable services, which can be integrated into different software systems belonging to separate business domains.

There are two major roles within Service-oriented Architecture:

Service provider:

- The service provider is the maintainer of the service and the organization that makes available one or more services for others to use.
- To advertise services, the provider can publish them in a registry, together with a service contract that specifies the nature of the service, how to use it, the requirements for the service, and the fees charged.

Service consumer:

- The service consumer can locate the service metadata in the registry and develop the required client components to bind and use the service.

12. Explain Function point metrics

- Function Point Analysis was initially developed by Allan J. Albercht in 1979 at IBM
- FPA gives a dimensionless number defined in function points which we have found to be an effective relative measure of function value delivered to our customer.
- FPA provides a standardized method to functionally size the software work product.
- This work product is the output of software new development and improvement projects for subsequent releases.
- It is the software that is relocated to the production application at project implementation.

- It measures functionality from the user's point of view i.e. on the basis of what the user requests and receives in return.
- Function Point Analysis (FPA) is a method or set of rules of Functional Size Measurement.
- It assesses the functionality delivered to its users, based on the user's external view of the functional requirements.
- It measures the logical view of an application, not the physically implemented view or the internal technical view.
- The Function Point Analysis technique is used to analyze the functionality delivered by software and Unadjusted Function Point (UFP) is the unit of measurement.

Objectives of FPA:

- The objective of FPA is to measure the functionality that the user requests and receives.
- The objective of FPA is to measure software development and maintenance independently of the technology used for implementation.

- It should be simple enough to minimize the overhead of the measurement process.
- It should be a consistent measure among various projects and organizations.

13. Explain Two tier and multi-Tier Client server architecture

Two-tier client/server:-

- It is a type of multi-tier computing architecture in which an entire application is distributed as two distinct layers or tiers.
- It divides the application logic, data and processing between client and server devices.
- A two-tier client/server works when most or all of the application logic and data is hosted on a server.
- The client integrates with the presentation layer and accesses the server for application specific tasks and processing.
- For example, the core application and data are installed at

a central server.

- One or more client devices uses its client-end application to request data or processes from the server.
- The server sends the required data or performs a process to fulfill the query.
- In another two-tier client/server instance, such as a data backup architecture, the application access and logic may be with the client device, whereas the server stores and provides the core data.

Multi-tier Architecture

- It is a software architecture in which different software components, organized in tiers (layers), provide dedicated functionality.
- The most common occurrence of a multi-tier architecture is a three-tier system consisting of a data management tier (mostly encompassing one or several database servers), an application tier (business logic) and a client tier (interface functionality).

- Novel deployments come with additional tiers.
- Web information systems, for instance, encompass a dedicated tier (web tier) between client and application layer.

14. Describe the various control styles of software systems

- Control models are models deployed in software engineering that are concerned with the control flow between the subsystems.
- They are distinct from the system decomposition model.

There are two types of control models: centralized and event-based control model.

1)Centralized Control Model

- Centralized model is a formulation of centralized control in which one subsystem has overall responsibility for control and starts and stops other subsystems.
- It is a control subsystem that takes responsibility for managing the execution of other subsystems.

Centralized models are classified into call-return and manager model.

i. ***Call-return Model:***

   In call-return model, it is a model which has top-down subroutine architecture where control starts at the top of a subroutine hierarchy and moves downwards.

ii. ***Manager Model:***

   Manager model is applicable to concurrent systems. One system component controls the stopping, starting and coordination of other system processes.

2)Event-based Control Model

- Event-based models are those in which each sub-system can respond to externally generated events from other subsystems or the system's environment.
- It is a system driven by externally generated events where the timing of the events is out with the control of the subsystems which process the event.

Event-based models are classified into broadcast and

interrupt-driven models.

i. ***Broadcast models:***

In these models, an event is, in principle, broadcast to all sub-systems. Any sub-system, which is designed to handle that event, responds to it.

ii. ***Interrupt-driven models:***

These are exclusively used in real-time systems where an interrupt handler detects external interrupts. They are then passed to some other component for processing.

15. Explain the term Ethnography

- Ethnography is a qualitative research method used to study people and cultures.
- It is largely adopted in disciplines outside software engineering, including different areas of computer science.
- Ethnography can provide an in-depth understanding of the socio-technological realities surrounding everyday software development practice, i.e., it can help to uncover not only

what practitioners do, but also why they do it.

