Software Engineering

(Full Complete Notes)

Important With Que & Ans

UNIT - 1

Introduction To Software Engineering

1. Define Software Engineering?

Software Engineering is the process of designing, developing, testing, and maintaining software. It is a systematic and disciplined approach to software development that aims to create high-quality, reliable, and maintainable software.

2. List Advantages & Disadvantages of Software Engineering?

Advantages -

- Systematic approach
- Quality assurance
- Efficient project management
- Clear Documentation
- Requirement Analysis
- Improved communication
- Scalibility and maintainability
- Cost effective development
- Adaptability to change
- Customer satisfication

- Ethical considerations
- Continuous improvement

Disadvantages -

- Complexity and overhead
- Time-consuming
- Rigidity in processes
- Learning curve
- Resource intensive

3. Define Requirement Validation ?

Requirement validation ensures that the specified requirements accurately reflect the needs of stakeholders and are consistent, complete, and feasible.

4. Define requirement Elicitation?

Requirement elicitation is the process of gathering information from stakeholders to understand their needs, desires, and expectations for the software system.

5. Explain Professional Software Development?

Professional software development in software engineering refers to the disciplined and systematic process of designing, building, testing, and maintaining software systems by following industry best practices, standards, and ethical guidelines.

Key Aspects of professional software development -

1. Requirements Analysis and Specification :

- Gathering requirements -
- Requirements specification -

2. System design :

- Architectural design
- Detailed design

3. Implementation :

- Coding
- Code review

4. Testing :

- Unit testing
- Integration testing
- System testing

5. Deployment :

- Release management
- Continuous Integration/Continous Deployment(CI/CD)

6. Maintainance and support :

- Bug Fixing
- Updates and Enhancements

7. Project Management and collaboration :

- Agile Methodologies
- Communication

8. Quality Assurance :

- Code quality
- Security

9. Ethical Consideration :

- Updates and Enhancements

10. Professional Development :

- Continuous learning
- Certifications

6. Functional and non-functional requirement?

Functional Requirement -

Functional requirements define the specific functionalities or features that a software system must possess. These requirements describe what the system should do in terms of input, processes, and output.

Characteristics -

- Behavioral descriptions
- Tangible features
- User-centric
- Specificity

Non - Functional Requirement -

Non-functional requirements specify the qualities or characteristics that describe how well the system performs its functions. Unlike functional requirements, non-functional requirements are not concerned with specific behaviors but rather with overall system attributes.

Characteristics -

- Performance
- Reliability
- Usability
- Scalability

- Security
- Availability
- Maintainibility

7. What is requirement analysis ?

Requirement analysis involves examining and understanding the gathered requirements to identify inconsistencies, ambiguities, and conflicts.

8. What is requirement management?

Requirement management involves the systematic organization, tracking, and control of requirements throughout the software development life cycle.

9. What is Software Requirement Specification(SRS) ?

The Software Requirements Specification (SRS) is a key document in software engineering that outlines the requirements for a software system.

10. Explain Requirements Engineering?

Requirements engineering is a critical phase in the software engineering process that focuses on defining, documenting, and managing the needs and expectations of stakeholders for a software system.

Key aspects of requirements engineering -

1. Requirement Elicitation
2. Requirement Analysis
3. Requirement Specification
4. Requirement Validation
5. Requirement Management

Benefits of requirements engineering -

- Risk Reduction

- Improved communication
- Quality Assurance
- Cost and Time Savings
- Customer Satisfaction

UNIT - 2

Software Processes

1. Define Software Process Model or SDLC?

A Software Process Model, also known simply as a Software Development Life Cycle (SDLC) model, is a systematic approach to the development, implementation, and maintenance of software.

2. Explain Process activities ?

In software engineering, **process activities refer to the** various tasks and actions that are performed during the software development life cycle.

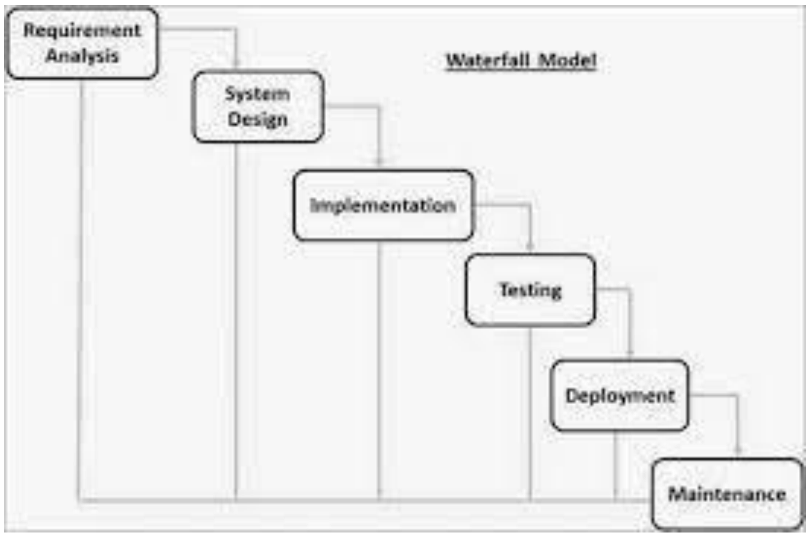
Key Stages of Process activities -

1. **Communication -** Establish effective communication channels among stakeholders.
2. **Planning -** Plan the project, defining scope, schedule, resources, and activities.

3. **Modelling -** Create models to represent the system, its architecture, and its components.
4. **Coding -** Translate design models into executable code.
5. **Testing -** Verify and validate the software to ensure it meets requirements.
6. **Deployment -** Release the software for end-users.
7. **Maintanence -** Address issues, implement updates, and ensure the ongoing viability of the software.
8. **Configuration management -** Manage and control changes to software artifacts.
9. **Quality Assurance -** Ensure that the software quality.
10. **Risk Management -** Identify, assess, and mitigate risks throughout the project.
11. **Documentation -** Create and maintain documentation to describe the software system.
12. **Project Monitoring and control -** Monitor and control project progress, ensuring adherence to plans.
13. **Communication Management -** Facilitate effective communication among team members and stakeholders.

