Ch-1

1) Difference between classical waterfall model and iterative waterfall model

Ans

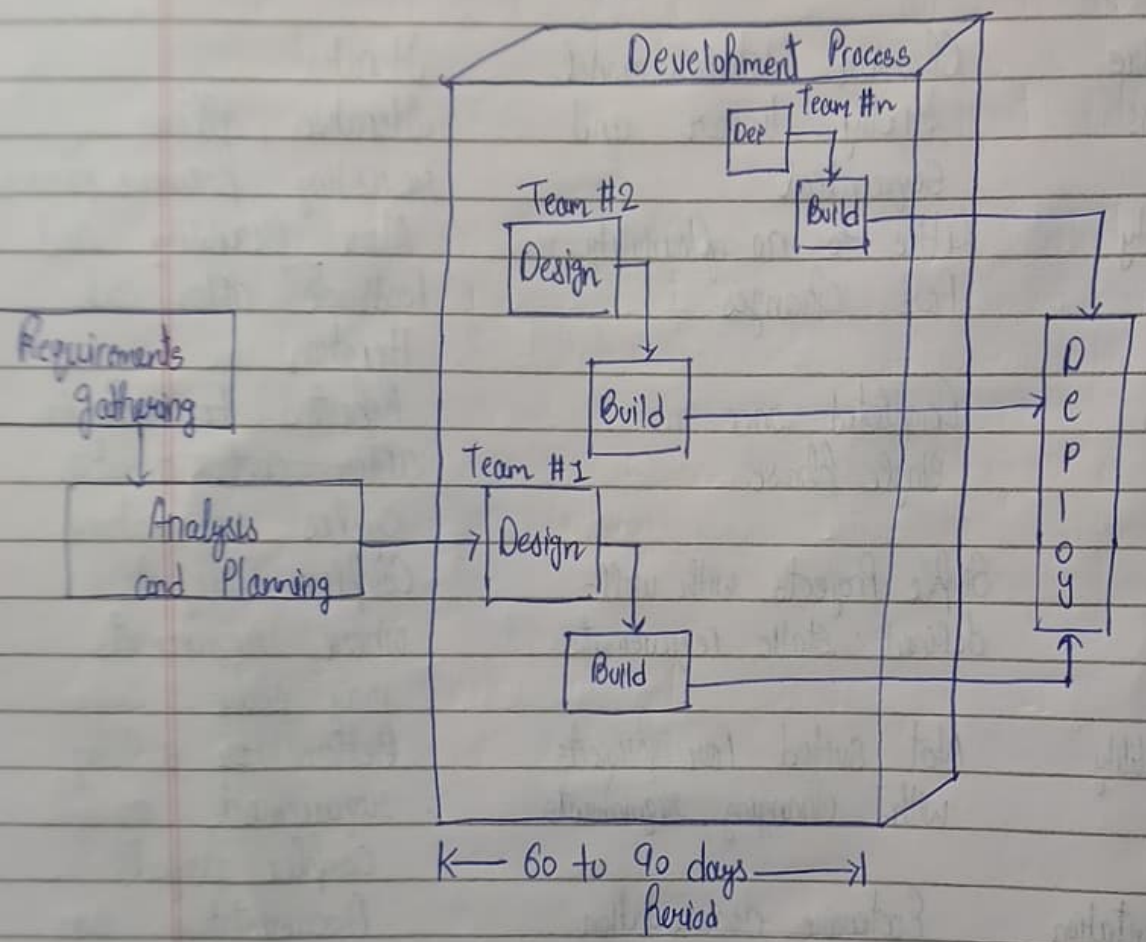
Feature	Classical Waterfall model	Iterative
1) Process Flow	Strictly linear and Sequential	Iterative, allows revisiting previous phases
2) Flexibility	Little to no flexibility for changes	More flexibility and feedback after each iteration
3) Phases	Completed once in single phase	Repeated Feedback in after each iteration cycles or iterations
4) Best For	Simple Projects with well-defined, stable requirements	Complex Projects where requirements may evolve
5) Adaptability	Not suited for projects with changing requirements	Better for evolving requirements or complex projects
6) Documentation	Extensive documentation at each stage before proceeding	Documentation can be adjusted after each iteration

2) How does RAD model work ? Discuss the pros and cons of RAD model

Ans The RAD Model is a type of incremental process model in which there is extremely short development cycle.
→ When the requirements are fully understood and the

Component based Construction approach is adopted then the RAD model is used.

→ Various phases in RAD are Requirements Gathering, Analysis and Planning, Design, Build or Construction and finally Deployment



1) Requirement Gathering: In the requirements gathering phase the developers communicate with the users of the system and understand the business process and requirements of the software system.
2) Analysis and Planning: During this phase, the analysis on the gathered requirements is made and a planning for various software development activities is done.
3) Design Phase: During this phase various models

are created. Those models are Business model, data model and process model

4) Build Phase: The build is an activity in which using the existing software components and automatic code generation tool the implementation code is created for software

o Pros
→ Quick delivery with rapid prototyping
→ Continuous feedback ensures the system meets user needs
→ Risks are Reduced because frequent iterations catch issues early
→ It is flexible

o Cons
→ It requires multiple teams or large number of people to work on the scalable projects
→ The projects using RAD model requires heavy resources
→ If commitment is lacking then RAD projects will fail
→ The projects using RAD model find it difficult to adapt new technologies

Q-3) Differentiate Product and Process

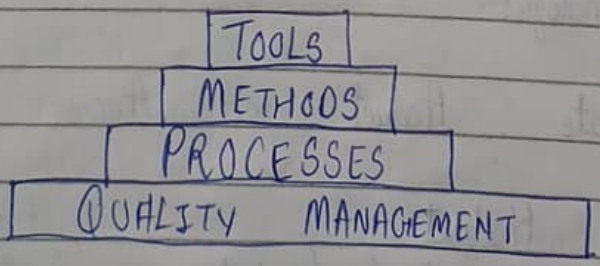
Ans

Product	Process
1) Product is the final production of the project	1) Process is a set of sequence steps that have to be followed to create a project
2) A product focuses on the final result	2) Whereas, the process is focused on completing each step being developed

3) In the case of products, the firm guidelines are followed	3) In contrast, the process consistently follows guidelines
4) A product tends to be short-term	4) long-term
5) The main goal of the product is to complete the work succesfully	5) While the purpose of a process is to make quality project better
6) Dependent on the completion of a specific product	6) Dependent on the implementation and management of tasks and workflows
7) Eg: A software application (eg. a car, a building	7) Eg: The steps involved in software development, manufacturing processes

Q-4 What are different layers of SE? Draw and explain it in short

Ans



1) Quality Management : It is the backbone of software engineering technology
→ An engineering approach must have focus on quality

2) Process Layer
→ It defines a framework with activities for effective technology delivery of software engineering technology

3) Method
→ It provides technical how-to's for building software
→ It encompasses many tasks including, communication, requirement analysis, design modeling, program construction testing and support

4) Tools
→ Software Engineering tools provide automated or semiautomated support for the processes and the methods
→ Computer aided Software Engineering (CASE) is the scientific application of a set of tools and methods

Q.5 Draw and explain the different phases of Waterfall Model

Ans Check Ch-1 notes

Q.6 What is Software Engineering? List down different myths for it

Ans Software Engineering is a discipline in which theories, methods and tools are applied to develop professional software

Myths related to Software Engineering are as follows:-
1) Using a collection of standards and procedures one can build software
2) Add more people to meet deadline of the project
3) Once the program is running then its over!

4) There is no need of documenting the software project, it unnecessarily slows down the development process
5) Even if the software requirements are changing continuously it is possible to accomodate these changes in the software

Q-7 Draw and explain Spiral Model with its advantages

Ans Check Ch-1 old notes

0 Advantages
1) Software is produced early in the software life cycle
2) Strong approval and documentation control
3) Risk handling is one of important advantages of the Spiral model
4) It is good for large and complex projects
5) Requirement changes can be made at every stage

Q-8 Compare Spiral Model with Prototype Model

Ans

	Prototype Model	Spiral Model
1)	A Prototype model is a software development model in which a prototype is built, tested and then refined as per customer needs	It is risk driven and is made with features of incremental waterfall, or evolutionary prototyping models
2)	It does not emphasize risk analysis	It takes special care about risk analysis and an alternative solution is undertaken

3	Large scale project is maintained	3) Low to medium project size is maintained
4	Cost-effective quality improvement is very much possible	4) Cost-effective quality improvement is not possible
5	Less costly due to the rapid creation of prototypes	5) Costly due to continuous iterations and risk analysis
6	High flexibility to change requirements	6) High flexibility to modify features and design based on feedback from prototype evaluation

Q-9 Compare Waterfall model with RAD model

Ans See Ch-1 old notes

Q-10 Define terms : Software Engineering, Process and Product

Ans See Ch-1 old notes

Q-11 Explain SE: A Layered technology

Ans See Ch-1 old notes

Q-12 Explain one of the evolutionary software process model with its pros & cons

Ans Check Ch-1 old notes for Spiral Model
Check Q-7 for advantages (pros)

Cons

1) It is not suitable for small projects
2) The complexity of spiral model can be more than the other sequential models
3) The cost of developing a product through spiral model is high
4) Documentation challenges: Frequent changes and iterations can lead to challenges in maintaining accurate and up-to-date documentation

Q-13 Distinguish between Process and Methods?

Ans See Ch-1 answers

Q-14 List the goals of SE

Ans See Ch-1 old notes

Q-15 Explain the Evolutionary and Incremental Model. What are the advantages and Disadvantages

Ans See Ch-1 old notes

Q-16 List and Explain various Software Development Myths and its reality

Ans

Ch-2 Agile Development

Q1 What is Agility? List down 12 principals of Agile Manifesto

Ans Check Ch-2

Q-2 Explain the merits and demerits of SCRUM

Ans Check Ch-2

Q-3 What is Extreme Programming

Ans Check Ch-2

Q-4 Define 1) Agile methods 2) Agile methods process

Ans 1) Agile methods: Frameworks like Scrum, Kanban, and XP that supports iterative, flexible development focused on customer needs

2) Agile Process: A development approach using short cycles and continuous feedback, allowing teams to adapt and deliver increments frequently

Q-5 Draw and explain different Phases of Agile Process Model

Ans

Q-6 Which are the Key asumptions that characterized a agile software process

Ans The Key asumptions that characterized any agile software process are

1) Requirements Change: Agile assumes requirements will evolve, so its adaptable
2) Customer Collaboration: Agile refers on regular input from customers for allignment
3) Incremental Delivery: Value is delivered in small, usable pieces over time
4) Self-Organized teams: Agile teams are enhavered to make decisions independently
5) Continuous Improvement: Agile teams constantly refine their processes for better results
6) Frequent Delivery Reduces Risk: Regular releases catch issues early

Q-7

Ch 3 Managing Software Model

Q) What are the different risk identification methods? Explain any of them in brief

Ans
(1) Brainstorming
(2) SWOT analysis
(3) Casual Mapping
(4) Flowchart Method
(5) Checklists
(6) Assumption Analysis

(1) Brainstorming: Brainstorming is a group activity where team members generate a list of potential risks in an open and unstructured way. It encourages creativity and free-flowing ideas, allowing team members to think of a wide range of risks. This method helps in uncovering risks that might not be obvious individually but are identified through collective thinking

Q-2 Explain different Project size estimation techniques

Ans Different Project size estimation techniques are as follows:-
1) Function Point Analysis (FPA)
2) Lines of Code (LOC)
3) Use Case Points
4) Expert Judgement
5) Analogy-Based Estimation
6) COCOMO (Constructive Cost Model)
7) Top-down and Bottom-up Estimation

1) Function Point Analysis (FPA)
→ Measures the functionality delivered by the software based on inputs, outputs, user interactions and files
→ FPA assesses the software's complexity and functionality to determine project size in function points, which are then translated into effort and cost

2) Lines of Code (LOC)
→ Estimates project size based on the total number of lines of code
→ By assessing the codebase size, it calculates effort and time needed based on historical productivity data per LOC

3) Use Case Points
→ This technique involves estimating the project size based on the number of use cases that the software must support
→ Use case points consider factors such as the complexity of each use case, the number of actors involved, and the number of use cases.

