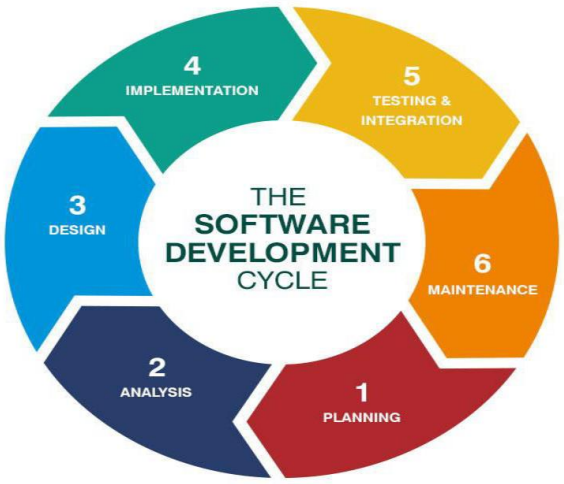
S.E. SEM 4 EXTERNAL EXAM SOLUTION

Explain the SDLC process

- 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

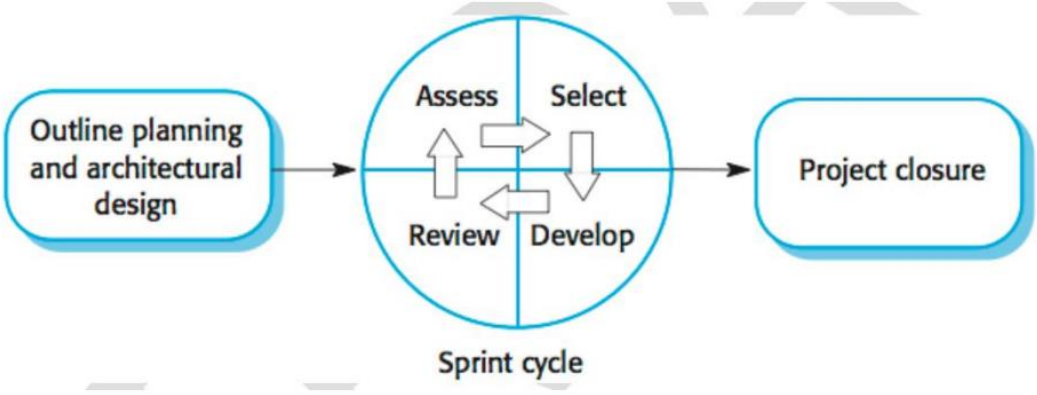
The main focus of this SDLC phase is to ensure that needs continue to be met and that the system continues to perform as per the specification mentioned in the first phase

2. Describe the SCRUM process

The principal responsibility of software project managers is to manage the project so that the software is delivered on time and within the planned budget for the project. They supervise the work of software engineers and monitor how well the software development is progressing. The standard approach to project management is plan driven.

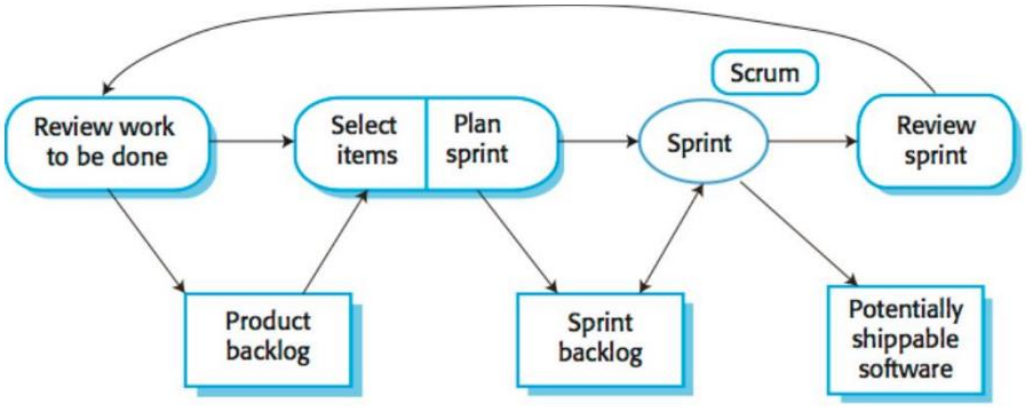
The innovative feature of Scrum is its central phase namely the sprint cycles. A Scrum sprint is a planning unit in which work to be done is assessed, features are selected for development and the software is implemented. At the end of a sprint the completed functionality is delivered to the stakeholders. Key characteristics of this process are as follows:

1) Sprints are fixed length, normally 2-4 weeks. They correspond to the development of a release of the system in XP.
2) The starting point for planning is the product backlog, which is the list of work to be done on the project.
3) The selection phase involves all of the project team who work with the customer (product owner) to select the features and functionality to be developed during the sprint.
4) Once these are agreed, the team organize themselves to develop the software. During this stage the team is relatively isolated from the product owner and the organization, with all communications channeled through the ScrumMaster. The role of the ScrumMaster is to protect the development team from external distractions.



5) At the end of the sprint the work done is reviewed and presented to stakeholders (including the product owner). Velocity is calculated during the sprint review; it provides an estimate of how much product backlog the team can cover in a single sprint. Understanding the team's velocity helps them estimate what can be covered in a sprint and provides a basis for measuring and improving performance.

The next sprint cycle then begins.



The ScrumMaster is a facilitator who arranges short daily meetings (daily scrums), tracks the backlog of work to be done, records decisions, measures progress against the backlog and communicates with the product owner and management outside of the team. The whole team attends daily scrums where all team members share information, describe their progress since the last meeting, problems that have arisen and what is planned for the following day.

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 establishing what services are required and the constraints on the system's operation and development. Requirements engineering process:

Feasibility study: is it technically and financially feasible to build the system?

Requirements elicitation and analysis: what do the system stakeholders require or expect from the system?