3. Explain waterfall Model?

- The Waterfall model is a linear and sequential approach.
- Each phase must be completed before moving on to the next.
- It is easy to understand and use, making it suitable for small projects with well-defined requirements.



1. Requirements analysis **-** The system's services, constraints, and goals are

established by consultation with system users.

2. System design - The systems design process allocates the requirements to either

hardware or software systems by establishing an overall system architecture.

3. Implementation - During this stage, the software design is realized as a set of programs or program units.

4. Testing - The individual program units or programs are integratedand tested as a complete system.

5. Maintenance - This is the longest life cycle phase. The system is installedand put into practical use.

Key Phases in Waterfall Model :

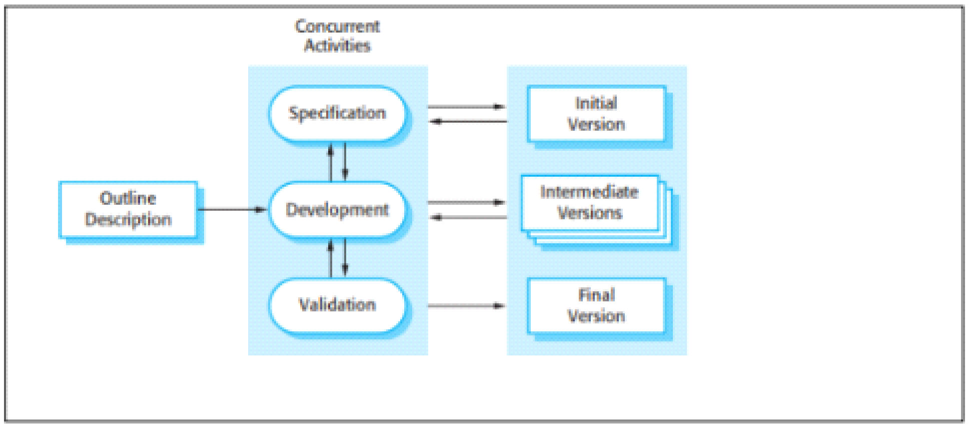
- Requirements

- Design
- Implementation (Coding Phase)
- Testing
- Deployment
- Maintenance

4. Define Incremental Process Model?

Incremental Development is a software development model where the system is designed, implemented, and tested incrementally until the product is finished.

- Similar to the iterative model, but in incremental development, the final system is built as separate prototypes.
- Each prototype represents a portion of the complete system.
- New functionality is added in increments until the full system is integrated



Incremental Development Phases -

1. Planning

2. Analysis
3. Design
4. Implementation (Coding)
5. Testing
6. Integration
7. Evaluation
8. Repeat

5. Explain Reuse-Oriented Software Engineering?

Reuse-Oriented Software Engineering (ROSE) is an approach in software engineering that emphasizes the reuse of software artifacts across different projects.

Key Concepts -

1. Reusable Components
2. Domain Analysis
3. Component Libraries
4. Benefits of reuse
5. Productivity improvement
6. Cost reduction
7. Quality Enhancement

Process of Reuse-oriented mode -

1. Identify Reusable Components - Conduct domain analysis to identify common functionalities, patterns, and components that can be reused.

2. Build Component Libraries - Develop libraries of reusable components, documenting their functionality, interfaces, and usage guidelines.

3. Reuse Planning - During project planning, assess the potential for reuse and identify suitable components from the libraries.

4. Adaptation and Integration - Customize and integrate reusable components into the new project, addressing any variability and ensuring compatibility.

5. Verification and Testing - Verify the functionality and quality of the reused components within the context of the new project.

6. Documentation and Maintenance - Update documentation to reflect any changes or adaptations made to the reusable components.

6. Explain Boehm's Spiral Model?

The Boehm's Spiral Model, proposed by Barry Boehm, is a risk-driven software development process model that combines the idea of iterative development with elements of the Waterfall Model.

Key Characteristics of Boehm's Spiral Model -

1. Risk Assessment -

The model begins with risk assessment, where potential risks are identified and analyzed. Risks could be related to requirements, technology, budget, schedule, etc.

2. Planning -

In this phase, the project is planned based on the results of the risk assessment.

3. Engineering -

This phase involves the actual development of the software.

4. Evaluation -

The software product is evaluated at the end of each iteration.

Advantages of Boehm's spiral model -

1. Risk Management
2. Flexibility
3. Client Feedback

Challenges of Boehm's Spiral Model -

1. Complexity
2. Costly
3. Not Suitable for Small Projects

7. Explain Agile Software Development? (*IMPORTANT*)

Agile software development is an iterative and flexible approach to software development that emphasizes collaboration, customer feedback, and the ability to adapt to changing requirements.

Agile Software Development Methathodologies or Practices -

1. Scrum:

Scrum is a agile software development methodology that helps teams to break down the complex project into smaller functional pieces.

It focuses on delivering incremental value through short development cycles called **sprints**.

Scrum Sprint cycle -

1. Sprint planning - Define what can be delivered in the sprint and how it will be achieved.

2. Daily Scrum - Facilitate quick and effective communication within the development team.

3. Sprint execution - Develop and deliver the product incrementally.

4. Sprint review - Inspect the increment and adapt the product backlog if needed.

5. Sprint retrospective - Reflect on the sprint and identify opportunities for improvement.

Benefits of scrum sprint cycle -

- Increased visibility
- Adaptability
- Frequent deliveries
- Continuos improvement
- Enhanced collaboration

2. Extreme Programming (XP):

Extreme Programming (XP) is an Agile software development methodology that emphasizes customer satisfaction, flexibility, and continuous feedback.

XP Release cycle -

1. Exploration

2. Planning game

3. Iteration(Development and testing)

4. Acceptance testing

5. Planning game (Repeat)
6. Continuous integration
7. Small release
8. Whole team collaboration
9. Refactoring
10. Customer feedback and continous improvement

UNIT - 3

Architectural Design

1. What is architectural design?

Architectural design is a process for identifying the sub-systems making up a system and the framework for sub-system control and communication.