4) Expert Judgement
→ In this technique, a group of experts in the relevant field estimates the project size based on their experience and ~~estimate~~ expertise.
→ This technique is often used when there is limited information available about the project

5) Analogy Based Estimation
→ This technique involves estimating the project size based on the similarities between the current

Project and Previously completed Projects. This technique is useful when historical data is available for similar Projects

6) COCOMO (Constructive Cost Model)
→ It is an algorithmic model that estimates effort, time and cost in software development Projects by taking into account several different elements

7) Top-Down and Bottom-Up Estimation
- Top-Down : Begins with a high-level estimate and breaks down the Project to refine estimates for each component
- Bottom-Up : Estimates each part of the Project individually and then aggregates for total

Q-3 Why Project Scheduling is required ? Discuss merits and demerits of any one Project scheduling technique

Ans Project Scheduling is required because it provides a roadmap for completing a Project within a specified time and budget
→ It allows teams to understand the timeline, allocate resources effectively and coordinate tasks to meet objectives
→ Scheduling also helps in anticipating and addressing potential risks, ensuring that Project milestones are achieved efficiently

1) Critical Path Method
⇒ Merits
→ CPM provides a clear timeline and helps identify the shortest time required to complete the Project

→ It identifies the most critical tasks, allowing project managers to focus resources and attention where delays could impact the project
→ Resources can be effectively allocated to ensure critical tasks are completed on time

o Demerits
→ For very large projects with numerous tasks, CPM charts can become overly complex and challenging to interpret
→ Non-critical tasks can be overlooked, leading to bottlenecks if these tasks encounter unforseen delays
→ CPM assumes that all tasks have a single duration which may not be realistic given the uncertainties in project

Q-4 Define: Risk Identification, Risk Refinement and Risk Mitigation

Ans 1) Risk Identification: Risk identification can be defined as the efforts taken to specify threats to the project plan

2) Risk Refinement: After analyzing potential risks, risk refinement involves analyzing and categorizing each risk in more detail to assess its likelihood and potential impact

3) Risk Mitigation: Risk mitigation is the development and implementation of strategies to reduce the likelihood or impact of identified risks

Page No.
Date

Q.5 Explain Software Metrics used for software cost estimation

Ans (1) Lines of Code (LOC)

- Defination: LOC is a metric that counts the number of lines of code in software
- Usage: It is one of the simplest measures and helps estimate effort and cost based on the amount of code to be written

(2) Function Point (FP) analysis

- Defination: Function Point measures the software's functionality from the user's perspective. It is based on assessing the software's functions, inputs, outputs, data files, and user interactions
- Usage: FP is language - and technology - independent, making it useful for comparing different projects. It provides a more reliable estimate than LOC, especially for projects focused on user interactions

3) Use Case Points (UCP)

- Defination: UCP is a metric based on the use case model of the system, assesing the number and complexity of use cases
- Usage: It provides estimates based on the functional requirements of the system. UCP is particularly used for object-oriented development and is easier to adapt to agile environments

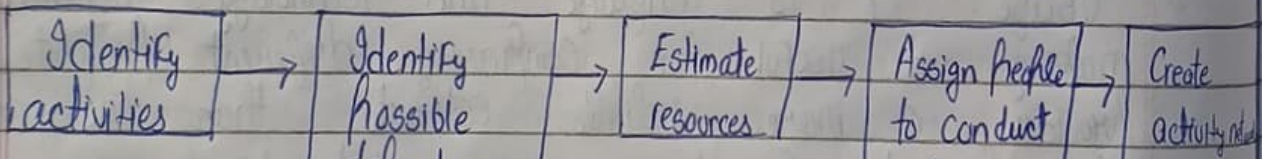
4) Process Based Estimation

- Defination: Process-based estimation involves breaking down a project into specific tasks or activities based on defined process and estimating the effort required for each task

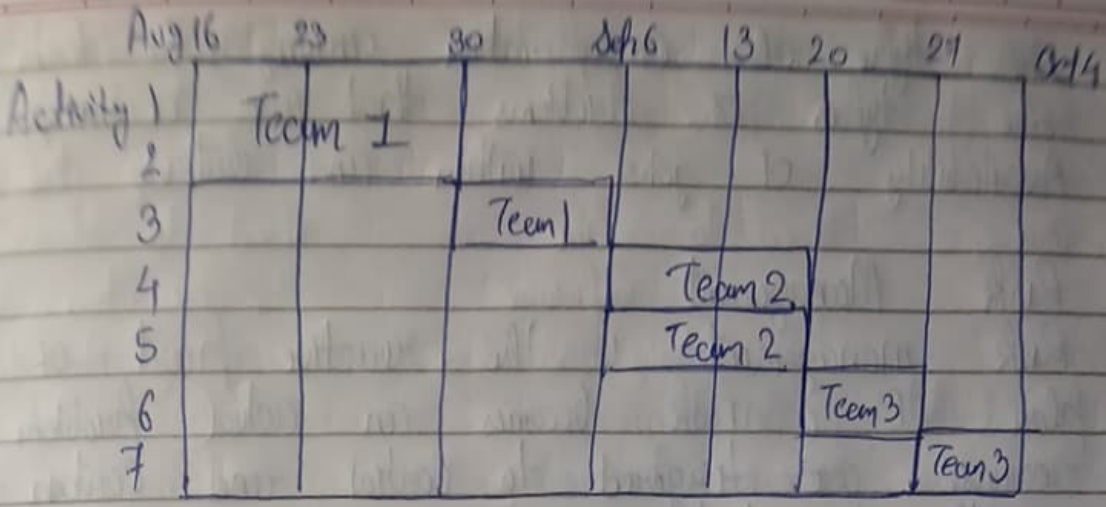
- Usage: It is commonly used by identifying key factors from the project scope and estimating efforts required for each task

Q.5 Explain Project Scheduling Process and Gantt Chart in detail

Ans While scheduling the project, the manager has to estimate the time and resource of the project.
→ During the project scheduling the total work is separated into various small tasks and time required for each activity must be determined by the project manager



- Gantt Chart
  A gantt chart, commonly used in project management is one of the most popular and useful ways of showing activities displayed against time
→ On the left of the chart is a list of the activities and along the top is a suitable time scale
→ Each activity is represented by a bar, the position and length of the bar reflects the start date duration and end date of the activity.



Q-1 Explain RMMM Plan

Ans RMMM Stands for risk mitigation, monitoring and management. There are three issues in strategy for handling the risk is

1) Risk Mitigation
2) Risk Monitoring
3) Risk Management

1) Risk Mitigation

→ Risk mitigation means preventing the risk to occur (risk avoidance)

→ For each identified risk, preventive measures are planned, such as allocating extra resources, building redundancies or scheduling additional testing

2) Risk Monitoring

→ In Risk Monitoring process following things must be monitored by the project manager

1) The approach or the behaviour of the team members as presence of project varies
2) The degree in which the team performs with the spirit of "team-work"

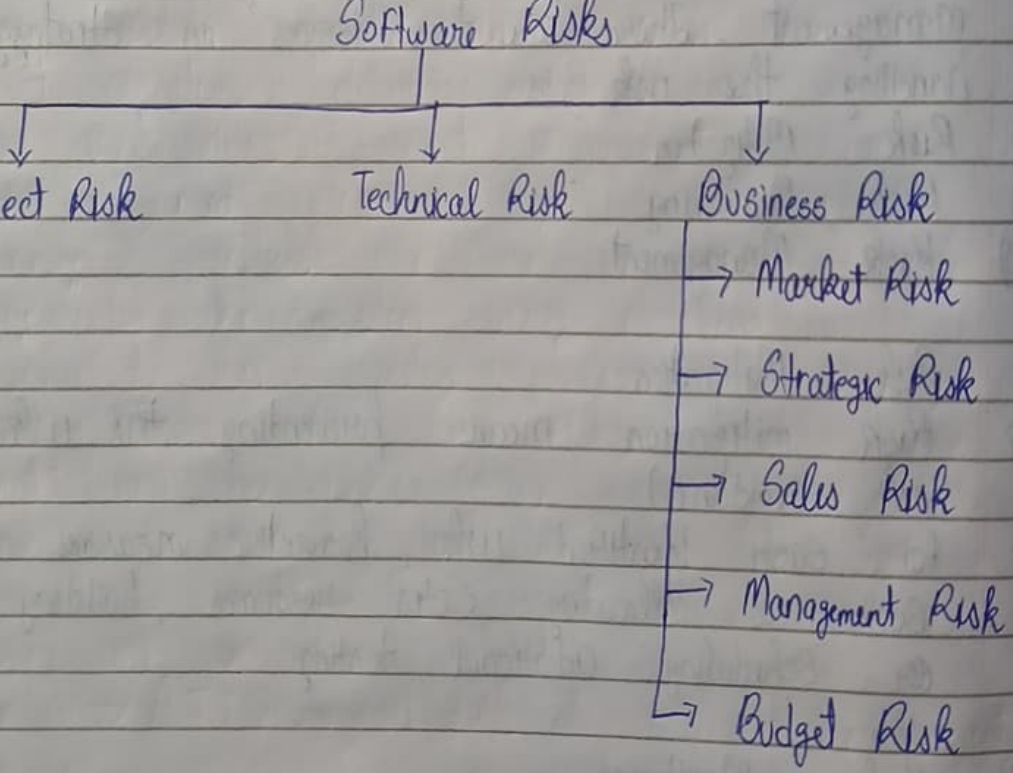
3) the type of co-operation among the team members
4) the types of problems that are occuring
5) Availability of jobs within and outside the organisation

Q-7) Risk Management

→ Risk management is the reactive part of the RMMM plan. If a risk becomes an actual problem, predefined actions are triggered to control and contain the impact on the project

Q-8 List and Explain different categories of Software Risks

Ans



1) Project Risk

→ Project risk arise in the Software development process when they basically affect budget, schedule, staffing resources and requirements

2) Technical risk
→ These risks affect quality and timeliness of the project. It Technical risks occur when problem becomes harder to solve