- Despite its potential, ethnography has not been widely adopted by empirical software engineering researchers, and receives little attention in the related literature.
- The main goal of this paper is to explain how empirical software engineering researchers would benefit from adopting ethnography.

This is achieved by explicating four roles that ethnography can play in furthering the goals of empirical software engineering:

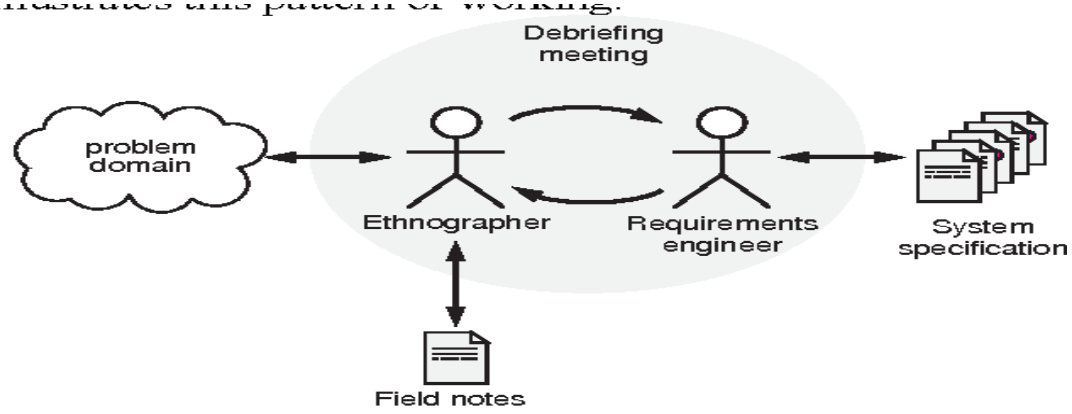
a. To strengthen investigations into the social and human aspects of software engineering.

b. to inform the design of software engineering tools.

c. To improve method and process development.

d. To inform research programmes.

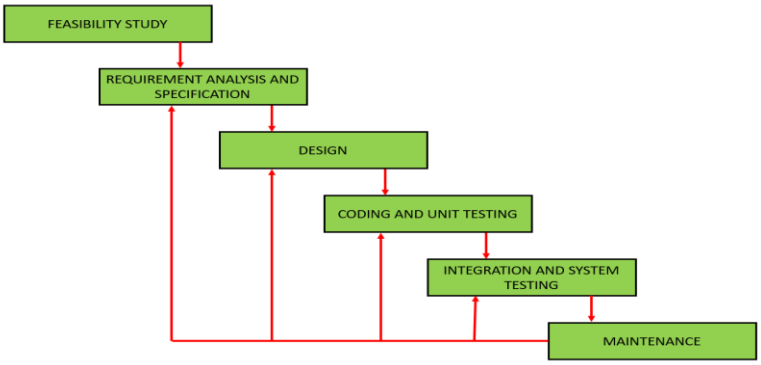
illustrates this pattern of working.



16. Explain the concept of Iterative development model.

- Iterative development is a way of breaking down the software development of a large application into smaller chunks.
- In iterative development, feature code is designed, developed and tested in repeated cycles.
- With each iteration, additional features can be designed, developed and tested until there is a fully functional software application ready to be deployed to customers.
- Typically iterative development is used in conjunction with incremental development in which a longer software development cycle is split into smaller segments that build upon each other.

- Iterative and incremental development are key practices in Agile development methodologies.
- In Agile methodologies, the shorter development cycle, referred to as an iteration or sprint, is time-boxed (limited to a certain increment of time, such as two weeks).
- At the end of the iteration, working code is expected that can be demonstrated for a customer.
- Iterative development contrasts with a traditional waterfall method in which each phase of the software development life cycle is "gated".
- Coding doesn't begin until design of the entire software application is complete and has gone through a phase gate review.
- The purpose of working iteratively is to allow more flexibility for changes.



17. Explain service engineering

- Service engineering represents a software engineering method in which software systems are broken down into individual components that can be reused within different software contexts.
- Sometimes referred to as service-oriented software engineering or service-oriented architecture, service engineering represents a cost-effective strategy for building web-based applications tailored to the needs of each organization within a short timeframe.
- In contemporary businesses and organizational settings, service engineering constitutes an essential part of building web services.