Advantages -

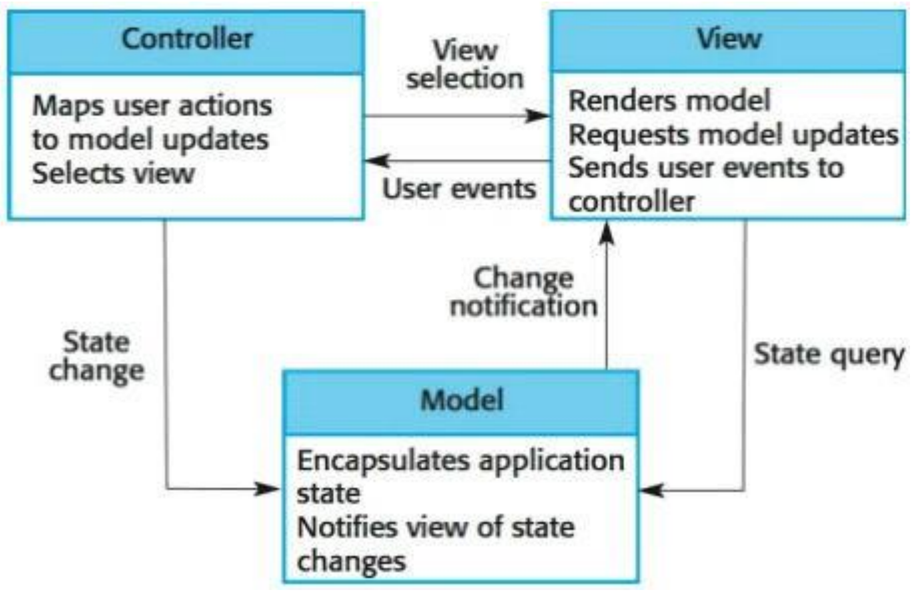
- Stakeholder communication
- System analysis
- Large scale reuse

2. List architectural Design patterns?

- Model-view controller
- Layered architecture
- Client-server architecture

3. Explain Model View Controller?

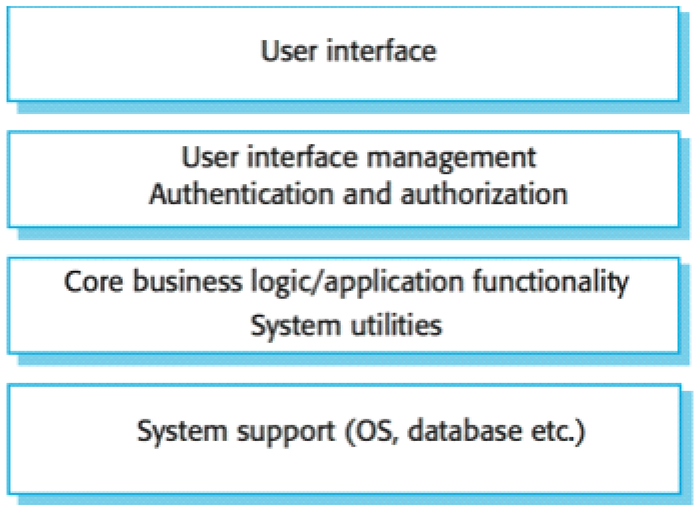
Model view controller serves as a basis of interaction management in many web-based system.



- **Model -** The model is the central component of the pattern that directly manages the data, logic and rules of the application. It is the application's dynamic data structure, independent of the user interface.
- **View -** A view can be any output representation of information, such as a chart or a diagram.
- **Controller -** The controller accepts input and converts it to commands for the model or view

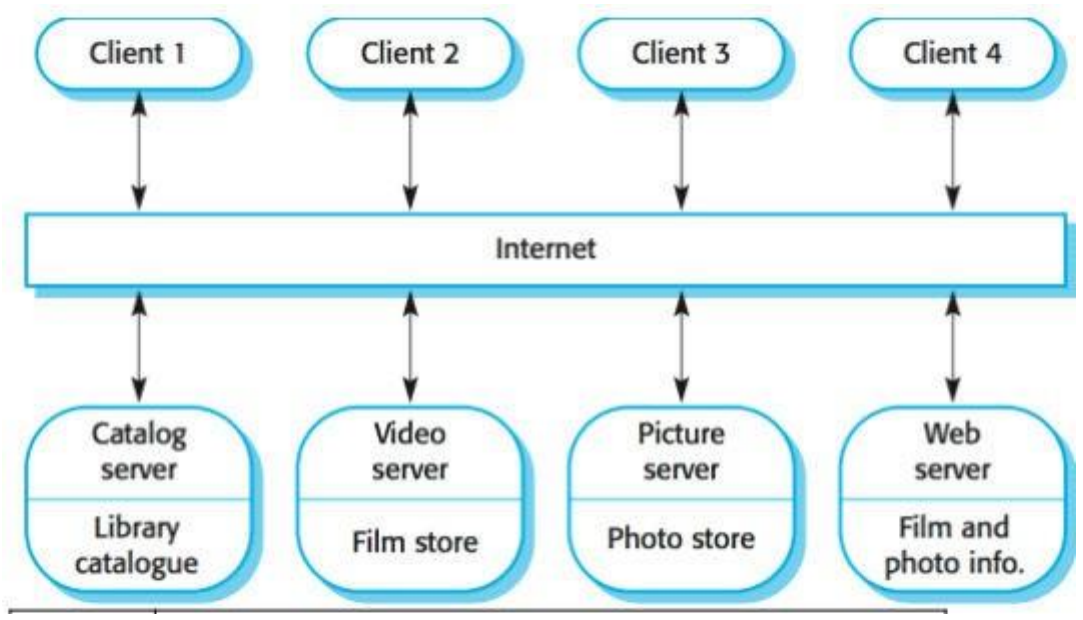
4. Explain layered architecture?

- Used to model the interfacing of sub-systems.
- Organizes the system into a set of layers each of which provide a set of services.
- Supports the incremental development of sub-systems in different layers. When a layer interface changes, only the adjacent layer is affected.


User interface
User interface management
Authentication and authorization
Core business logic/application functionality
System utilities
System support (OS, database etc.)