3) Business Risk
(i) Market Risk - When a quality software product is built but if there is no customer for this product then it is called market risk
(ii) Strategic Risk :- When a product is built and if it is not following the company's business policies then such a product brings strategic risks
(iii) Sales risk : When a product is built but how to sell is not clear then such a situation brings sales risk
(iv) Management Risk : When senior management or the responsible staff leaves the organization then management risk occurs
(v) Budget Risks : Losing the overall budget of the project is called budget risk

Q.3 Describe the difference between process and project

Ans

Aspect	Process Metrics	Project Metrics
1) Definition	Process metrics measure the efficiency of software development process	It assess the status and progress of a specific project
2) Focus	On improving and optimizing the software development process itself	On monitoring the projects progress, performance and outcomes
3) Scope	Concerned with long-term trends and patterns over multiple projects	Concerned with specific project or its current phase

4)	Impact	Helps in long-term improvements	Helps in tracking project health and making real-time decisions
5)	Usage	Used to refine processes, improve productivity and reduce defects	Used to manage and control the execution of a project ensuring it stays on track
6)	Example	→ Defect Density → Code Complexity	→ Project Cost → Schedule Variance

Q-10 Explain the W5HH Principle

Ans Barry Boehm suggested an approach for addressing project objectives, deciding milestones and schedules, roles and responsibilities, project management, technical approaches and required resources by W5H principle
Following questions are there in W5HH Principle

1) Why is the system being developed
Answer to this question assesses the validity of business reasons for the software work
2) What will be done? (ATTO)
Answer to this question will help the software team member to identify the project tasks and milestones
3) When will be done
The → ATTO will help to prepare the project schedule
4) Who is responsible for a function?
The ATTO will define the roles and responsibilities required to develop the system
5) Where are they organisationally located?
All the roles cannot be defined within the software team itself. There are customer users and other

stakeholders holding some responsibilities

2) How to do the job technically with proper management ATIQ will help to establish the technical strategy about the project

3) How much of each resource required the answer will be useful for deriving the project estimate or cost of the project

Q-11 Explain the difference between an error and a defect

Ans

Aspect	Error	Defect
1) Defination	A human mistake made during development	A flaw in the software cause by an error
2) Cause	Caused by incorrect coding, design or logic	Caused by errors, resulting in software issues
3) Detection	Detected during development	Detected during tested or after release
4) Impact	Doesn't directly affect functionality unless it leads to defect	Affects software behaviour, causing malfunction
5) Example	Incorrect variable type or logic in code	Software crashes or produces wrong output
6) Fixing Responsibility	Fixed by the developer during coding	Fixed by the development team during testing or maintanence

Q-12 Explain the four P's of software project management

Ans

4P's are
1) Product
2) Process
3) People
4) Project

1) Product
→ As the name inferred, this is the deliverable or the result of the project
→ The project manager should clearly define the product scope to ensure a successful result and control the team members
→ The product can consist of both tangible or intangible such as shifting the company to a new place or getting a new software in a company

2) Process
→ In every planning, a clearly defined process is the key to the success of any product
→ The process has several steps involved like documentation phase, implementation phase, deployment and interaction phase

3) People
→ We need to have a good team in order to save our time, cost and effort. Some assigned roles in software project planning are project manager, team leaders, stakeholders analysts and other IT professionals

4) Project
→ The last and final P in software project planning is Project

→ In this phase, the Project manager plays a critical role. They are responsible to guide the team members to achieve the Project's target and objectives, helping & assisting them with issues, checking on cost and budget, and making sure that the Project stays on track with the given deadlines.

Q.3 What are the elements of Analysis Model?

Ans Not here in Ch-4

Q.4 Discuss the concept of risk assesment and risk control

Ans 1) Risk assesment

Risk assesment is the process of identifying, analyzing and evaluating risks that could affect the success of a project or the quality of a product

→ It helps project managers and teams understand potential threats, uncertainties, and challenges that may arise during the lifecycle of project

o Key steps in Risk Assesment

(i) Risk Identification: Identifying potential risks that may affect the project, such as technical challenge, scope changes or security threats

(ii) Risk Analysis: Evaluating the likelihood of each risk occuring and estimating its potential impact on the project

(iii) Risk Evaluation: Prioritizing the risks based on their severity and probability of occurence, allowing the team to focus on the most critical risks first

2 Risk Control

Risk Control refers to the purpose of developing strategies and actions to manage and mitigate identified risks

Key strategies in Risk Control

- Risk Avoidance : Modifying plans or objectives to eliminate the risk entirely
- Risk Mitigation : Reducing the likelihood or impact of the risk
- Risk Transfer : Shifting the risk to a third party, such as outsourcing a task to reduce risk exposure or purchasing insurance
- Risk Acceptance : Acknowledging the risk and accepting it without any mitigation, usually when the cost of mitigation exceeds the potential impact of risk

Q-5 Explain how Breakdown structure is used in SE. Discuss how software project scheduling helps in timely release of product

Ans

Page No.
Date

Ch-4 Requirement Analysis and Specification

Q1 How does Functional Requirements differ from Non-Functional Requirements

Ans

Aspect	Functional Requirements	Non-Functional Requirements
1) Definition	Describes what the system should do i.e specific functionality or tasks	Describes how the system should perform ie system attributes or quality
2) Purpose	Focuses on behaviour and features of system	Focuses on performance, usability, and other quality attributes
3) Scope	Defines the actions and operations of the system	Defines constraints or conditions under which the system must operate
4) Documentation	Typically documented in use cases, functional specifications etc	Documented through performance criteria, technical specifications etc
5) Dependency	Determines what the system must do to meet user needs	Depends on how well the system performs the required tasks
6) Examples	User authentication, data Input/Output, transaction processing	Scalability, security, response time, reliability, maintainability

Q-2 What is Requirment Engineering ? List the Functional and Non-Functional Requirements for Blood Bank Management System

Ans Requirement Engineering is the process of gathering, analyzing, documenting and managing the requirements

for a system software.

→ It ensures that the system meets the needs and expectations of stakeholders and provides a clear understanding of the functional and non-functional aspects of the system.

The Process involves several stages

1) Requirements Elicitation : Collecting requirements from stakeholders
2) Requirements Analysis : Analyzing and refining the gathered requirements
3) Requirements Specification : Documenting the requirements in a structured format
4) Requirements Validation : Ensuring the requirements meet the stakeholder needs and are feasible
5) Requirements Management : Managing and tracking changes to the requirements throughout the software development lifecycle

0 Functional Requirements

1) Blood Donor Registration
2) Blood Inventory Management
3) Blood Donation Tracking
4) Reporting
5) Request for blood
6) Alerts and Notifications

0 Non-Functional Requirements

1) Performance
2) Security
3) Scalability
4) Availability
5) Usability
6) Compliance

Q.3 Differentiate Verification and Validation

Ans

Aspect	Verification	Validation
1) Definition	Verification refers to the set of activities that ensure software correctly implements the specific function	Validation refers to the set of activities that ensure that the software that has been built is traceable to customer requirements
2) Focus	It includes checking documents, designs, codes and programs	It includes testing and validating the actual product
3) Type of Testing	Verification is the static testing	Validation is the dynamic testing
4) Execution	It does not include execution of code	It includes the execution of the code
5) Timing	It comes before validation	It comes after verification
6) Performance	Verification finds about 50 to 60% of the device	Validation finds about 20 to 30% of the defects

Q.4 What are Functional Requirements ?

Ans See Ch-3 old notes

Q.5 What are Non-Functional requirements ?

Ans See Ch-4 old notes

Q.6 State and Explain the Requirement engineering tasks in detail

Ans Requirement engineering process performs seven distinct functions

1) Inception
2) Elicitation
3) Elaboration
4) Negotiation
5) Specification
6) Validation
7) Requirements Management

1) Inception
→ This is the first phase of the requirements analysis process
→ This phase gives an outline of how to get started on a project
→ A basic understanding of the problem is gained and the nature of the solution is addressed
→ Effective communication is very important in this stage as this is foundation as to what has to be done further

2) Elicitation
→ This phase focuses on gathering detailed requirements from stakeholders. The requirements define the project purpose and scope, but challenges can arise
- Scope Issues: Requirements might be too detailed, vague or impractical
- Understanding Problems: Miscommunication or ambiguity between the customer and developer can lead to incorrect requirements
- Volatility: Requirements may change over the time, affecting the project timeline and resources

3) Elaboration
→ In this phase, the team refines and expands upon the gathered requirements
→ This phase aims to develop a clear and structured view of the system functionality, enabling a deeper understanding of complex requirements before moving forward

4) Negotiation
→ Here, the focus is on reaching an agreement between the developer and customer on priorities, resources and scope
→ Customers are asked to prioritize requirements, and conflicts are resolved within business constraints
→ Risks, delivery timelines, costs and resource availability are discussed to set realistic expectations for both parties

5) Specification
→ In this phase requirements are documented in various formats such as written documents, models, use-cases

or Prototypes.

→ the specification serves as a clear reference for guiding the system's design and development

6) Validation

→ The goal of this phase is to verify that the requirements are complete, accurate, and feasible.

→ Validation methods include requirements reviews, prototyping test case generation and automated consistency checks to ensure the requirements align with stakeholder expectations

7) Requirement Management

→ This final phase involves tracking and managing requirements throughout the project.

→ the team evaluates the impact of these changes on the overall system, prioritizes them and incorporates necessary adjustments to ensure successful project completion

Q-7 Draw and explain the diagram of translating the Requirements Model into the Design Model

Ans § Ch-5 Question One

Ch 5 Software Design

Q1 Why low coupling and high cohesion is one of the desired properties of software design?

Ans Low coupling and high cohesion are desired in software design because they create modular, maintainable and teasable systems.

→ Low coupling minimizes dependencies between modules making changes easier and reducing the impact of modifications

→ High cohesion ensures each module has a clear focused responsibility, improving readability and maintainability

→ Together they make the software more robust, scalable and adaptable

Q-2 Which are the important characteristics of having good software design

Ans 1) Modularity: Divides the system into distinct, manageable modules, each with a specific purpose
2) Low Coupling: Reduces dependancies between modules, making the system easier to modify and understand
3) High Cohesion: Ensures each module has a focused responsibility, improving clarity and maintainability
4) Scalability: Allows the system to handle growth in workload or feature complexity without major changes
5) Reusability: Enables components to be reused in different parts of the system or in other projects, saving development time
6) Readability: Makes the code easy to read and understand

which is essential and for maintenance and collaboration.
7) Flexibility: Supports easy modification and extension, enabling future enhancements with minimal network
8) Testability: Facilitates efficient testing of each component, improving software reliability and reducing bugs
9) Performance: Ensures the software runs efficiently, meeting necessary performance standards for users

Q-3 How does one design good user interface for the software?