Requirements specification: defining the requirements in detail

Requirements validation: checking the validity of the requirements

Software design and implementation:

The process of converting the system specification into an executable system.

Software design: design a software structure that realizes the specification;

Implementation: translate this structure into an executable program;

The activities of design and implementation are closely related and may be interleaved

Architectural design: identify the overall structure of the system, the principal components (sometimes called sub-systems or modules), their relationships and how they are distributed.

Interface design: define the interfaces between system components

Component design: take each system component and design how it will operate.
Database design: design the system data structures and how these are to be represented in a database.

Software Validation:

Verification and validation (V & V) is intended to show that a system conforms to its specification and meets the requirements of the system customer.

Validation: are we building the right product (what the customer wants)?

Verification: are we building the product, right? V & V involves checking and review processes and system testing. System testing involves executing the system with test cases that are derived from the specification of the real data to be processed by the system.

Testing is the most commonly used V & V activity and includes the following stages:

Development or component testing: individual components are tested independently; components may be functions or objects or coherent groupings of these entities.

System testing: testing of the system as a whole, testing of emergent properties is particularly important.

Acceptance testing: testing with customer data to check that the system meets the customer's needs.

Software evolution:

Software is inherently flexible and can change. As requirements change through changing business circumstances, the software that supports the business must also evolve and change. Although there has been a demarcation between development and evolution (maintenance) this is increasingly irrelevant as fewer and fewer systems are completely new.

4. Explain Requirement Validation. What are its techniques

Requirements validation is the process of checking that requirements actually define the system that the customer really wants. Requirements validation is important because errors in a requirements document can lead to extensive rework costs when these problems are discovered during development or after the system is in service.

During the requirements validation process, different types of checks should be carried out on the requirements in the requirements document. These checks include:

1. Validity checks - A user may think that a system is needed to perform certain functions.
2. Consistency checks - Requirements in the document should not conflict.
3. Completeness checks - The requirements document should include requirements that define all functions and the constraints intended by the system user.
4. Realism checks - Using knowledge of existing technology, the requirements should be checked to ensure that they can actually be implemented.
5. Verifiability - To reduce the potential for dispute between customer and contractor, system requirements should always be written so that they are verifiable.

There are a number of requirements validation techniques that can be used individually or in conjunction with one another:

1. Requirements reviews - The requirements are analyzed systematically by a team of reviewers who check for errors and inconsistencies.
2. Prototyping - In this approach to validation, an executable model of the system in question is demonstrated to end-users and customers. They can experiment with this model to see if it meets their real needs.
3. Test-case generation - Requirements should be testable. If the tests for the requirements are devised as part of the validation process, this

often reveals requirements problems. If a test is difficult or impossible to design, this usually means that the requirements will be difficult to implement and should be reconsidered. Developing tests from the user requirements before any code is written is an integral part of extreme programming.

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 systems in chemical and pharmaceutical plants and automobile control systems.
Safety requirements are exclusive requirements i.e., they exclude undesirable situations rather than specify required system services. 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

• Hazard avoidance
  – The system is designed so that hazard simply cannot arise.
  – E.g., Press 2 buttons at the same time in a cutting machine to start
• Hazard detection and removal
  – The system is designed so that hazards are detected and removed before they result in an accident.
  – E.g., Open relief valve on detection over pressure in chemical plant.

• Damage limitation
  – The system includes protection features that minimise the damage that may result from an accident
  – Automatic fire safety system in aircraft.

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 prerequisite for availability, reliability and safety Example: Viruses, unauthorised use of service/data modification

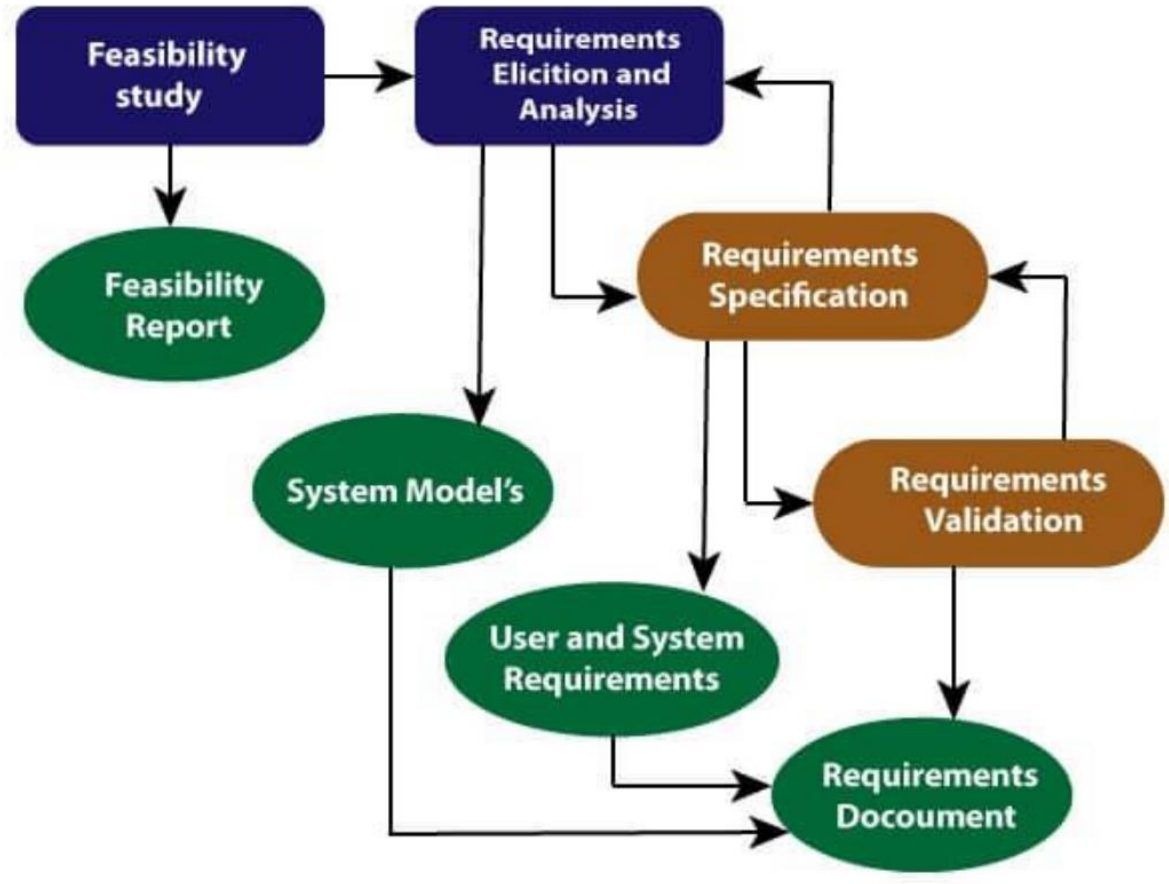
Damage from insecurity
• Denial of service
  – The system is forced into a state where normal services become unavailable.
• Corruption of programs or data
  – The system components of the system may alter in an unauthorised way, which affect system behaviour & hence its reliability and safety
• Disclosure of confidential information
  – Information that is managed by the system may be exposed to people who are not authorised to read or use that information