5. Explain client-server architecture?

In a client-server architecture, the functionality of the system is organized into services, with each service delivered from a separate server. Clients are users of these services and access servers to make use of them.



It is used when data in a shared database has to be accessed from a range of locations. Because servers can be replicated, may also be used when the load on a system is variable.

Advantage -

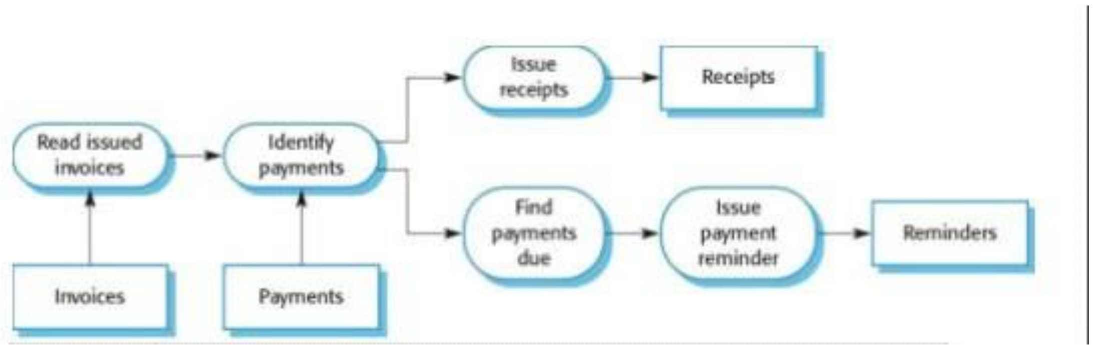
The principal advantage of this model is that servers can be distributed across a network.

Disadvantage -

Each service is a single point of failure so susceptible to denial of service attacks or server failure.

6. Explain pipe and filter architecture?

The processing of the data in a system is organized so that each processing component (filter) is discrete and carries out one type of data transformation. The data flows (as in a pipe) from one component to another for processing.



Commonly used in data processing applications (both batch- and transaction based) where inputs are processed in separate stages to generate related outputs.

Advantages -

- Easy to understand and supports transformation reuse.
- Workflow style matches the structure of many business processes.

Disadvantage -

- The format for data transfer has to be agreed upon between communicating transformations.

7. What is application architecture?

A generic application architecture is an architecture for a type of software system that may be configured and adapted to create a system that meets specific requirements.

8. What is system modelling?

System modeling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system.

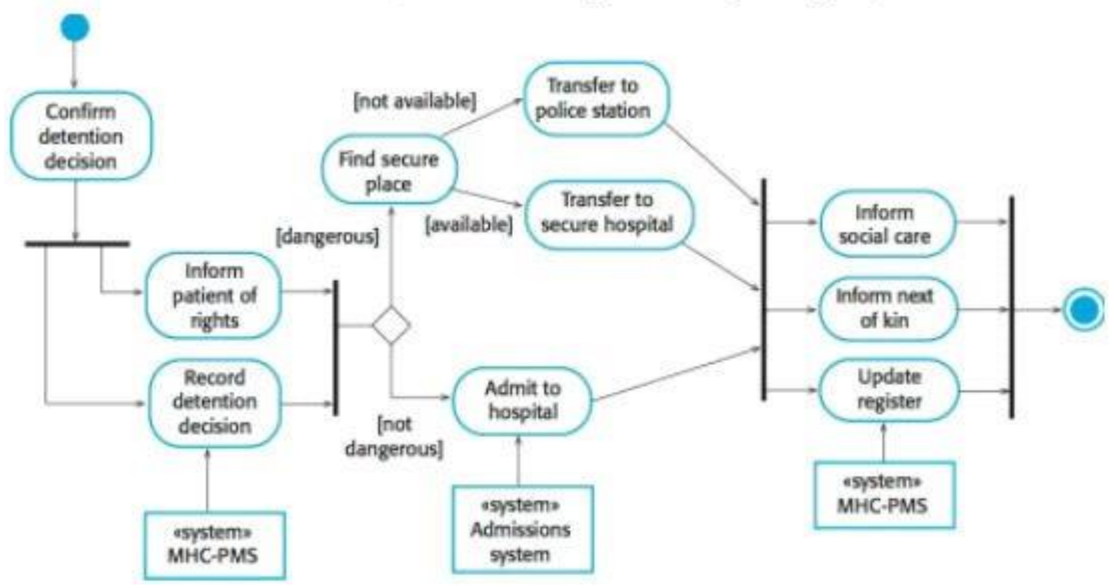
9. Define UML and list its types?

Unified modelling language is a standardized visual modelling language used in software engineering.

Types -

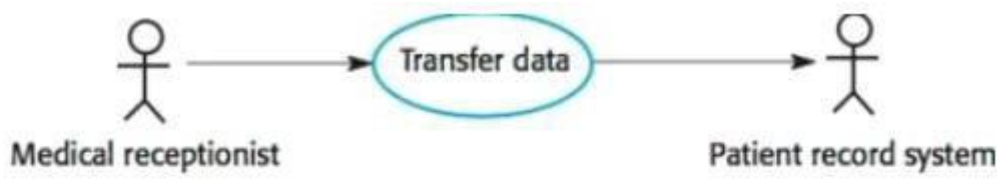
1. Activity Diagram -

Activity diagrams, which show the activities involved in a process or in data processing.