Ans (1) Prioritize Simplicity: Make it intutive and easy to use, minimizing unnecessary elements
(2) Consistency: Use consistent fonts, colors and layouts across screens for a cohesive look and feel
(3) User-centred: Focus on user needs and workflows, ensuring the interface supports tasks effectively
(4) Clear Navigation: Make it easy for users to navigate, with clear menus and logical flows
(5) Feedback: Provide visual feedback for user actions, like loading or confirmation messages
(6) Accessibility: Ensure the UI is accessible to all users, including those with disabilities

Q-4 What is Coupling? What is Cohesion? Explain different types of Cohesion and Coupling with proper example

Ans ○ Coupling
→ Coupling effectively represents how the modules can be "connected" with other module or with the outside world

→ Coupling depends on the interface complexity between modules
→ The goal is to strive for lowest possible coupling among modules in software design
→ Various types of coupling are
1) Data Coupling: The data coupling is possible by parameter passing or data interaction
2) Control Coupling: The modules share related control data in control coupling
3) Common Coupling: In common coupling common data or a global data is shared among the modules
4) Content coupling: Content Coupling occurs when one module makes use of data or control information maintained in another module

0 Cohesion
→ With the help of cohesion the information hiding can be done
→ A cohesive module performs only "one task" in software procedure with little interaction with other modules. In other words cohesive module performs only one thing
0 Different types of cohesion are:
1) Coincidentally cohesive: The modules in which the set of tasks are related with each other loosely then such modules are called coincidentally cohesive
2) Logically cohesive - A module that performs the tasks that are logically related with each other is called logically cohesive
3) Temporal Cohesion: The module in which the tasks need to be executed in some specific time span is called temporal cohesion

4 Procedural Cohesion :- When processing elements of a module are related with one another and must executed in some specific order then such module is called Procedural Cohesive.

5 Communicational Cohesion :- When the processing elements a module share the data then such module is Communicational cohesion

Q-5 Define Coupling and Cohesion. What is difference between cohesion and coupling ?

Ans See Ch-5 old notes

Q-6 State the difference between Procedural Design and Object Oriented design

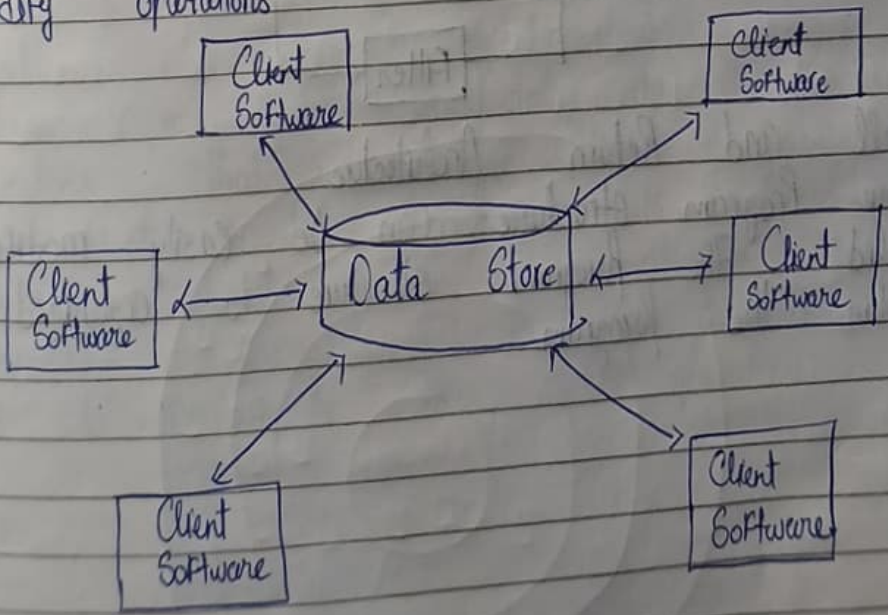
Ans

	Feature	Procedural Design	Object-Oriented Design
1)	Design Approach	Based on Functions and procedures	Based on objects and classes
2)	Data Handling	Data and Functions are separate	Data and methods are bundled in objects
3)	Modularity	Achieved through Function decomposition	Achieved through objects and class hierarchy
4)	Reusability	Limited reusability, mostly at function level	High reusability with inheritance and Polymorphism
5)	Abstraction	Procedural abstraction (using functions)	Data abstraction (using encapsulated objects)
6)	Ease of Maintenance	Can be harder to maintain for large systems	Easier to maintain due to modular structure

Q-7 What is an architectural design? Enlist different style and patterns of architecture

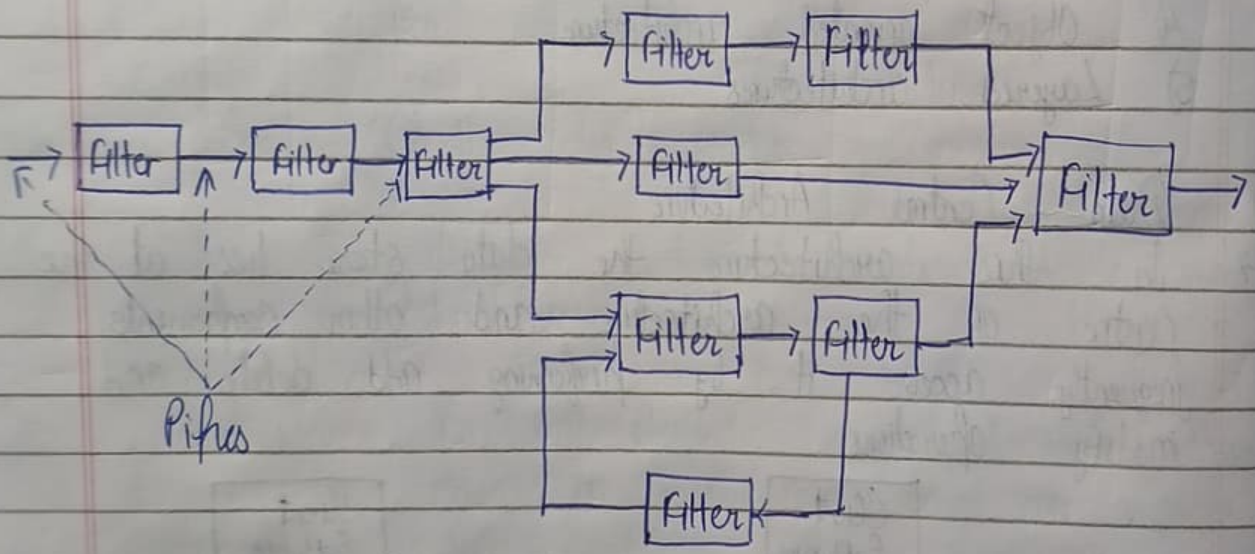
Ans Architectural design represents the structure of data and program components
The commonly used architectural styles are
1) Data centred architecture
2) Data flow architecture
3) Call and return architecture
4) Object oriented architecture
5) Layered architectures

1) Data Centred Architecture
→ In this architecture the data store lies at the centre of the architecture and other components frequently access it by performing add, delete and modify operations



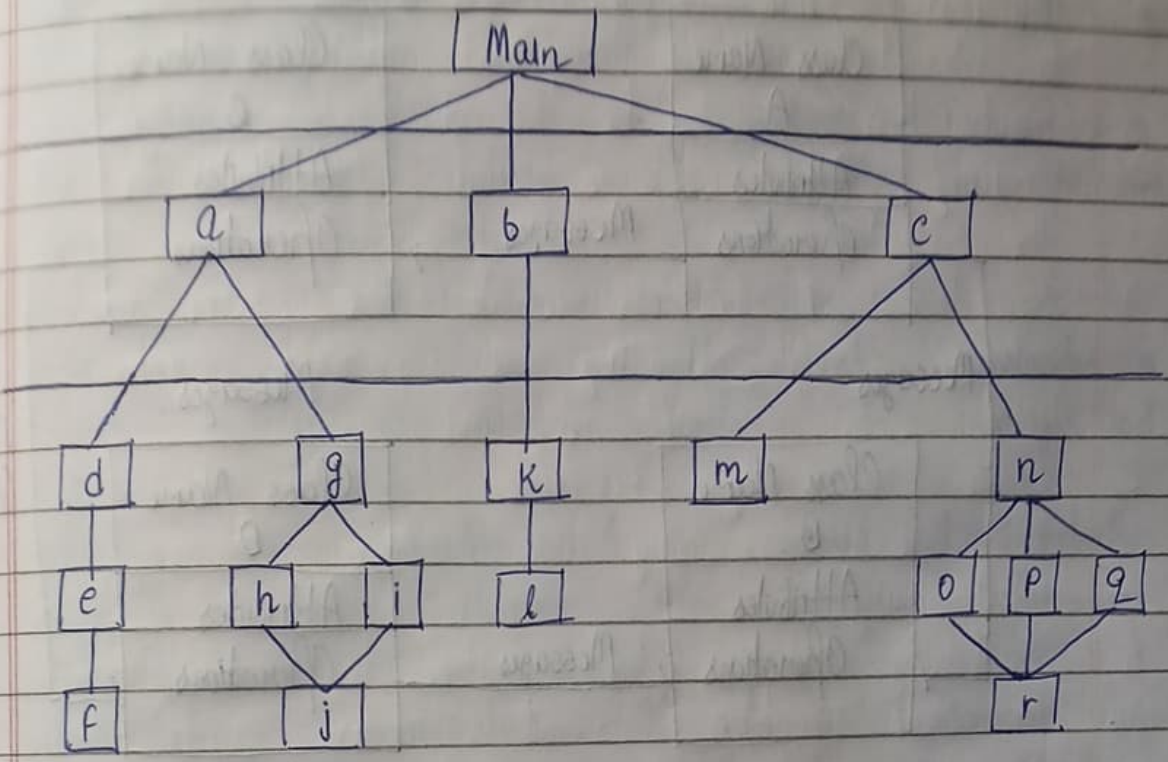
2) Data Flow Architecture

In this architecture series of transformations are applied to produce the output data. The set of components called filters are connected by pipes to transform the data from one component to another. These filters work independently without a bothering about the working of neighbouring filter



3) Call and Return Architecture

The Program structure can be easily modified or scaled. The Program structure is organized into modules within the Program