Security assurance
• Vulnerability avoidance
  – The system is designed so that vulnerabilities do not occur.
  For example, if there is no external network connection then external attack is impossible
• Attack detection and elimination
  – The system is designed so that attacks on vulnerabilities are detected and remove them before they result in an exposure.
  For example, virus checkers find and remove viruses before they infect a system

• Exposure limitation

– The consequences of a successful attack are minimised.
For example, a backup policy allows damaged information to be restored

6. Explain Requirement Engineering process



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

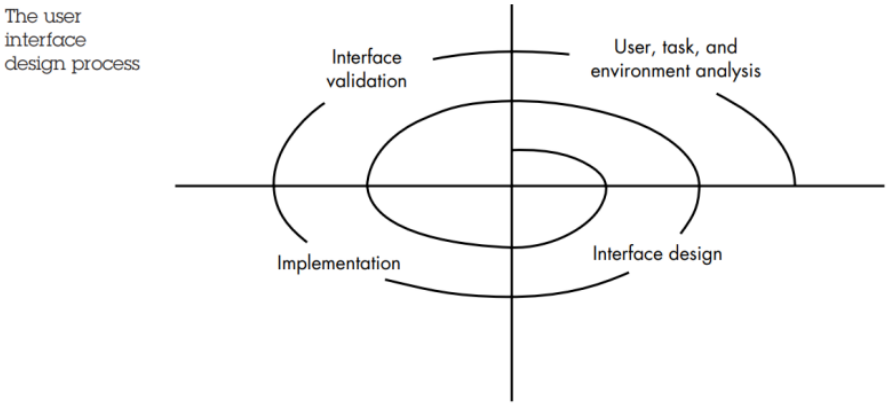
controlling changes, and ensuring that the requirements are still valid and relevant.

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

The design process for user interfaces is iterative and can be represented using a spiral model. The user interface design process encompasses four distinct framework activities

1. User, task, and environment analysis and modeling
2. Interface design
3. Interface construction
4. Interface validation



The spiral shown in Figure implies that each of these tasks will occur more than once, with each pass around the spiral representing additional elaboration of requirements and the resultant design. In most cases, the implementation activity involves prototyping—the only practical way to validate what has been designed.

1. User, task, and environment analysis and modeling

   Before designing any solution, you need to know what actually you have to design. In this framework, you have to study the user's profile who will use this interface.

   Once all the requirements are gathered a detailed analysis is conducted. In the analysis part, the tasks that the user performs to establish the goals of the system are identified, described and elaborated. The analysis of the user environment focuses on the physical work environment. Among the questions to be asked are:

   Where will the interface be located physically? Will the user be sitting, standing, or performing other tasks unrelated to the interface? Does the interface hardware accommodate space, light, or noise constraints? Are there special human factors considerations driven by environmental factors?

2. Interface design

   After completing the interface analysis, the designer identifies the task that the end-user requires. Interface designing is also an iterative process. The designer first identifies the objects and the operations performed on those objects. The designer also has to define the events that would change the state of the user interface. Further, the designer has to outline each state of the user interface as it would appear to the enduser The designer also has to define the events that would change the state of the user interface. Further, the designer has to outline each state of the user interface as it would appear to the enduser.

3. Interface construction

   After identifying the objects, operations and events the designer creates a prototype. This prototype helps in the evaluation of the real-world scenario. This process is also iterative and the

construction continues till the prototype is approved for conducting real-world scenarios.
As iterative design process continues a User Interface toolkit that allows the creation of windows, menus, device interaction, error messages, commands, and many other elements of an interactive environment can be used for completing the construction of an interface.

4. Interface validation
   The validation phase is performed to see whether the interface can perform user tasks along with all the variations in the real world. Like, as to whether the interface can perform all general user tasks? Is it easy to use, and understand and we can accept it as a useful tool?

8. Describe the steps in Project Scheduling

i) A project schedule provides a general overview of your project, including the timeline, project tasks, dependencies, and assigned team members.
ii) Essentially, a project schedule should be able to tell you everything you need to know about your project at first glance.
iii) Following are the steps for project scheduling:-

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

b) **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.

c) **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.

d) **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.

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

f) **Work backward to set due dates for each task.**
igure 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.

g) **Organize your project schedule in one tool**
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.

