G. PULLAIAH COLLEGE OF ENGINEERING AND TECHNOLOGY: KURNOOL
RAVINDRA COLLEGE OF ENGINEERING FOR WOMEN: KURNOOL
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Subject : SOFTWARE ENGINEERING **Code : 15A05401**

B.Tech. II Year-II Semester (R15) **Two Marks Questions**

UNIT - 1

1. **Define software?**

Software can be defined as follows:

- Instructions or computer programs that when executed provide desired features, function & performance.
- Data structures that enable the programs to adequately manipulate operation.
- Documents (descriptive information) in both hard copy and virtual forms that describes the operation and use of the programs.

2. **What is Software Engineering? What are its applications?**

Software Engineering: (1) The application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software; that is, the application of engineering to software.

Applications are:

- Business Process Re-Engineering.
- Communication Networks.
- Computer Graphics.
- Cooperative Work Support.
- e-Commerce.
- Education.
- Embedded Systems Programming.
- m-Commerce.

3. **List out the characteristics of software?**

The software quality model identifies 6 main quality characteristics, namely:

- Functionality.
- Reliability.
- Usability.
- Efficiency.
- Maintainability.
- Portability.

4. Draw software and hardware failure rate curves as a function of time?

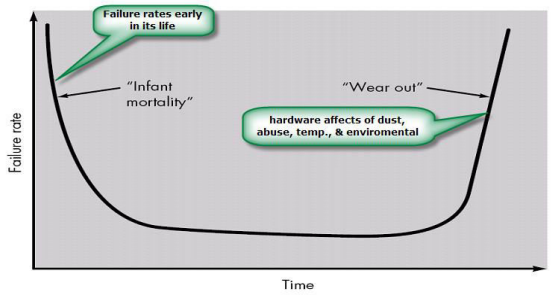


Fig: Hardware failure rate as a function of time BATH TUB CURVE

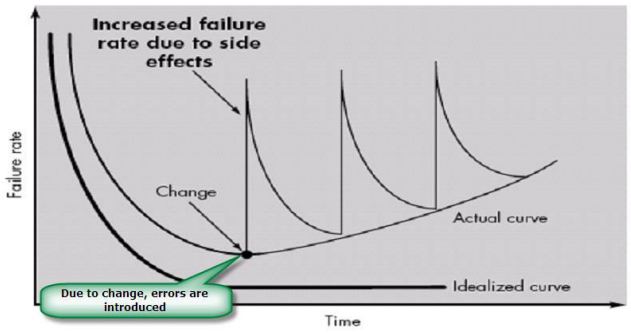


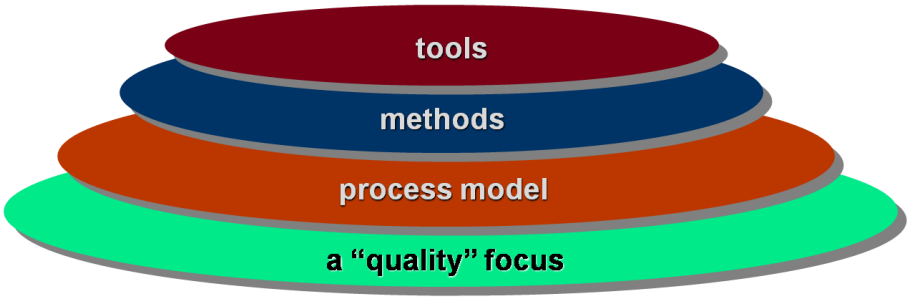
Fig: Software Failure Curve

5. List out the types of software?

The various types of the software are:

System Software
Application Software
Engineering /Scientific software
Embedded Software
Product line software
Web Applications
Artificial Intelligence software

6. What are the layers of software engineering?



Software Engineering

7. What is software process? Give its importance.

A software process is a set of activities, methods, practices, and transformations that people use to develop and maintain software and the associated products (e.g., project plans, design documents, code, test cases, and user manuals)

Process being a fundamental tool for carrying out community consensus and facilitating very large number of people to work together on a collaborative project

8. What are the elements of generic process framework?

A generic process framework for software engineering encompasses five activities:

- **Communication**
- **Planning**
- **Modelling** (Analyze, Design)
- **Construction** (Code, Test)
- **Deployment**

9. Define a software myth? List out the types of software myths?

Definition: Beliefs about software and the process used to build it. Myths have number of attributes that have made them insidious (i.e. dangerous).