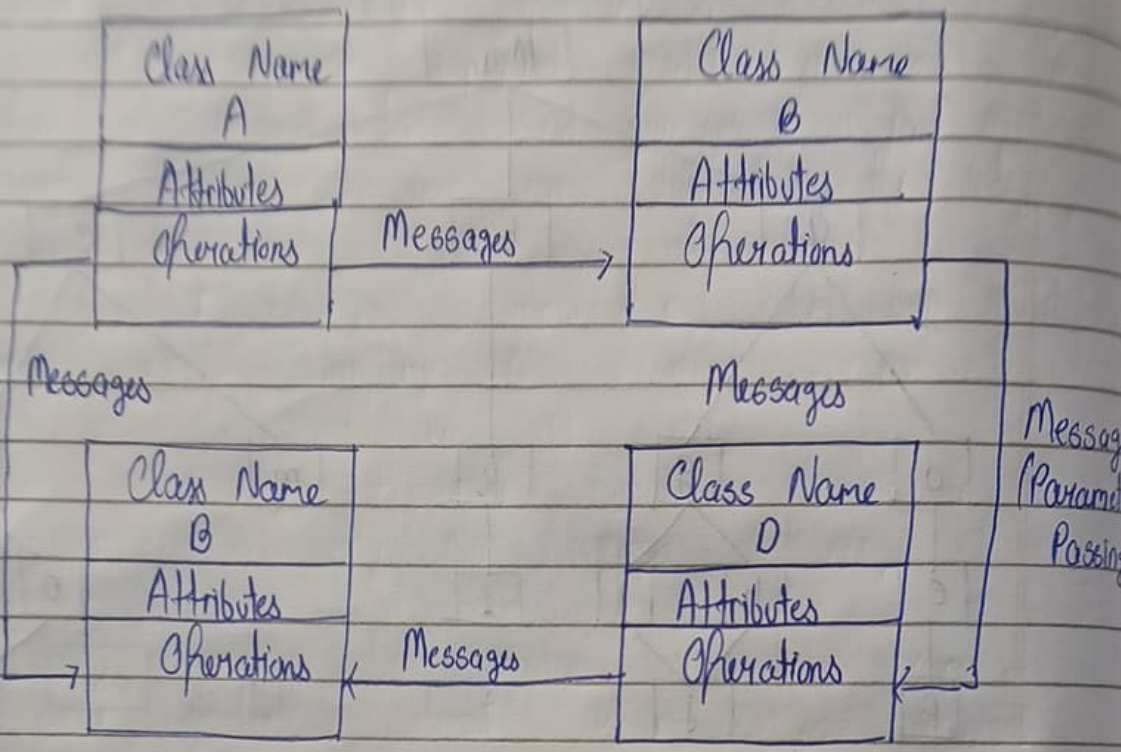


(A) Object Oriented Architecture

In this architecture the system is decomposed into number of interacting objects

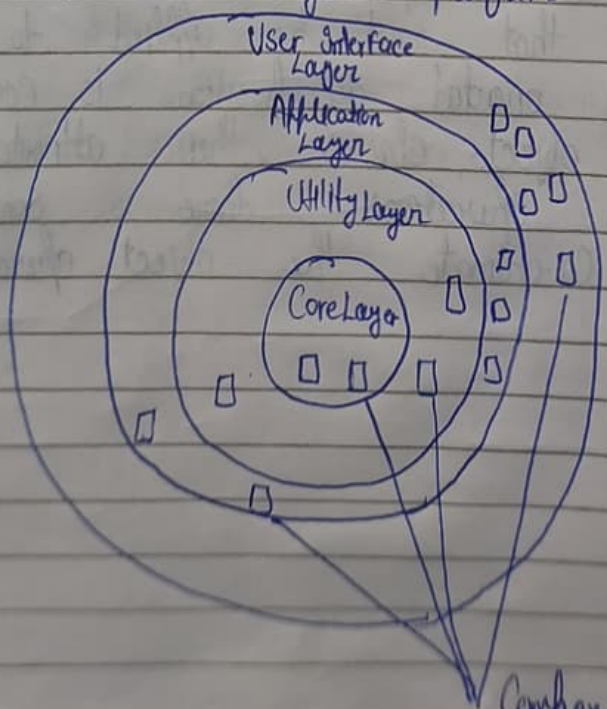
→ These objects encapsulate data and the corresponding operations that must be applied to manipulate the data

→ The object oriented decomposition is concerned with identifying objects, classes, their attributes and the corresponding operations. There is some control models used to co-ordinate the object operations



Layered Architecture

The layered architecture is composed of different layers. Each layer is intended to perform specific operations so machine instruction set can be generated. Various components in each layer perform specific operations



Q.8 What is Software Architecture and why it is important?

Ans Software Architecture is the high-level structure of a software system, defining its components, their relationships and how they interact.

Importance of Software Architecture

1) Foundation for Design and Development: It provides a structural framework, guiding developers in building and assembling components
2) Ensures System Quality: A well thought-out architecture supports essential attributes like scalability, performance, reliability and security
3) Facilitates Communication: It acts as a common language among stakeholders, helping developers, designers, and business teams understand the system
4) Enhances Maintainability: A good architecture makes it easier to modify and expand the system over time, ensuring it can evolve with changing requirements
5) Risk Mitigation: Architectural planning helps identify and address technical risks early, reducing the likelihood of costly rework later on

Q.9 Define design process. List the principles of a software design

Ans The Design Process in software engineering is a series of steps that developers follow to transform software requirements into a blueprint for constructing the software.

Principles of Software Design

1) Modularity: Break down the system into smaller, manageable and independent modules

2. Abstraction: Hide complex implementation details, allowing users to interact with a simplified model of the system.
3. Encapsulation: Keep data and functions within modules, limiting their exposure and protecting the integrity of the system.
4. Separation of Concerns: Divide the system into distinct sections, each handling a specific concern, to make it easier to understand and modify.
5. Single Responsibility Principle: Ensure each module or class has one responsibility or purpose, enhancing clarity and maintainability.
6. Scalability: Design the system to handle increased load or new features without needing major changes.

Q-10 Draw and explain the diagram of Translating the Requirements Model into the Design Model

Ans

Page No.
Date

Q-II Explain the Feasibility studies. What are the outcomes? Does it have either Implicit or explicit effects on Software requirement collection?

Ans Feasibility studies is an assessment process conducted at the early stages of a software project to evaluate if a proposed solution is viable, cost-effective and achievable within given constraints

Types of Feasibility Studies

1) Technical Feasibility
→ Evaluates whether the technology, tools and resources needed for the project are available or can be developed

2) Economic Feasibility
→ Assesses whether the project's financial benifits outweigh the costs.

3) Operational Feasibility
→ It evaluates the practical aspects, such as whether the system will integrate well with current processes and if users are likely to adopt it.

4) Schedule Feasibility: Estimates if the project can be completed within the timeline required or specified by stakeholders, considering time constraints and deadlines

0 Outcomes of a Feasibility Study
1) Go/No-Go Decision
2) Risk Assessment
3) Preliminary Project Scope
4) Budget and Timeline Estimates.

0 Effects on Software Requirement Collection
1) Explicit Effects
2) Implicit Effects

Ch-6 Software Coding & Testing

Q) How does the behaviour testing method work?

Ans Behaviour Testing focuses on testing the software's functionality based on expected user behaviour rather than the system's internal code structure. It emphasizes verifying that the software does what users expect in various scenarios

Q-7 What do you understand by Performance testing? What are the different types of Performance testing?

Ans Performance Testing

→ Performance testing evaluates the run time performance of the software, especially real time software

→ In performance testing resource utilization such as CPU Load, throughput, response time, memory usage can be measured.

→ For big systems (eg banking systems) involving many users connecting to servers (eg using internet) Performance testing is very difficult

→ Beta testing is useful for performance testing

Types of Performance testing

1) Load testing
2) Stress testing
3) Spike testing
4) Scalability testing

Q-3 How does the glass box testing method work? (write)

Ans Glass Box testing is a software testing method where the internal workings of the application are fully visible to the tester

→ Unlike black-box testing, which focuses on the system's outputs without knowledge of its internal code, glass box testing involves looking at the code and structure of the system to create test cases.

How Glass Box Testing works

1) Access to Internal Code
→ The tester has access to the source code, design documents and system architecture.
→ This allows the tester to understand how the system works and test individual parts of the code
2) Test Design
→ Test Cases are designed based on knowledge of the code, design documents, and system architecture
3) Test Execution
→ The test cases are executed, and the tester checks the outputs against expected results. Since the tester knows the internal code, they can identify potential issues

that are not easily detected by black-box methods

4 Bug Identification

If a test case fails, the tester can trace the error back to the specific code, making it easier to pinpoint and fix issues. This is particularly useful for finding problems in the underlying code that might not be obvious through functional testing

5 Test Coverage

7 Glass box testing ensures through test coverage by focusing on areas that might be missed with black box testing. Since the tester has visibility into the internal code, they can ensure code paths, conditions, and logic flows are tested thoroughly

Q-4 Differentiate between integration testing and System testing

Ans

Comparision	System testing	Integration testing
1) Basic	Tests the finished product	Validates the collection and interface modules
2) Performed	After integration system	After unit testing
3) Emphasis	On the behaviour of all module as a whole	System functionalities interface between individual modules
4) Covers	Functional as well as non-functional tests	Only functional testing
5) Executed	Only by test engineers	By test engineers as well as developers
6) Test Cases	Created to imitate real life scenarios	Build to stimulate the interaction between two modules

7) Examples — Testing the interaction between a database module and a business logic module — Testing the end-to-end functionality of an application

Q-5 What are the basic challenges in reuse of program.

Ans Here are the basic challenges in program reuse:
1) Compatibility Issues: Reusable components may not fit the new environment, requiring adjustments
2) Lack of Documentation: Poor documentation makes reused components harder to understand and integrate
3) Quality Assurance: Reused components might not meet the quality or security standards of the new project
4) Maintenance: Updates or fixes in reused components need to be managed to ensure compatibility
5) Contextual Differences: Components may need changes to suit a new application, reducing reuse benefits
6) Licensing Issues: Licensing restrictions on third-party components can limit reuse options.

Q-6 What is Software testing? Explain Black-Box and White Box Testing in details along with example

Ans Software testing is the process of evaluating a software application to identify defects, verify that it meets specified requirements and ensures it functions correctly.
→ Testing helps improve software quality, performance, security and usability by detecting issues before release

Page No.
Date

- Black Box testing
→ Black Box testing also called behavioral testing, it tests the software's functionality without knowledge of its internal code structure
→ Testers interact with the system from the outside, focusing on input-output behaviours rather than how the code works
Examples of Black-Box Testing
1) Equivalence Partitioning: Divide inputs into equivalent groups, testing one input from each group is assumed to cover the whole group. For eg, if an application accepts ages between 18-60, equivalence classes could be below 18, between 18-60, and above 60
2) Boundary Value Analysis (BVA): Test at the boundaries of input ranges. For instance, if an age input should be between 18-60, tests would include 17, 18, 60 and 61
3) Decision Table Testing: Map out complex input combinations and expected outputs in a table format to ensure all possible cases are tested
Eg: Testing a login function
→ Test Cases: Enter valid credentials, incorrect username, incorrect password, blank inputs
→ Expected Outcome: Valid credentials allow login, while invalid inputs trigger error messages

- White Box Testing
→ White Box testing, also called glass-box or structural testing is based on knowing the internal structure, logic and code of the application. Testers focus on testing specific code segments to verify that all paths, branches and conditions are working as intended

- Examples of White-Box Testing techniques
1) Statement Coverage: Ensures every line of code is executed atleast once during testing
2) Branch Coverage: Ensures that each branch (eg true/false conditions in if statements) is tested.
3) Path Coverage: Tests all possible paths in the code to catch potential errors in complex structures.

Code:

```
def is_even(number):
    if number % 2 == 0
        return True
    else:
        return False
```

- Test Cases: Run tests with numbers like 2, 3, 0, and negative numbers to check both branches (even and odd outcomes)
- Expected Outcome: Correct handling of even and odd inputs and valid return values.

Q.7 What is Software Testing? What is Role of a Software Tester? Compare Black Box and White Box testing

Ans Software Testing is the process of evaluating a software application to identify defects, verify that it meets specified requirements and ensures it functions correctly.