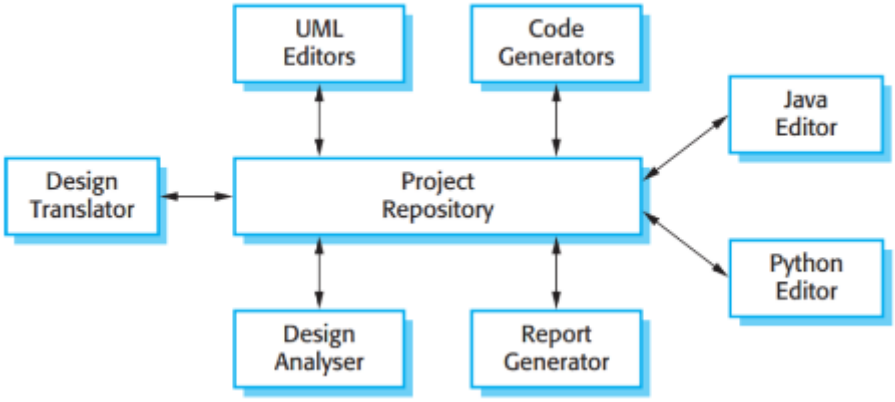
9. Write a note on Repository Model and Client Server Model

i) **Repository Architecture**

a) The majority of systems that use large amounts of data are organized around a shared database or repository.
b) This model is therefore suited to applications in which data is generated by one component and used by another.

c) Examples of this type of system include command and control systems, management information systems, CAD systems, and interactive development environments for software.

d)



e) The repository in this case might be a version-controlled environment that keeps track of changes to software and allows rollback to earlier versions.

f) versions. Organizing tools around a repository is an efficient way to share large amounts of data. There is no need to transmit data explicitly from one component to another.

ii) Client Server Model

a) A system that follows the client–server pattern is organized as a set of services and associated servers, and clients that access and use the services.

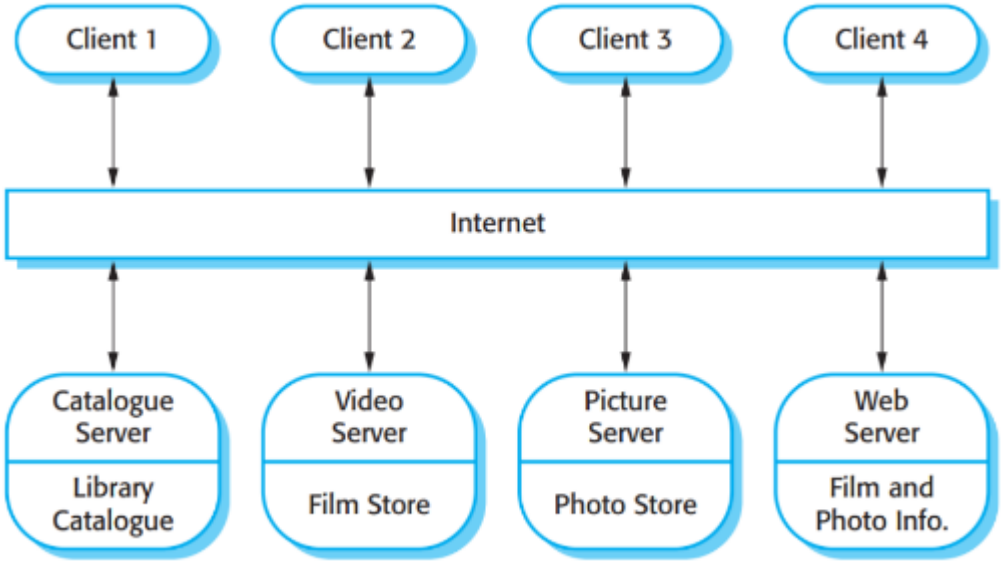
b) The major components of this model are:

1) A set of servers that offer services to other components. Examples of servers include print servers that offer printing services, file servers that offer file management services which offers programming language compilation services.

2) A set of clients that call on the services offered by servers. There will normally be several instances of a client program executing concurrently on different computers.

3) A network that allows the clients to access these services. Most client–server systems are implemented as distributed systems, connected using Internet protocols.

c) Clients may have to know the names of the available servers and the services that they provide.



d)

10. Explain Verification and Validation

In software project management, software testing, and software engineering, verification and validation (V&V) is the process of checking that a software system meets specifications and requirements so that it fulfills its intended purpose

Barry Boehm described verification and validation as the following:

Verification: Are we building the product, right?

Validation: Are we building the right product?

Verification is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfills the requirements that we have. Verification is Static Testing.

Activities involved in verification:

1. Inspections

2. Reviews
3. Walkthroughs
4. Desk-checking

Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e., it checks what we are developing is the right product. it is validation of actual and expected product. Validation is the Dynamic Testing

Activities involved in validation:
1. Black box testing
2. White box testing
3. Unit testing
4. Integration testing

The role of Verification involves checking that the software contains to its specifications. We should check that it meets its specified functional and non – functional requirements. The aim of Validation is to ensure that the software system meets the customers expectations.
The ultimate goal of the verification and validation is to establish confidence that the software system is “fit for purpose”.

11. Explain Service Oriented architecture

i) Service-Oriented Architecture (SOA) is a stage in the evolution of application development. It defines a way to make software components reusable using the interfaces.
ii) SOA is an architectural approach in which applications make use of services available in the network

iii) It uses common communication standards to speed up and streamline the service integrations in applications.
iv) SOA allows users to combine a large number of facilities from existing services to form applications.

v) As services are independent of each other they can be updated and modified easily without affecting other services.

vi) It Provides interoperability between the services and methods for service encapsulation, service discovery, service composition.

vii) There are two major roles within Service-oriented Architecture:

a) **Service provider:**

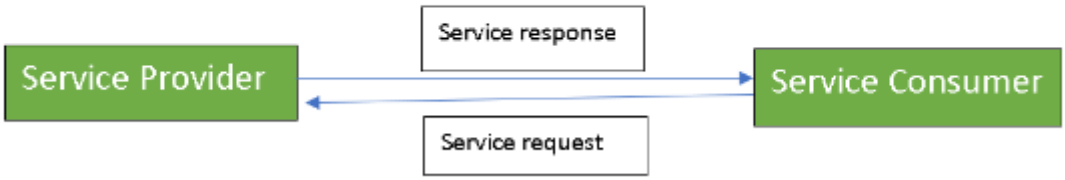
1) The service provider is the maintainer of the service and the organization that makes available one or more services for others to use

2) 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.

b) **Service consumer:**

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

2) Services might aggregate information and data retrieved from other services or create workflows of services to satisfy the request of a given service consumer.