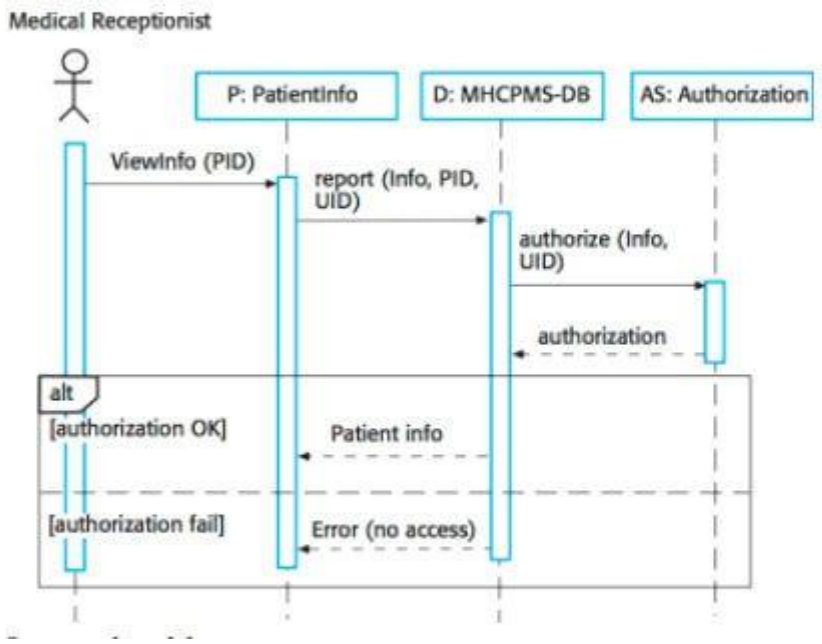
2. Use case -

Use case diagrams, which show the interactions between a system and its environment.



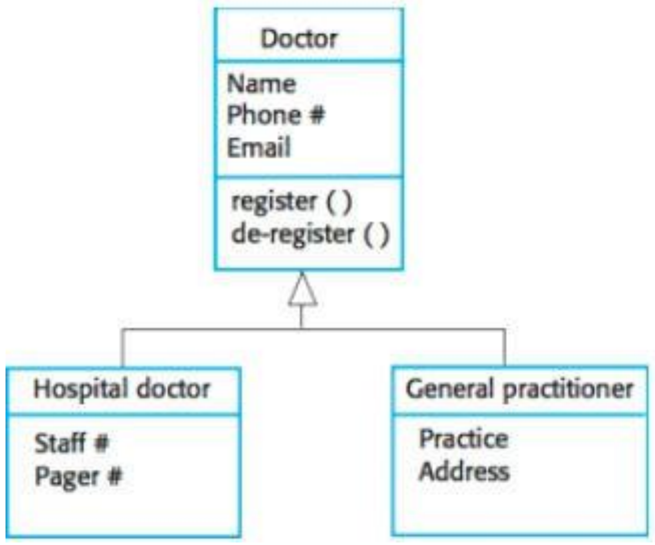
3. Sequence Diagram -

Sequence diagrams, which show interactions between actors and the system and between system components.



4. Class Diagram -

Class diagrams, which show the object classes in the system and the associations between these classes.



5. State Diagram -

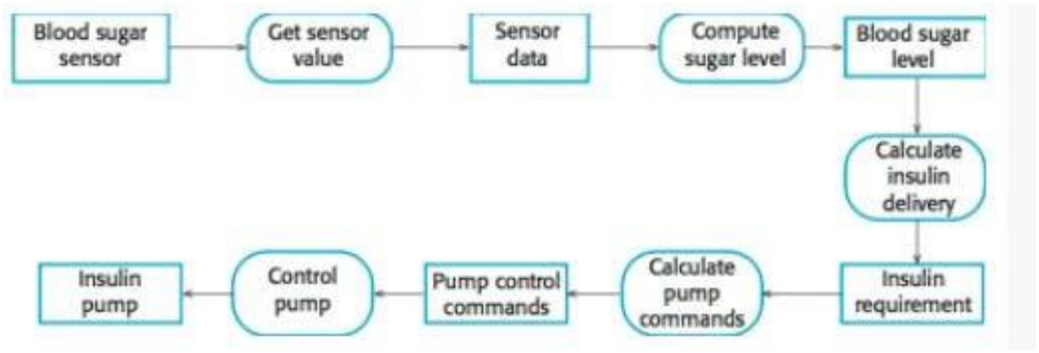
State diagrams, which show how the system reacts to internal and external events.

10. Explain behavioral model?

Behavioral models are models of the dynamic behavior of a system as it is executing. Theyshow what happens or what is supposed to happen when a system responds to a stimulus from its environment.

Two types of stimuli -

- Some **data** arrives that has to be processed by the system.
- Some **event** happens that triggers system processing. Events may have associated data, although this is not always the case.



11. Explain DFD?

Data Flow Diagrams (DFD) are graphical representations of a system that illustrate the flow of data within the system.

Advantages of DFDS -

- Easy to understand
- Improves system analysis
- Supports system design
- Enables testing and verifcation

Disadvantages of DFDS -

- Can be time comsuming

- Limited focus
- Can be difficult to keep up-to-date
- Requires technical expertise

Basic components of DFD -

1. External Entity :

External entity is the entity that takes information and gives information to the system. It is represented with rectangle.

2. Data Flow :

The data passes from one place to another is shown by data flow. Data flow is represented with arrow and some information written over it.

3. Process :

It is used to process all the information.It is represented with circle and name of the process and level of DFD written inside it.

4. Data Store :

It is used to store the information and retrieve the stored information.It is represented with double parallel lines.

12. Explain Levels of DFD?

1. Level 0 DFD :

This is the highest-level DFD, which provides an overview of the entire system. It shows the major processes, data flows, and data stores in the system, without providing any details about the internal workings of these processes.

2. Level 1 DFD :

This level provides a more detailed view of the system by breaking down the major processes identified in the level 0 DFD into sub-processes.

3. Level 2 DFD :

This level provides an even more detailed view of the system by breaking down the sub-processes identified in the level 1 DFD into further sub-processes.

4. Level 3 DFD :

This is the most detailed level of DFDs, which provides a detailed view of the processes, data flows, and data stores in the system.

UNIT - 4

User Interface Design

1. How does consistency contribute to a good UI design?

- Maintain consistency in design elements, layout, and interactions.
- Consistency enhances predictability, making the interface more intuitive.

2. Describe user centric design(UCD) and its significance?

UCD places users at the center of the design process, involving them in requirements gathering, prototyping, and testing. It ensures that the UI is tailored to user needs and preferences.

3. Explain Golden Rule?

1. Clarity :

- Ensure that the interface communicates information clearly and concisely.

2. Consistency :

- Maintain consistency in design elements, layout, and interactions throughout the application.