- Role of a Software Tester
1) Understanding Requirements: Reviewing and understanding the software requirements and identifying the key functions to be tested

2) Designing test Cases: Developing test cases based on requirements and use cases to cover different scenarios
3) Executing tests: Performing tests, including manual and automated tests and recording results to identify issues
4) Reporting and tracking Bugs: Documenting defects and tracking them through resolution, ensuring the software performs as expected
5) Regression Testing: Re-testing after bug fixes or updates to confirm changes haven't introduced new issues
6) Collaborating with the Team: Working closely with developers, product managers, and other stakeholders to ensure quality across the software development lifecycle

Aspect	Black Box testing	White Box testing
1) Focus	1) Tests functionality based on user requirements	1) Tests internal code structure and logic
2) Knowledge of Code	2) Not required	2) Required
3) Approach	3) Validates input-output behaviours and UI features	3) Analyzes code paths, conditions and data flow
4) Performed by	4) Testers, often non developers	4) Testers or developers with coding knowledge
5) Goal	5) Ensure software meets user requirements	5) Ensure code works as intended and has no hidden issues
6) Limitations	6) May miss errors in internal logic	6) Can be time-consuming and requires technical skills

Q-8 What is BVA ? Explain merits and demerits of it.

Ans Boundary Value analysis is done to check boundary conditions

→ A boundary value analysis is a testing technique in which the elements at the edge of the domain are selected and tested

→ Using boundary value analysis, instead of focusing on input conditions only, the test cases from output domain are also derived.

→ Boundary value analysis is a test case design technique that complements equivalence partitioning technique.

⇒ How BVA works

BVA tests input values at the boundaries

- Lower Boundary: Test just below, at, and just above the ~~minimum~~ minimum acceptable input
- Upper Boundary: Test just below, at, and just above the maximum acceptable input

For example, if an application accepts age between 18 and 60

- Test Cases : 17 (below), 18 (boundary), 19 (just above), 59 (just below), 60 (boundary), 61 (above)

- Merits of BVA

1) High error Detection Rate

→ Boundary conditions are prone to errors, and BVA helps catch these errors efficiently

2) Reduced Number of Test Cases

→ Compared to testing the entire range, BVA allows focusing on a few key cases, making testing faster and more efficient

3) Effective for Critical Conditions
→ It's particularly useful in scenarios with strict input ranges, as it helps ensure correct handling of edge cases.
4) Applicable in Multiple Domains
→ BVA is versatile and can be used in various applications, such as financial systems, healthcare and more

c) Demerits of BVA
1) Limited Coverage of Complex Scenarios
→ BVA only tests boundaries and may miss errors within the middle of input ranges or those requiring more complex combination of inputs
2) Not effective for Non-Range Inputs
→ BVA is ideal for inputs with a defined range but is less effective for Boolean, categorical or non-numeric data
3) Inability to Detect Internal Logic Errors
→ Since BVA focuses on external behaviour, it may miss errors in the code's internal structure or logic
4) Risk of Overlooking Equivalence Partitions
→ If testers only rely on BVA without using Equivalance Partitioning, they might miss certain errors outside boundary cases

Q-9 What are the levels at which testing is done?

Ans The testing is done at following levels which are as follows
1) Unit testing
2) Integration testing
3) Validation testing
4) System testing

1) Unit testing :- In this type of testing techniques are applied to detect the errors from each software component individually
2) Integration testing :- It focuses on issues associated with verification and program construction as components begin interacting with one another
3) Validation testing - It provides assurance that the software validation criteria meets all functional, behavioural and performance requirements
4) System testing - In System testing all system elements forming the system is tested as a whole

Q-10 Define basic path testing

Ans Basic Path testing is a type of White Box testing. In this method the procedural design using basis set of execution path is tested. This basis set ensures that every execution path will be tested at least once.
→ Flow Graph Notation and Graph Matrices are types of Basic Path testing.

Q-11 What do you mean by integration testing ? Explain their outcomes

Ans → A graph of dependent components are tested together to ensure their quality of their integration unit.
→ The objective is to take unit tested components and build a program structure that has been dictated by software design

Integration testing approach

- Non-Incremental Integration
  - Big Bang
- Incremental Integration
  - Top Down testing
  - Bottom Up testing
  - Regression testing
  - Smoke testing

0 Outcomes

1) Early Detection of Interface Issues
→ Identifies compatibility problems at module interfaces, such as mismatches in data formats or functional method invocation errors.

2) Validation of Data Integrity
→ Ensures data passed between modules is accurate and consistent, which is crucial for applications that rely on correct data flow

3) Improved Reliability of System
→ By testing module interactions, integration testing increases confidence that the system will function correctly in more complex scenarios

4) Reduced Risks of Future Errors
→ Detecting and fixing integration-related errors helps reduce the risk of more significant issues arising later, improving overall software stability

5) Documentation of Defects
→ Integration testing generates records of found defects, helping teams understand typical interaction issues, document them and prevent future recurrences

Q-12 What is White Box testing ? Discuss different techniques of it? Why it is required

Ans White box testing also known as clear box testing, glass box testing is a software testing method where the tester has knowledge of the internal structure, logic and code of the software.

Why White Box testing is Required

1) Ensuring Code Quality : It helps ensures that each line of code, is functional, efficient and free of defects
2) Finding Hidden Errors : White Box testing uncovers issues like within the internal logic, such as unreachable code, boundary errors and incorrect algorithms
3) Optimizing Code : Testing the structure of code enables the identification of redundant or inefficient sections, which can be optimized for better performance
4) Security Verification : White box testing allows for rigorous checking of code, which is crucial for identifying and mitigating security vulnerabilities

0 Techniques of White Box ~~Techniques~~ Testing

1) Statement Coverage

→ In this technique, the aim is to traverse all statements at least once. Hence, each line of code is tested.

→ Since all lines of code are covered, it helps in pointing out faulty code

2) Branch Coverage

→ In this technique, test cases are designed so that each branch from all decision points is traversed at least once.

3. Condition Coverage
→ Tests each individual condition within a decision to confirm it evaluates correctly for all possible outcomes.
→ To detect errors in complex conditions by testing each part of a condition

4. Multiple Condition Coverage
→ In this technique, all the possible combinations of the possible outcomes of conditions are tested at least once.

5. Basis Path Testing
→ In this technique, control flow graphs are made from code or flowchart and then Cyclomatic Complexity is calculated which defines the number of independent paths so that the minimal number of test cases can be designed for each independent path

6. Loop Testing
→ Loops are widely used and these are fundamental to many algorithms hence, their testing is very important. Errors often occur at the beginings and ends of loops.

2-13 How Cyclomatic Complexity is used in White Box Testing

Ans 1) Determine Minimum Test Cases: It tells the minimum number of independent paths needed to fully test all logic paths, ensuring comprehensive coverage
2) Access Code Quality: High complexity indicates harder-to-maintain code, helping identify areas that may benifit from simplification

3 Improve test coverage: Ensures all branches and decisions are tested, reducing the risk of missed errors

4 Guide Refactoring: Highlighting complex sections that may need restructuring for better readability and reliability

5 Identify Risky Areas: High complexity often correlates with higher defect risks, helping prioritize testing efforts

Q-14 List all of the review guidelines and brief which do you think is most important and why?

Ans (1) Follow Coding Standards
(2) Check Code Logic and Functionality
(3) Test for Edge Cases and Error Handling
(4) Review for Performance and Efficiency
(5) Evaluate Security aspects
(6) Ensure Proper Documentation
(7) Verify Reusability and Modularity
(8) Focus on Code Simplicity and Clarity

Q-15 What is Smoke Testing? Explain activities it encompasses

Ans The smoke testing is a kind of integration testing technique used for time critical projects wherein the projects needs to be assessed on frequent basis

Activities in Smoke Testing

1) Software components already translated into code are integrated into a "build". The "build" can be data files, libraries, reusable modules or program components

2) A series of tests are designed to expose errors from build so that the "build" performs its functioning correctly
3) The "build" is integrated with the other builds and the entire product is smoke tested daily

Q-16 What should be testing approaches for Mobile applications ?

Ans Here are some essential testing approaches for mobile applications
1) Functional Testing
→ Testing features, workflows, and ensuring that the app meets all functional requirements
2) Usability Testing
→ This testing often involves real users to provide feedback on the design and usability
3) Performance Testing
→ Check the app's responsiveness, speed, and stability under various conditions
4) Compatibility Testing
→ Test on different devices, resolutions and platforms (Android, iOS)
5) Security testing
→ Protects user data and prevent vulnerabilities
6) Interrupt testing
→ Ensure the app can handle interruptions like calls, messages, notifications or power loss

Ch-7

Quality Assurance and Management

Q-1 Write short note on Software CMM Levels

Ans The Capability Maturity Model (CMM) is used in assessing how well an organizations processes allow to complete and manage new software projects
- Various process maturity levels are

Level 1: Initial - Few processes are defined and individual efforts are taken

Level 2: Repeatable - To track cost schedule and functionality basic project management processes are established. Depending on earlier successes of projects with similar applications necessary process discipline can be repeated.

Level 3: Defined: The process is standarized, documented and followed. All the projects use documented and approved version of software process which is useful in developing and supporting software

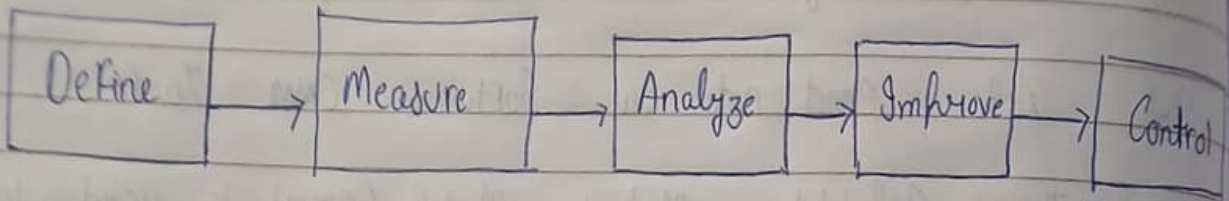
Level 4: Managed: Both the software process and product are quantitatively understood and controlled using detailed measures

Level 5: Optimizing: Establish mechanisms to plan and implement change. Innovative ideas and technologies can be tested

Q-2 Write short note on SIX SIGMA

Ans Six Sigma is widely used statistical software quality assurance strategy.

→ It is a business driven approach to process improvement, reduced costs and increased profit.