12. Explain Function point metrics

i) Function point metrics provide a standardized method for measuring the various functions of a software application
ii) It measures the functionality from the user's point of view, that is, on the basis of what the user requests and receives in return.
iii) FPs of an application is found out by counting the number and types of functions used in the applications.
iv) Various functions used in an application can be put under five types which are:-
a) **EI** – The number of external inputs. These are elementary processes in which derived data passes across the boundary from outside to inside.

b) **EO** – The number of external output. These are elementary processes in which derived data passes across the boundary from inside to outside.

c) **EQ** – The number of external queries. These are elementary processes with both input and output components that result in data retrieval from one or more internal logical files and external interface files.

d) **ILF** – The number of internal log files. These are user identifiable groups of logically related data that resides entirely within the applications boundary .

e) **ELF** − The number of external log files. These are user identifiable groups of logically related data that are used for reference purposes only, and which reside entirely outside the system.

v) All the parameters mentioned above are assigned some weights that have been experimentally determined. Here that weighing factor will be simple, average, or complex for a measurement parameter type.

The Function Point (FP) is thus calculated with the following formula.

FP=Count-total*[0.65+0.01*∑(f_i)]
= Count-total * CAF

where Count-total is obtained from the above Table.

CAF = [0.65 + 0.01 *∑(f_i)]

vi) LOCs of an application can be estimated from FPs. That is, they are interconvertible. **This process is known as backfiring.**

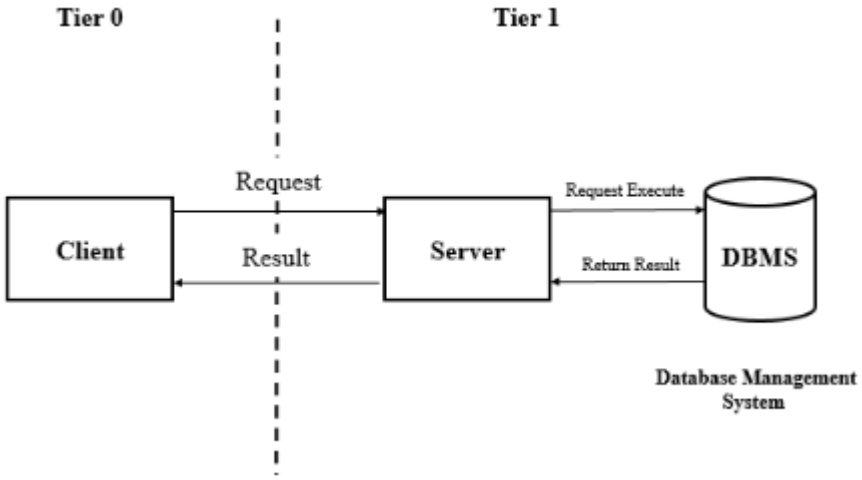
13. Explain Two tier and multi-Tier Client server architecture

i) **Two tier**

a) In a two-tier architecture, the client is on the first tier. The database server and web application server reside on the same server machine, which is the second tier.
b) This second tier serves the data and executes the business logic for the web application.

c) The client integrates with the presentation layer and accesses the server for application specific tasks and processing.
d) The advantage of this model is this structure is quite easy to maintain and modify.The communication between the client and server in the form of request response messages is quite fast.
e) The disadvantage of this model is If the client nodes are increased beyond capacity in the structure, then the server is not able to handle the request.
f) An example of a two tier client/server structure is a web server. It returns the required web pages to the clients that requested them.

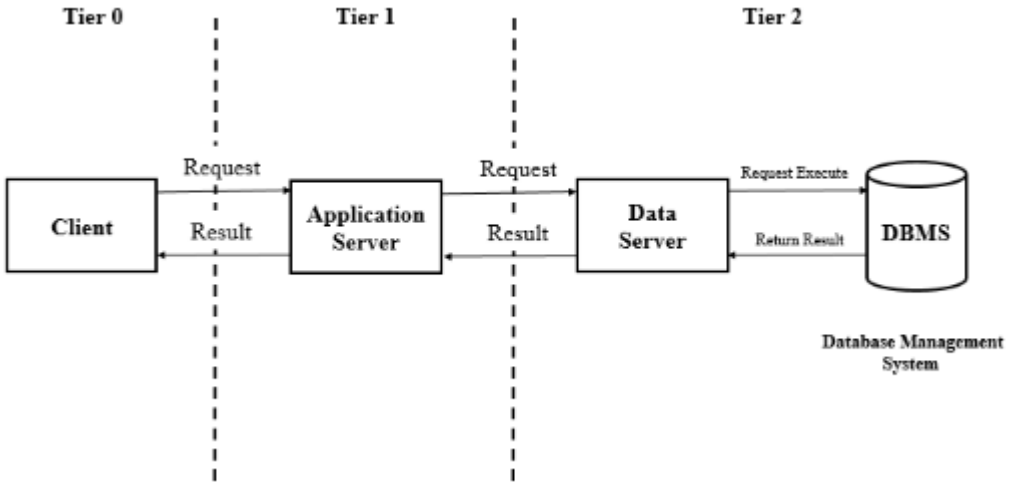
g)



ii) Multi-Tier architecture

a) Multi-Tier (with N more than 3) is really 3 tier architectures in which the middle tier is split up into new tiers.
b) The application tier is broken down into separate parts. What these parts are differs from system to system.
c) The primary advantage of N-tier architectures is that they make load balancing possible.
d) N-tiered architectures are also more easily scalable, since only servers experiencing high demand, such as the application server, need be upgraded.
e) The primary disadvantage of N-tier architectures is that it is also more difficult to program and test an N-tier architecture due to its increased complexity.