3. Visibility :

- Important features and functions should be easily visible and accessible.

4. Feedback :

- Provide immediate and relevant feedback for user actions.

5. Efficiency :

- Design workflows and interactions to be efficient and minimize user effort.

6. Aesthetics :

- Create visually appealing designs that align with the brand and user expectations.

7. Hierarchy and Prioritization :

- Organize information and features based on importance.

8. Simplicity :

- Keep the interface simple and avoid unnecessary complexity.

9. Flexibility and Efficiency of Use :

- Design for both novice and expert users, providing features that cater to different skill levels.

10. Error Prevention and Handling :

- Implement design elements that prevent errors where possible.

11. Accessibility :

- Design with accessibility in mind, ensuring that the interface is usable by people with disabilities.

12. User Control :

- Empower users with control over the interface and their actions.

13. Familiarity :

- Leverage established design patterns and conventions to create a familiar and intuitive user experience.

14. User Feedback Integration :

- Actively seek and incorporate user feedback into the design process.

4. Explain User Interface Design Process?

1. Requirements Gathering :

- Understand the needs and expectations of users, as well as business requirements.

2. User Research :

- Explore the target audience's preferences, behaviors, and pain points.

3. Task Analysis :

- Identify key workflows and prioritize features based on user needs.

4. Prototyping :

- Prototypes allow for early testing and feedback.

5. Visual Design :

- Apply visual elements such as colors, typography, and imagery to create a visually appealing interface.

6. Interaction Design:

- Focus on creating a seamless and intuitive user experience.

7. Usability Testing :

- Conduct usability testing with real users to gather feedback on the design.

8. Implementation :

- Work closely with developers to ensure the design is accurately implemented.

9. Evaluation and Iteration :

- Iterate on the design based on evolving requirements and user needs.

10. Documentation :

- Documentation ensures consistency in implementation and future updates.

5. Explain UI Design Process?

1. Research :

- Understand the target audience, their goals, and the context of use. Analyze competitors and industry trends.

2. Wireframing :

- Focus on functionality without detailing visual elements.

3. Prototyping :

- Prototypes help validate design concepts and gather feedback.

4. Visual Design :

- Apply visual elements such as colors, typography, and imagery.

5. Testing :

- Conduct usability testing with real users to identify issues and gather feedback.

6. Implementation :

- Work closely with developers to ensure the design is implemented accurately.

7. Iteration :

- Continuously iterate on the design based on user feedback and evolving requirements.

6. List UI Design Tools?

- Sketch
- Figma
- Adobe XD
- InVision
- Azure RP

7. Explain user interface analysix and design models?

1. Waterfall Model:

- This model follows a sequential approach, where each phase must be completed before moving on to the next.

- Phases include requirements gathering, system design, implementation, testing, deployment, and maintenance.

2. Iterative Model:

- In an iterative model, the UI design process is repeated and refined through cycles.
- Each iteration incorporates feedback and improvements, making it flexible to changing requirements.

3. Agile Model:

- Agile methodologies emphasize collaboration, flexibility, and responsiveness to change.
- UI design tasks are integrated into short development cycles known as sprints, allowing continuous improvement based on user feedback.

4. User-Centered Design (UCD):

- UCD places the user at the centfer of the design process, involving users throughout the development lifecycle.
- It typically includes phases such as user research, prototyping, testing, and iteration.

5. Component-Based Development (CBD):

- CBD involves the creation of reusable UI components, allowing for consistency and efficiency in design.
- Designers focus on building and refining components that can be assembled to create the overall interface.

8. Explain user interface desing principles and guidelines for web and mobile?

- Consistency
- Responsive design
- Web and mobile
- Visual hierarchy
- Feedback
- Touch-Friendly design
- Performance
- Typography

9. Define software quality?

Software quality refers to the degree to which a software product or system meets specified requirements and user expectations.

10. List and Explain McCalls Quality factors?

1. Product Revision Factors:

- **Correctness:** The degree to which the software performs its intended functions without errors.
- **Reliability:** The ability of the software to maintain performance under stated conditions for a specified period.

2. Product Transition Factors:

- **Portability:** The ease with which the software can be transferred from one environment to another.
- **Reusability:** The extent to which software components can be reused in other applications.

3. Product Operation Factors:

- **Maintainability:** The ease with which the software can be modified to correct defects, meet new requirements, or improve performance.
- **Flexibility:** The ease with which the software can accommodate changes in its specifications.

11. List ISO 9126 Quality factors?

Functionality, Reliability, Usability, Efficiency, Maintainibility, Portability.

11. Define Host Target Development?

Host-target development is a process in software engineering where the development environment (host) is distinct from the execution environment (target).

UNIT - 5

Software Testing

1. Define Software Testing?

Software Testing is a method to assess the functionality of the software program. The

process checks whether the actual software matches the expected requirements and

ensures the software is bug-free.

2. Define Manual Testing & Automation Testing?

Manual Testing - Manual testing includes testing software manually, i.e., without using any

automation tool or script.

Automation Testing - Automation testing, which is also known as Test Automation, is when

the tester writes scripts and uses another software to test the product.

3. Define verification & Validation in software testing?

Verification - It refers to the set of tasks that ensure that the software correctly

implements a specific function.

Validation - It refers to a different set of tasks that ensure that the software that has

been built is traceable to customer requirements.

4. Define Unit Testing?

Unit testing involves the testing of each unit or an individual component of the software application.

Unit testing techniques -

- Data flow Testing
- Control Flow Testing
- Branch Coverage Testing
- Statement Coverage Testing

- Decision Coverage Testing

Advantages -

- Unit testing uses module approach due to that any part can be tested without waiting for completion of another parts testing.
- Unit testing allows the developer to refactor code after a number of days and ensure the module still working without any defect.

Disadvantages -

- It cannot identify integration or broad level error as it works on units of the code.
- It is best suitable for conjunction with other testing activities.

5. Explain White box testing or glass box testing?

White box testing is testing, structural testing, clear box testing, open box testing and transparent box testing. It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs.