- Service engineering represents a cost-effective method for developing enterprise-based systems because it enables users to design platforms with specific functions instead of paying for software with unneeded features.
- A service refers to any self-contained unit of programming that carries out a specific function.
- By identifying the specifications needed by the client, the engineer can use pre-existing services to quickly build new programs or interfaces that address the organization's needs.

Some examples of potential services are:

a. Checking a customer's account

b. Processing a payment

c. Generating an invoice

d. Processing a loan application

e. Finding a patient's medical record

18. Write a note on SRS.

- The production of the requirements stage of the software development process is Software Requirements Specifications (SRS) (also called a requirements document).
- This report lays a foundation for software engineering activities and is constructing when entire requirements are elicited and analyzed.
- SRS is a formal report, which acts as a representation of software that enables the customers to review whether it (SRS) is according to their requirements.
- Also, it comprises user requirements for a system as well as detailed specifications of the system requirements.
- The SRS is a specification for a specific software product, program, or set of applications that perform particular functions in a specific environment.
- It serves several goals depending on who is writing it. First, the SRS could be written by the client of a system.
- Second, the SRS could be written by a developer of the

system.

- The two methods create entirely various situations and establish different purposes for the document altogether.
- The first case, SRS, is used to define the needs and expectation of the users.
- The second case, SRS, is written for various purposes and serves as a contract document between customer and developer.

19. Describe various Management activities

Software Project Management consists of many activities, that includes planning of the project, deciding the scope of product, estimation of cost in different terms, scheduling of tasks, etc.

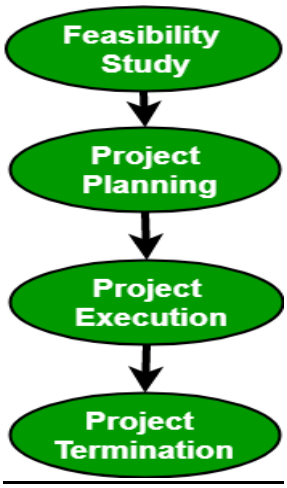
The list of activities are as follows:

a. Feasibility study

b. Project Planning

c. Project Execution

d. Project Termination



Feasibility Study:

- A feasibility study explores system requirements to determine project feasibility.
- There are several fields of feasibility study including economic feasibility, operational feasibility, and technical feasibility.
- The goal is to determine whether the system can be implemented or not.
- The process of feasibility study takes as input the required details as specified by the user and other domain-specific details.

Project Planning:

A detailed plan stating a stepwise strategy to achieve the listed objectives is an integral part of any project.

Planning consists of the following activities:

- Set objectives or goals
- Develop strategies
- Develop project policies
- Determine courses of action
- Making planning decisions
- Set procedures and rules for the project
- Develop a software project plan
- Prepare budget
- Conduct risk management
- Document software project plans

Project Execution:

- A project is executed by choosing an appropriate software

development lifecycle model(SDLC).

- It includes a number of steps including requirements analysis, design, coding, testing and implementation, testing, delivery, and maintenance.
- There are a number of factors that need to be considered while doing so including the size of the system, the nature of the project, time and budget constraints, domain requirements, etc.
- An inappropriate SDLC can lead to the failure of the project.

Project Termination:

- There can be several reasons for the termination of a project.
- Though expecting a project to terminate after successful completion is conventional, at times, a project may also terminate without completion.
- Projects have to be closed down when the requirements are not fulfilled according to given time and cost

constraints.

The Primary goal is to increase productivity with fewer errors.

20. Write a note on Release testing

What is release testing?

- Release testing refers to coding practices and test strategies that give teams confidence that a software release candidate is ready for users.
- Release testing aims to find and eliminate errors and bugs from a software release so that it can be released to users.
- Release testing comprises all the development and testing activities that ensure a software release candidate is ready for users.
- In other words, the objective of release testing is to build confidence into a release candidate. Release testing is a testing approach or strategy rather than one single grand testing method.
- Development teams focus their quality assurance efforts

on this one release candidate, trying to break it so it can be fixed before it gets to actual users.

Release testing environment

- Before we can start release testing, we need to prepare the environment where our release candidate will be running.
- Once we can run the release candidate software, we can start testing it according to various release test methods.
- Our release testing environment should match our production environment.
- However, in practice, maintaining a testing environment identical to our production environment isn't possible.
- As a consequence, we always run our release candidate within an approximation of our production environment.