f)



14. Describe the various control styles of software systems

i) To work as a system, sub-systems must be controlled so that their services are delivered to the right place at the right time.

ii) There are two generic control styles that are used in software systems:

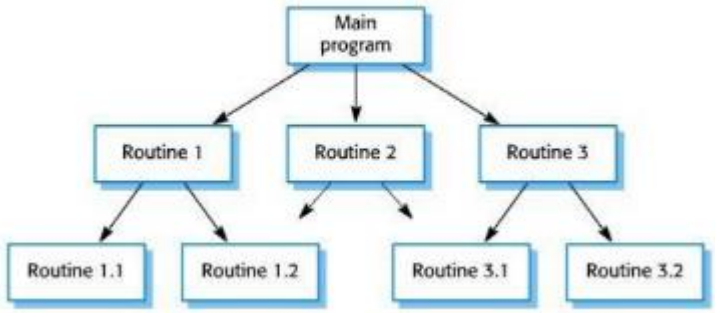
a) **Centralized control:**

Centralized model is a formulation of centralized control in which one subsystem has overall responsibility for control and starts and stops other subsystems.

Further centralized control model is divided in 2 type which are as follows:-

- **Call-return Model**

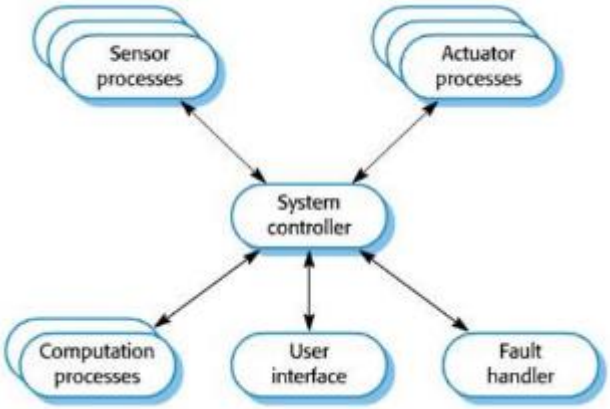
  1) In call-return model, it is a model which has top-down subroutine architecture where control starts at the top of a subroutine hierarchy and moves downwards.
  2) It is applicable to sequential systems.
  3) This familiar model is embedded in programming languages such as C, Ada and Pascal.



4)

- **Manager Model**

  1) Manager model is applicable to concurrent systems. One system component controls the stopping, starting and coordination of other system processes.
  2) It can be implemented in sequential systems as a case system.
  3) It is often used in real time systems which do not have very tight constraints.



4)

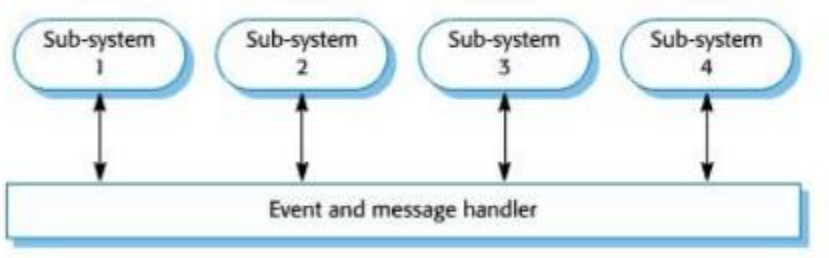
b) Event based control:

Event-based models are those in which each sub-systemcan respond to externally generated events.

Further event based models are of 2 types:-

- **Broadcast Model**

  1) It is a model in which an event is broadcast to all subsystems. Any subsystem which can handle the broadcasting event may behave as a broadcast model.
  2) network. The advantage of this model is that evolution is simple. This distribution is transparent to other components.
  3) The disadvantage is that the components don't know if the event will be handled.

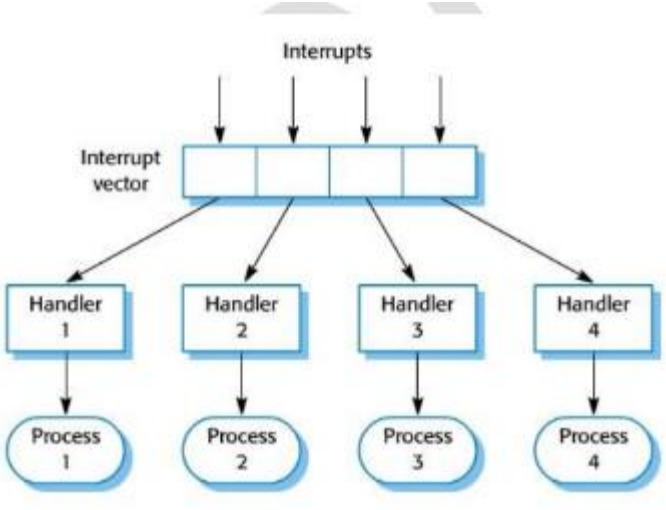


  4)

- **Interrupt-driven Model**

  1) Interrupt-driven model is used in real time systems where interrupts are detected by and interrupt handler and passed to some other component for processing.
  2) The advantage is that it allows very fast responses to events to be implemented.
  3) The disadvantage is that it is complex to program a difficult to validate.

4)



15. Explain the term Ethnography

Ethnography is an observational technique that can be used to understand operational processes and help derive support requirements for these processes. An analyst immerses himself or herself in the working environment where the system will be used. The day-to-day work is observed and notes made of the actual tasks in which participants are involved. The value of ethnography is that it helps discover implicit system requirements that reflect the actual ways that people work, rather than the formal processes defined by the organization.