Advantages -

- White box testing optimizes code so hidden errors can be identified.
- Test cases of white box testing can be easily automated.
- It can be started in the SDLC phase even without GUI.

Disadvantages -

- White box testing is too much time consuming when it comes to large-scale programming applications.

- White box testing is much expensive and complex.
- It can lead to production error because it is not detailed by the developers.
  o

6. Explain Black box testing?

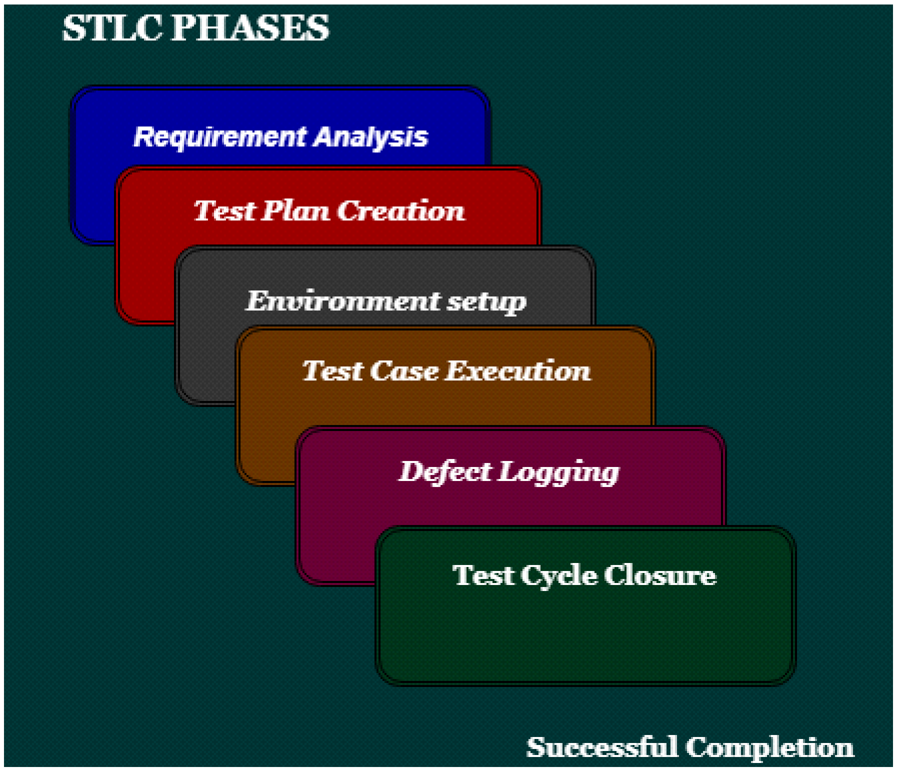
Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding.

The primary source of black box testing is a specification of requirements that is stated by the customer.

Generic steps of black box testing -

- The black box test is based on the specification of requirements, so it is examined in the beginning.
- In the second step, the tester creates a positive test scenario and an adverse testscenario by selecting valid and invalid input values to check that the software is processing them correctly or incorrectly.
- In the third step, the tester develops various test cases such as decision table, all pairs test, equivalent division, error estimation, cause-effect graph, etc.
- The fourth phase includes the execution of all test cases.
- In the fifth step, the tester compares the expected output against the actual output.
- In the sixth and final step, if there is any flaw in the software, then it is cured and tested again.

7. Explain Software Testing Life Cycle With Diagram?



1. Requirement Analysis -

The first step of the manual testing procedure is requirement analysis. In this phase, tester analyses requirement document of SDLC to examine requirements stated by the client.

2. Test Plan Creation -

Test plan creation is the crucial phase of STLC where all the testing strategies are defined. Tester determines the estimated effort and cost of the entire project.

3. Environment setup -

Setup of the test environment is an independent activity and can be started along with Test Case Development.

4. Test case Execution -

Test case Execution takes place after the successful completion of test planning. In this phase, the testing team starts case development and execution activity.