There are three core steps in six sigma method
1) Define - The customer requirements, project goals and deliverables are defined by communicating the customers
2) Measure - The existing process and its output is measured in order to determine current quality performance
3) Analyze - In this phase defect metrics are analyzed in order to determine the few causes
4) Improve - By eliminating the root causes of defects the process can be improved
5) Control - The process can be controlled in such a way that the causes of defects can not be reintroduced

Q-3 What is Software Quality? List down different Software Quality Metrics

Ans Software Quality refers to the degree to which a software product meets specified requirements, satisfies user expectations and is free from defects.
→ It is measured against specific standards and requirements to ensure it fulfills its intended purpose in a robust and secure manner

0 Software Quality Metrics
1) Product Quality metrics
2) Process Quality metrics
3) Project management metrics
4) User Satisfaction Metrics
5) Security metrics

Q.4 How Software organization go from different maturity level of SEI CMM ? Explain it

Ans See Q-1

Q.5 Write a Short Note on Six Sigma Standard

Ans See Q-2

Q.6 Explain the SQA activities

Ans The SQA plan is a document created for summarizing all the SQA activities conducted for the software project The standard for this plan is published by IEEE standard

SQA plan

1) Purpose and scope of the plan
2) Description of work product
3) Applicable standards
4) SQA activities
5) Tools and methods/ standards used
6) SCM procedures for managing change
7) Methods for maintaining

Q.7 What is the important of SQA

Ans SQA is importance due to following reasons

(1) Enhances Software Reliability and Performance
(2) Improves Customer Satisfaction and Confidence
(3) Reduces Development and Maintanence Costs
(4) Facilitates Compliance with Standards and Regulations
(5) Ensures Security and Data Integrity
(6) Promotes efficiency and Productivity

Page No.
Date

Q.8 How Organization can get Iso 9000 certification? Explain the process

Page No.
Date

Q-9 Explain Formal Technical Review

Ans A Formal technical Review (FTR) is a software quality control activity performed by software engineers (and others)

- The objectives of FTR are

1) To uncover errors in function, logic or implementation; for any representation of the software
2) To verify that the software under review meets its requirements
3) To ensure that the software has been represented according to predefined standards
4) To achieve software that is developed in a uniform manner
5) To make projects more manageable

Ch - 8
Software Maintanence and Configuration Management

Q-1 List down various software Design principles to College Mananagement

Ans
(1) Modularity and Seperation of Concerns
(2) Single Responsibility Principle (SRP)
(3) Encapsulation
(4) Reusability
(5) Open / Closed Principle
(6) Database Normalization
(7) Loose Coupling
(8) Scalability

Q-2 Write a short note on: (1) Function-Oriented Design
(2) User Interface Design

Ans (1) Function - Oriented Design
→ Function - oriented design is a software design approach focused on identifying and defining functions or procedures that the system must perform.
→ It centers around breaking down a large system into smaller, manageable functions that each handle a specific task.
- Characteristics: This approach emphasizes actions and behaviors over data, with a primary focus on functionality and process flow. Each function typically operates on global data shared across the system, which can lead to tightly coupled components.
- Usage: Function Oriented Design is commonly used in

Procedural Programming and legacy systems, and it suits applications where processes are predictable and data requirements are stable

2 User Interface (UI) Design

→ User Interface Design is the process of designing interfaces that allow users to interact with a system effectively and efficiently. It involves creating visual and interactive elements that provide a seamless and intuitive experience for users.

→ Process: UI design typically begins with understanding user requirements, followed by wireframing and prototyping. Designers work closely with developers to ensure the final interface is both visually appealing and functional

→ Goals: The main goal of UI design is to make interactions smooth, reduce cognitive load and enhance usability.

Q-3 Write short note on Software Configuration management

Ans The primary objectives of Software Configuration management

1) Configuration Identification: Identify the items that define the software configuration
2) Change Control: Manage changes to one or more items
3) Version Control: Facilitate to create different versions of the application
4) Configuration Authentication: To ensure that the quality of the software is maintained as the configuration evolves over time

The SCM process must be developed in such a way that the software team must answer the following set of

Questions

1) How does the software team identify the software configuration items?
2) How does the software team control the changes in the software before and after delivering it to the customer?
3) How does the software team manage the versions of the programs in the software package?
4) How does team get ensured that the changes are made properly?
5) Who is responsible for approving the changes in the software?

Q-4 Write short note on Reverse-Engineering OR Explain Reverse Engineering

Ans Reverse engineering is the process of design recovery. In reverse engineering the data, architectural and procedural information is extracted from a source code. There are three important issues in reverse engineering:-

1) Abstraction Level:- This level helps in obtaining the design information from the source code. It is expected that abstraction level should be high in reverse engineering High abstraction level helps the software engineerin to understand the program.
2) Completeness level:- The completeness means detailing of abstract level. The completeness decreases as abstraction level increases.
3) Directionality level: Directionality means extracting the information from source code and give it to software engineer. The directionality level can be one way or two way. The one way directionality means extracting

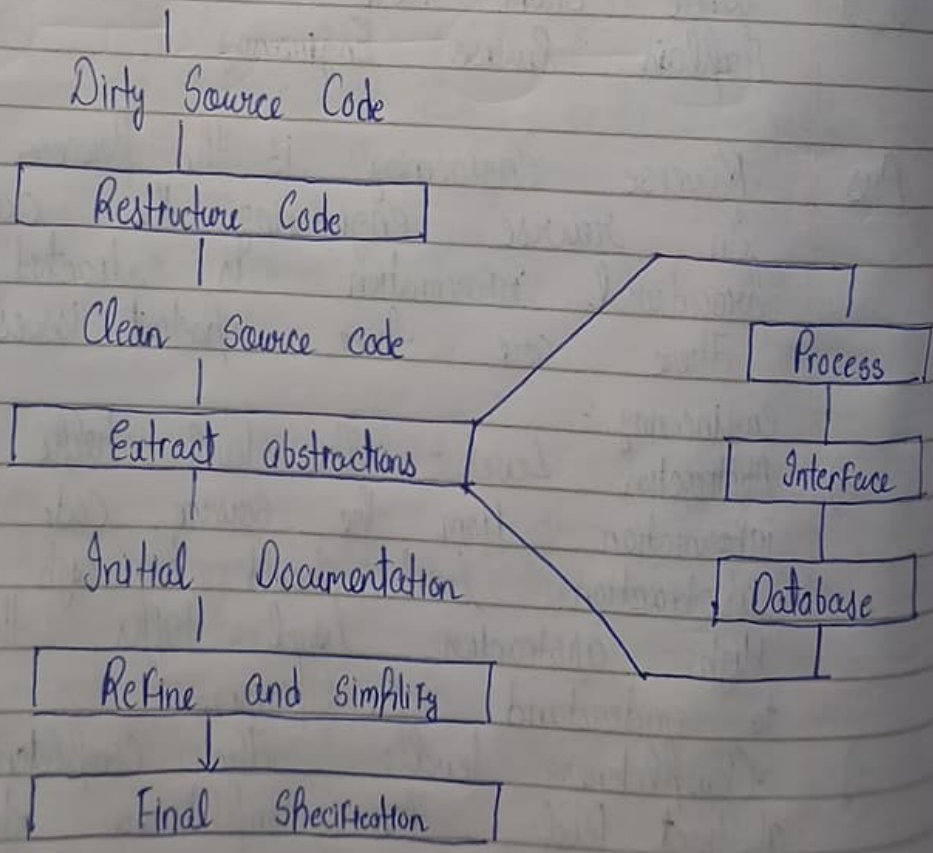
all the information from source code and give it to software engineer.

→ The two way directionality means the information taken from source code is fed to a re-engineering tool that attempts to restore restructure or regenerate old program.

→ Initially the dirty source code or unstructured source code is taken and processed and the code is restructured

→ After restructuring process the source code becomes clean

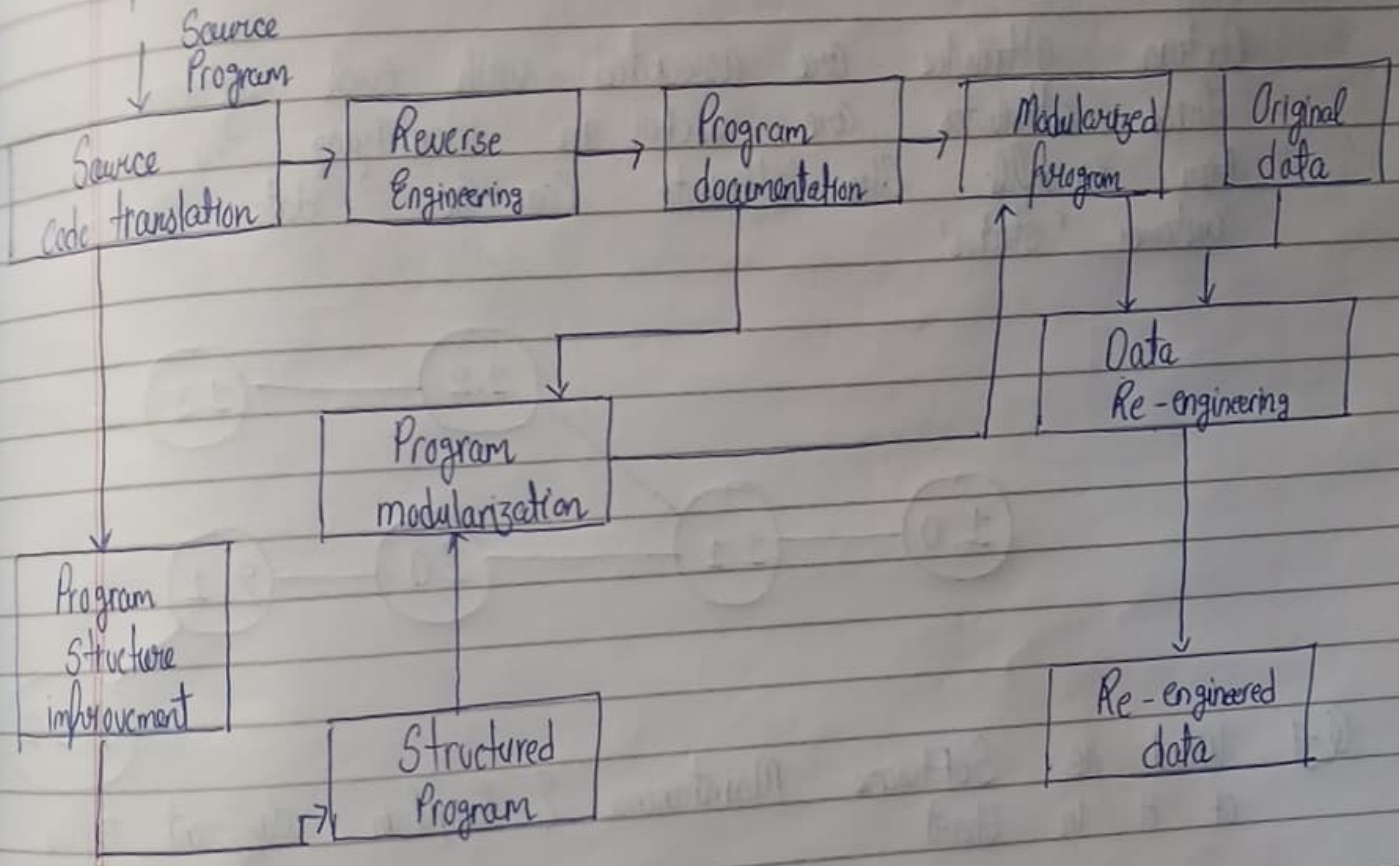
→ The core to reverse engineering is an activity called extract abstractions.