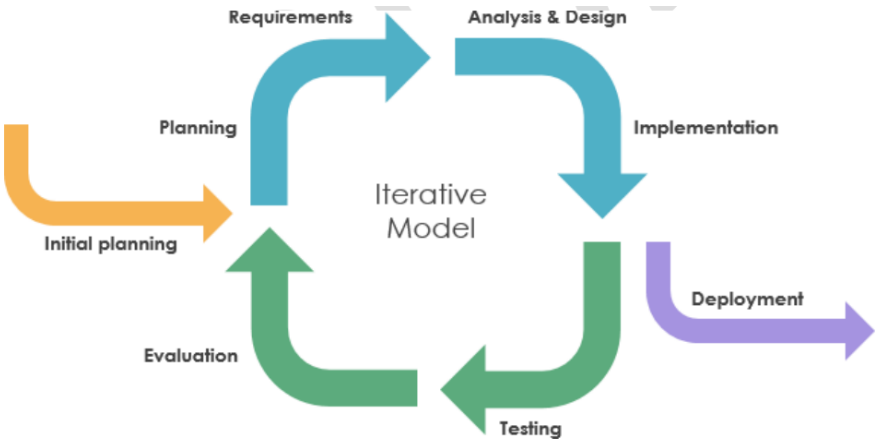
People often find it very difficult to articulate details of their work because it is second nature to them. They understand their own work but may not understand its relationship to other work in the organization. Social and organizational factors that affect the work, but which are not obvious to individuals, may only become clear when noticed by an unbiased observer. For example, a work group may self-organize so that members know of each other's work and can cover for each other if someone is absent. This may

not be mentioned during an interview as the group might not see it as an integral part of their work.

Ethnography is particularly effective for discovering two types of requirements:

1. Requirements that are derived from the way in which people actually work, rather than the way in which process definitions say they ought to work. For example, air traffic controllers may switch off a conflict alert system that detects aircraft with intersecting flight paths, even though normal control procedures specify that it should be used. They deliberately put the aircraft on conflicting paths for a short time to help manage the airspace. Their control strategy is designed to ensure that these aircrafts are moved apart before problems occur and they find that the conflict alert alarm distracts them from their work.
2. 2. Requirements that are derived from cooperation and awareness of other people's activities. For example, air traffic controllers may use an awareness of other controllers' work to predict the number of aircrafts that will be entering their control sector. They then modify their control strategies depending on that predicted workload. Therefore, an automated ATC system should allow controllers in a sector to have some visibility of the work in adjacent sectors.

16. Explain the concept of Iterative development model.



An iterative life cycle model does not attempt to start with a full specification of requirements. Instead, development begins by specifying and implementing just part of the software which can then be reviewed in order to identify further requirements. This process is then repeated producing a new version of the software for each cycle of the model.

The various phases of Iterative model are as follows:

1) Requirement gathering & analysis: In this phase, requirements are gathered from customers and check by an analyst whether requirements will fulfil or not. Analyst checks that need will achieve within budget or not. After all of this, the software team skips to the next phase.
2) Design: In the design phase, a software solution to meet the requirements is designed. This may be a new design or an extension to an earlier design.
3) Implementation and Test phase: The software is coded, integrated and tested
4) Review phase: The software is evaluated; the current requirements are reviewed and changes and additions to requirements are proposed.
5) The key to successful use of an iterative software development life cycle is rigorous validation of requirements and verification of each version of the software against those requirements within each cycle of the model
6) The first three phases of the iterative model is in fact an abbreviated form of sequential V model or Waterfall model of development. Each cycle of the model produces software that requires testing at the unit level, for software integration, for system integration and for acceptance testing.
7) As the software evolves through successive cycles test have to be repeated and extended to verify each version of the software.

Iterative model – Advantages

1. Testing and debugging during smaller iteration is easy.

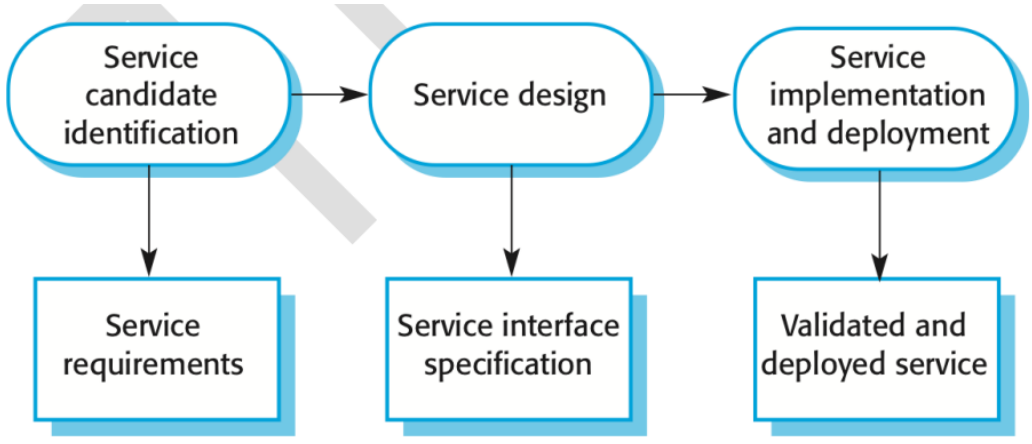
2. A Parallel development can plan.
3. It is easily acceptable to ever-changing needs of the project.
4. Risks are identified and resolved during iteration.
5. Limited time spent on documentation and extra time on designing

Iterative model – Disadvantages
1. It is not suitable for smaller projects.
2. More Resources may be required.
3. Design can be changed again and again because of imperfect requirements.
4. Requirement changes can cause over budget.
5. Project completion date not confirmed because of changing requirements.

17. Explain service engineering

Service engineering is the process of developing services for reuse in service-oriented applications. The service has to be designed as a reusable abstraction that can be used in different systems. Generally useful functionality associated with that abstraction must be designed and the service must be robust and reliable. The service must be documented so that it can be discovered and understood by potential users.