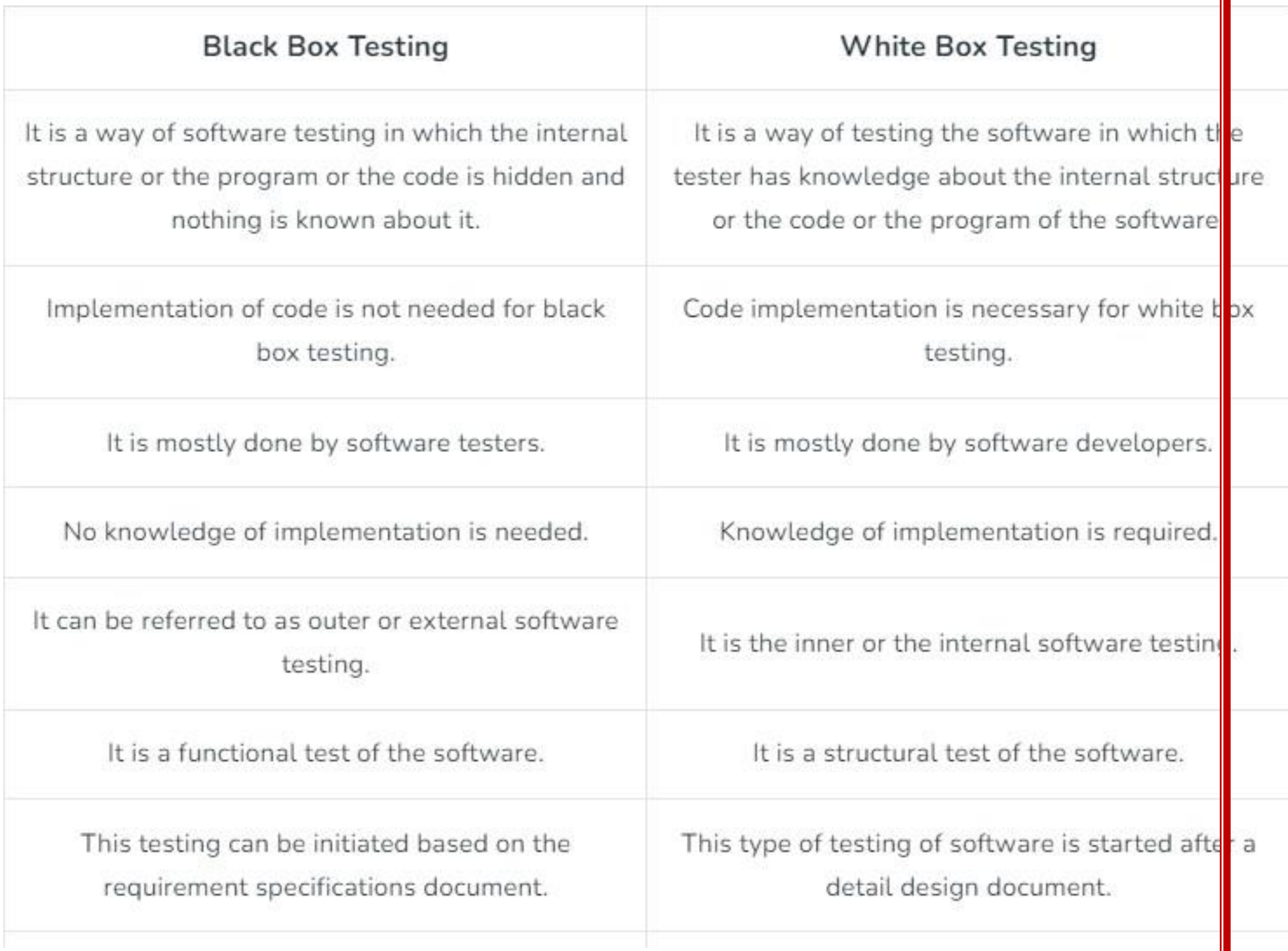
5. Defect Logging -

Testers and developers evaluate the completion criteria of the software based on test coverage, quality, time consumption, cost, and critical business objectives.

6. Test Cycle Closure -

The test cycle closure report includes all the documentation related to software design, development, testing results, and defect reports.

8. List differences between white box and black box testing?



Black Box Testing	White Box Testing
It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.	It is a way of testing the software in which the tester has knowledge about the internal structure or the code or the program of the software
Implementation of code is not needed for black box testing.	Code implementation is necessary for white box testing.
It is mostly done by software testers.	It is mostly done by software developers.
No knowledge of implementation is needed.	Knowledge of implementation is required.
It can be referred to as outer or external software testing.	It is the inner or the internal software testing.
It is a functional test of the software.	It is a structural test of the software.
This testing can be initiated based on the requirement specifications document.	This type of testing of software is started after a detail design document.

9. Define Functional Testing?

Functional testing is a type of software testing that validates the software systems against the functional requirements.

10. Define Non-functional Testing?

Non-functional testing is a type of software testing that checks the application for non-functional requirements like performance, scalability, portability, stress, etc.

11. Define Maintainance Testing?

Maintainance testing is the process of changing, modifying, and updating the software to keep up with the customer's needs.

12. Types of Functional testing?

- Unit testing
- Integration testing
    - *Incremental testing*
    - *Non-incremental testing*
- System testing

13. Types of Non-Functional testing?

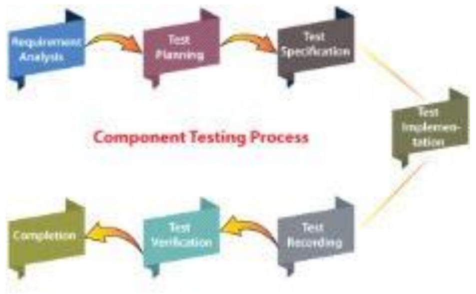
- Performance testing -
    - Load testing, stress testing, scalability testing, stability testing.
- Usability Testing

- Compatibility Testing

14. Explain component testing?

It is used to test all the components separately as well as the usability testing.

Component testing process -



Step1: Requirement Analysis -

The first step of component testing is requirement analysis, where the user requirement associated with each component is detected.

Step2: Test Planning -

Once the requirement analysis phase is done, we will mov to the next step of component testing process, which is test planning. In this phase, the test is designed to evaluate the requirement given by the users/clients.

Step3: Test Specification -

Here, we will identify those test cases that needs to be executed and missed

Step4: Test Implementation -

The forth step in the component testing process is Test implementation. When the test cases are identified as per the user requirements or the specification, then only we can implement the test cases.

Step5: Test Recording -

In this step of the component testing process, we have the records of those defects/bugs discovered during the implementation of component testing.

Step6: Test Verification -

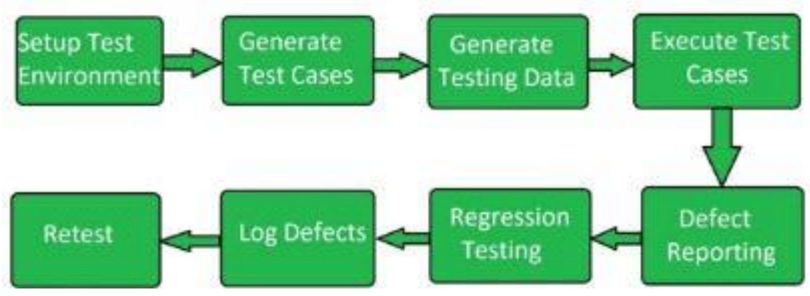
Once the bugs or defects have been recorded successfully, we will proceed to the test verification phase. It is the process of verifying whether the product fulfils the specification or not.

Step7: Completion -

After completed all the above steps successfully, we will come to the last step of the component testing process. In this particular step, the results will be evaluated in order to deliver a good quality product.

15. Define and explain system testing?

System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements.



- **Test Environment Setup:** Create testing environment for the better quality testing.
- **Generate Test Case:** Generate test case for the testing process.
- **Generate Test Data:** Generate the data that is to be tested.
- **Execute Test Case:** After the generation of the test case and the test data, test cases are executed.
- **Defect Reporting:** Defects in the system are detected.

- **Regression Testing:** It is carried out to test the side effects of the testing process.
- **Log Defects:** Defects are fixed in this step.
- **Retest:** If the test is not successful then again test is performed.