The Output of reverse engineering process is a clear, unambiguous final

Q-5 Write short note on Re-engineering OR
Explain a Software Re-engineering Process Model using the diagram

Ans Software re-engineering means re-structuring or re-writing part of all of the software engineering system



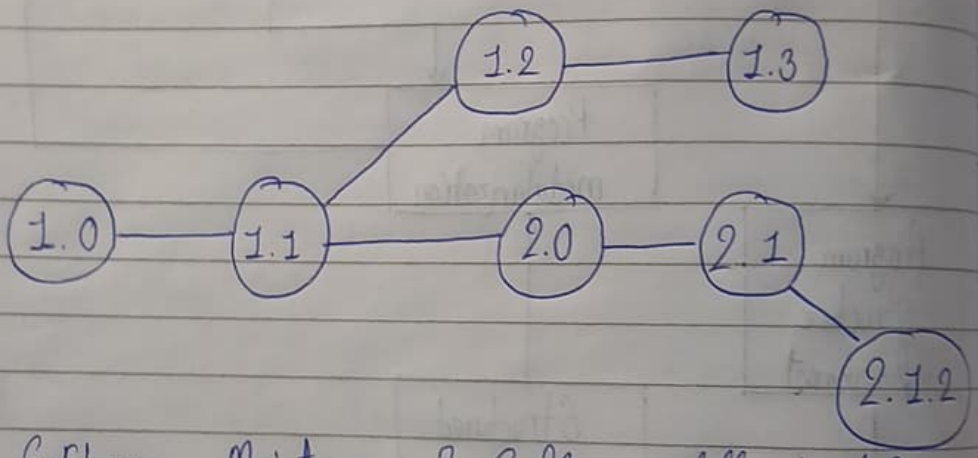
Re-engineering process activities

- Source code translation: In this phase the code is converted to a new language
- Reverse engineering: Under this activity the program is analysed and understood thoroughly
- Program structure improvement: Restructure automatically for understability
- Program modularization: The program structure is reorganized
- Data re-engineering: Finally clean up and restructure system data

Q-6 Write short note on Version Control

Ans Version is an instance of a system which is functionally distinct in some way from other system instances.

Certain attributes are associated with each software version. These attributes are useful in identifying the version. For example: The attribute can be 'date', 'creator', 'customer', 'status'.



Q-7 What is Software Maintanence? Explain different types of it in short

Ans Software maintanence is the process of modifying and updating software applications after they have been deployed to correct issues, improve performance, or adapt to new requirements. Software maintanence can be categorized into four main types

1) Corrective Maintanence: Fixes bugs and errors identified after the software releases. This type addresses fault that affect functionality or cause unexpected behaviours
2) Adaptive Maintanence: Updates the software to work with new or changing environments, like updated operating systems hardware or cause unexpected behaviours

5) Perfective Maintanence : Enhances software performance and usability by optimizing processes, adding new features, or improving user experience based on feedback

4) Preventive Maintanence : Involves making changes to prevent future issues, improving software reliability and longevity by addressing potential faults or obsolescence proactively

Q-5 How version and change are controlled within and across organizations ? Explain it

Ans

Q) Explain the version control and change control

Ans Change Control refers to the systematic process of managing any modifications or adjustments made to a project, system, product or service

Need for change
↓
Change Request
↓
Evaluation of Request
↓
Generation of Change Report
↓
Decision of CCA for Change

- Denial of change request
  ↓
  Intimate user about denial
- Change request accepted and ECO generated
  ↓
  Assign individuals to configuration object
  ↓
  Check out configuration objects
  ↓
  Make changes, review them
  ↓
  Check the object that have been changed
  ↓
  Perform SQA and testing activities
  ↓
  Promotes changes in next release
  ↓
  Rebuit attributable version
  ↓
  Distribute the new version

Step 1: First of all there arises a need for the change

Step 2: The change request is then submitted by user

Step 3: Developers evaluate this request to access technical merits, potential side effects and overall impact on system functions and cost of the project

Step 4: A change report is then generated and presented to the Change Control Authority (CCA)

Step 5: The change control authority is a person or a group of people who makes a final decision on status or priority of the change

Step 6: An Engineering Change Order (ECO) is generated when the change gets approved. In ECO change is described, the restrictions and criteria for review and audit are mentioned

Step 7: The object that needs to be changed is checked out of the project database

Step 8: The changes are then made on the corresponding object and appropriate SQA activities are then applied

Step 9: The changed object is then checked in to the database and appropriate version control is made to create new version

Q-16 How organization can get ISO 9000 certification? Explain the process

Ans

Ch-9 DevOps

1) What are the difference between Agile and DevOps?

Ans

	Agile	DevOps
1)	The idea in agile is to develop software in small iterations and thus be able to adapt to the changing customer needs	DevOps is to deliver technology to business units in a timely fashion and ensure the technology runs without interruption or disruption
2)	It adopts rapid development approach	Thus is not a rapid development approach
3)	The focus of agile development is merely on software development and release	The focus of DevOps is not only on software development, its release but on its safest deployment in working environment
4)	Communication in agile development is informal and in the form of daily meetings	In DevOps communication, documents are involved and it is formal. It does not occur on daily basis
5)	The team is small in nature	Large team size and multiple teams are required in DevOps
6)	Agile is about software development	DevOps is about software development and management
7)	Agile is less flexible	DevOps is more flexible
8)	Agile has limited scope	DevOps has broader scope

Q-2 Discuss 7 C's of DevOps Lifecycle For Business Agility OR Explain 7 C's of DevOps Lifestyle OR Explain

Ans the 7C's DevOps are as follows

1) Continuous business planning : During this phase the planning for identifying skills, resources required and outcome is made
2) Continuous development : In this phase, development sketch plan is prepared and programming techniques are identified. Before continous integration, development teams would never write a bunch of code for three to four months. Then those teams would merge their code in order to release it.
3) Continous Integration : Continous integration is the practice of quickly integrating newly developed code with the main body of code that is to released. Continous integration saves a lot of time when the time is ready to release the code
4) Continous Deployment : It is the practice of deploying all the way into production without any human intervention. Teams that utilize continous delivery don't deploy untested code ; instead, newly created code runs through automated testing before it gets pushed out of production
5) Continous Testing : Continous testing (CT) is the process of executing automated test cases as part of the software delivery pipeline. Its goal is to obtain immediate and continous feedback on the business risks associated with a software release candidate
6) Continous delivery and monitoring : With continous delivery, every code change is built, tested and then pushed to a non-production testing or staging environment. There can be

multiple, parallel test stages before a production deployment.

→ Continuous Feedback: This is a process which allows for an immediate response from your customers for your product and its features and helps you modify accordingly

Q3 What are different challenges for adopting DevOps?
OR Describe the different challenges with DevOps implementation OR Explain DevOps life cycle

Ans (1) Cultural change: Any organization must provide the collaborative culture for effective implementation of DevOps. The leaders should bring transparency in the work process. There must be positive atmosphere in an organization.
(2) Bringing Silos and Sectors together: There is a tendency of maintaining silos and sectors within the team while developers are constantly writing pieces of code in order to build a system, testers perform a thorough analysis to ensure product stability before the final delivery to the customer. Due to this practice, there lies a big gap between teams. Both Dev and Ops work in silos leading to a lack of transparency and poor teamwork.
(3) Giving up legacy systems: Organizations must give up the old or outdated systems and must adopt modern and efficient systems.
→ Handling new systems along with the old existing systems in the organization can be challenging many times
(4) Tool selection mechanism: Confusion
→ There are number of tools available in the market which tempt the DevOps developers to choose different tools.

5) Different Metrics : During product development process each team measure their performance using different metrics.
6) Resistance to change : The teams are normally unwilling to accept the changes. They are resistance to change their preferred working style. They do not open up the silos to other teams. This makes it difficult to other teams to work and may create an unhealthy environment.
7) Process Challenges : DevOps strategy does not define specific rules to implement the process or to use the tools. The only restriction is to follow the project goal.

Q-4 What is DevOps ? Explain its importance and benefits of DevOps OR What is DevOps ? Provide Importance and benefits for the same

Ans DevOps : DevOps is a practice in which development and operation engineers participate together in entire life cycle activities of system development from design, implementation to product support.

- DevOps Benifits and Importance
- Importance

1) DevOps enhance the organization's performance, improves the productivity and efficiency of development and operations team
2) Bringing the two teams together centralizes the responsibility on the entire team and not specific individuals working
3) DevOps is more than just a tool or a process change. It inherently requires an organizational culture shift

☆ Benifits

- Technical Benefits
  1) Continuous Software delivery is possible
  2) There is less complexity to manage the project
  3) The problems in the project gets resolved faster
- Cultural Benefits
  1) The productivity of teams get selected
  2) There is higher employee engagement
  3) There arise greater professional development opportunities
- Business Benifits
  1) The faster delivery of the product is possible
  2) The operating environment becomes stable
  3) More time is available for innovation rather than fixing and maintaining

Q.5 What is Devops ? List down its toolchain for development process

Ans Tool chain for development process

1) Planning and Collaboration : Jira, Slack, MS Teams
2) Source Code Management (SCM) : Git, Github
3) Build : Apache Maven, Jenkins
4) Continous Integration (CI) : Jenkins, CircleCI, Github CI/CD
5) Continous Delivery (CD) : Jenkins, Cucumber
6) Continous Testing : JUnit, TestNG, Cucumber

Q.6 How Devops practice be adopted for software development activities

Ans (1) Perspective Consideration : The difference between the developers and practitioners perspective must be considered

2) Flexibility : The organization must have flexibility to switch between the delivery of the product using DevOps value and without DevOps values
3) Integrate Changes : It must be possible to accomodate the required changes in the system
4) Agility : The process as well as tool selection needs to be agile according to project needs
5) Transparency : There must be transparency about the goals, delivery plan and activities between developers and operational engineers

Page No.
Date

Ch-10 Advanced topics in Software Engineering

1) What is web engineering ? Explain any three web engineering methods.

Ans

Page No.
Date

Q-2 What is Component Based Software Engineering ? What are its advantages ?

Ans The CBSE is an approach of defining, implementing, integrating loosely coupled components into system

Benifits/ advantages

1) By CBSE, it becomes easy to construct understandable and maintainable software
2) Components are independent entities and they do not interfere in other component's operation
3) Component implementation is hidden
4) Communication among the components is through well defined interfaces
5) Component platform can be shared and ultimately it reduces the cost of development

Q-3 Explain Computer-Aided Software Engineering in detail

Ans A CASE (Computer Aided Software Engineering) tool is generic term used to denote any form of automated support for software engineering

→ A CASE tool means any tool used to automate some activity associated with software development

o The primary reasons for using a CASE tools are:-

→ To increase productivity

→ To help produce better quality software at lower cost.

o Components of CASE

1) CASE repository
   → Central Components of any CASE tool
   → Also known as the information repository

→ Allows easy sharing of information between tools and SDLC activities
→ Used to store graphical diagrams and prototype forms and reports during analysis and design workflows
→ Provides wealth of information to project manager and allows control over project

2) Diagramming tools
→ Allow you to represent a system and its components visually
→ Allows higher level processes to be easily decomposed
→ Can examine processes or data models at high or low level

3) Screen and report generators
→ Used to Create, modify and test prototypes of Computer displays and reports.
→ Identify which data items to display or collect for each screen or report.
→ Some tools have templates

4) Analysis tools
→ Generate reports that help identify possible inconsistencies, redundancies and omissions (deficiency)
→ Generally focus on
  o diagram completeness and consistency
  o data structures and usage

5) CASE documentation generator tools
→ Create standard reports based on contents of repository
→ High-quality documentation leads to 80% reduction in system maintenance effort in comparision to average quality documentation