There are three logical stages in the service engineering process, as shown in Figure. These are as follows:



1. Service candidate identification, where you identify possible services that might be implemented and define the service requirements.
   Three fundamental types of service:
   • Utility services that implement general functionality used by different business processes.
   • Business services that are associated with a specific business function e.g., in a university, student registration.
   • Coordination services that support composite processes such as ordering.

2. Service design, where you design the logical and WSDL service interfaces. Involves thinking about the operations associated with the service and the messages exchanged.
   Interface design stages:
   • Logical interface design. Starts with the service requirements and defines the operation names and parameters associated with the service. Exceptions should also be defined.
   • Message design (SOAP). For SOAP-based services, design the structure and organization of the input and output messages. Notations such as the UML are a more abstract representation than XML. The logical specification is converted to a WSDL description.
   • Interface design (REST). Design how the required operations map onto REST operations and what resources are required.

3. Service implementation and deployment, where you implement and test the service and make it available for use. Programming services using a standard programming language or a workflow language. Services then have to be tested by creating input messages and checking that the output messages produced are as expected. Deployment involves publicizing the service and

installing it on a web server. Current servers provide support for service installation.

18. Write a note on SRS.

The software requirements document (sometimes called the software requirement specification or SRS) is the official statement of what the system developers should implement. It should include both the user requirements for a system and a detailed specification of the system requirements. In some cases, the user and system requirements are integrated into a single description. In other cases, the user requirements are defined in an introduction to the system requirements specification. If there are a large number of requirements the detailed system requirements may be presented in a separate document.
The requirements documents have a diverse set of users ranging from the senior management of the organization that is paying for the system to the engineers responsible for developing the software. The diversity of possible users means that the requirements has to be a compromise between

Users of a requirement document	**System customers**	Specify the requirements and read them to check that they meet their needs. Customers specify changes to the requirements.
	Managers	Use the requirements document to plan a bid for the system and to plan the system development process
	System Engineers	Use the requirements to understand what system is to be developed
	System Test Engineers	Use the requirements to develop validation tests for the system
	System Maintenance Engineers	Use the requirements to understand the system and the relationship between its parts

communicating the requirements to customers, defining the requirements in precise detail for developers and testers including information about possible system evolution.

The most widely known standard is IEEE/ANSI 830-1998 (IEEE,1998).

This IEEE standard suggests the following structure for requirements documents:

1. Introduction
    - 1.1 Purpose of the requirements document
    - 1.2 Scope of the product
    - 1.3 Definitions, acronyms and abbreviations
    - 1.4 References
    - 1.5 Overview of the remainder of the document
2. General description
    - 2.1 Product perspective
        - 2.1.1 System interfaces
        - 2.1.2 User interfaces
        - 2.1.3 Hardware interfaces
        - 2.1.4 Software interfaces
        - 2.1.5 Operations
    - 2.2 Product functions
    - 2.3 User characteristics
    - 2.4 General constraints, Assumptions and dependencies
3. Specific requirements
    - 3.1 External interface requirements
    - 3.2 Functional requirements
    - 3.3 Performance requirements
    - 3.4 Design constraints
    - 3.5 Logical database requirement
    - 3.6 Software system attributes
4. Other requirements
5. Appendices
6. Index

19. Describe various Management activities

Software systems are often new and technically innovative. Engineering projects (such as new transport systems) that are innovative often also have schedule problems. It is impossible to write a standard job description for a software project manager.
However, most managers take responsibility at some stage for some or all of the following activities:

1. Proposal writing: The first stage in a software project may involve writing a proposal to win a contract to carry out an item of work. The proposal describes the objectives of the project and how it will be carried out. It usually includes cost and schedule estimates and justifies why the project contract should be awarded to a particular organization or team

2. Project planning: Project managers are responsible for planning, estimating and scheduling project development, and assigning people to tasks. They supervise the work to ensure that it is carried out to the required standards and monitor progress to check that the development is on time and within budget.

3. Risk management: Project managers have to assess the risks that may affect a project, monitor these risks, and take action when problems arise.

4. People management: Project managers are responsible for managing a team of people. They have to choose people for their team and establish ways of working that lead to effective team performance.

5. Reporting: Project managers are usually responsible for reporting on the progress of a project to customers and to the managers of the company developing the software. They have to be able to communicate at a range of levels, from detailed technical information to management summaries. They have to write concise, coherent documents that

abstract critical information from detailed project reports. They must be able to present this information during progress reviews.

20. Write a note on Release testing

Release testing is the process of testing a particular release of a system that is intended for use outside of the development team. The primary goal of the release testing process is to convince the customer of the system that it is good enough for use. Release testing, therefore, has to show that the system delivers its specified functionality, performance and dependability, and that it does not fail during normal use. Release testing is usually a black-box testing process where tests are only derived from the system specification. Release testing is a form of system testing. Important differences:

- A separate team that has not been involved in the system development, should be responsible for release testing.
- System testing by the development team should focus on discovering bugs in the system (defect testing). The objective of release testing is to check that the system meets its requirements and is good enough for external use (validation testing).

Requirements-based testing involves examining each requirement and developing a test or tests for it. It is validation rather than defect testing: you are trying to demonstrate that the system has properly implemented its requirements.

Scenario testing is an approach to release testing where you devise typical scenarios of use and use these to develop test cases for the system. Scenarios should be realistic and real system users should be able to relate to them. If you have used scenarios as part of the requirements engineering process, then you may be able to reuse these as testing scenarios.

Part of release testing may involve testing the emergent properties of a system, such as performance and reliability. Tests should reflect the profile of use of the system. Performance tests usually involve planning a series of tests where the load is steadily increased until the system performance becomes unacceptable. Stress testing is a form of performance testing where the system is deliberately overloaded to test its failure behaviour.