- □ Misleading Attitudes - caused serious problem for managers and technical people.

There are three types of myths:

1. Management myths
2. Customer myths
3. Practitioner myths

10. Define a software pattern?

A *process pattern*

i. Describes a process-related problem that is encountered during software engineering work,
ii. Identifies the environment in which the problem has been encountered, and
iii. Suggests one or more proven solutions to the problem.

11. List out the types of process flows?

They are 4 types of process flows

i. Linear Process flow
ii. Iterative Process flow
iii. Evolutionary Process flow
iv. Parallel process flow

12. What is legacy software? What are the characteristics of legacy software?

Definition: Legacy software is old software that is still useful

Characteristics:

a. It cannot be simply discarded because it contains experienced and validated knowledge
b. It was developed to answer precise needs that are not covered by existing software
c. The system might be running on old hardware.
d. The system might be currently used by other people

13. List out the prescriptive process models for the development of software?

The list of prescriptive models is:

1. Waterfall model
2. V-model
3. Incremental model
4. Evolutionary models: Prototyping model, Spiral model and Concurrent model

14. What are the advantages of waterfall model?

Advantages of waterfall model:

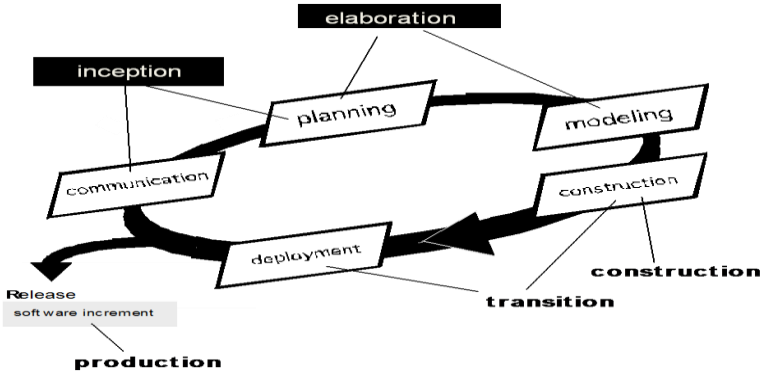
i. Simple and Easy to understand and implement.
ii. Reinforces good habits: define-before- design, design-before-code
iii. Phases are processed and completed one at a time.
iv. Works well for smaller projects where requirements are very well understood.

15. What are the advantages of Prototype model?

Advantages of Prototype model:

i. Users are actively involved in the development
ii. Since in this methodology a working model of the system is provided, the users get a better understanding of the system being developed.
iii. Errors can be detected much earlier.
iv. Quicker user feedback is available leading to better solutions.
v. Missing functionality can be identified easily

16. What are the different phases of unified process?



17. Define the roles in scrum?

There are three roles in the Scrum method of software development:

Product Owner
ScrumMaster, and
The team.

18. What is an Agile Process?

Agile SDLC model is a combination of iterative and incremental **process** models with focus on **process** adaptability and customer satisfaction by rapid delivery of working **software** product. **Agile** Methods break the product into small incremental builds. These builds are provided in iterations.

Agile model believes that every project needs to be handled differently and the existing methods need to be tailored to best suit the project requirements. In agile the tasks are divided to time boxes (small time frames) to deliver specific features for a release.Iterative approach is taken and working software build is delivered after each iteration. Each build is incremental in terms of features; the final build holds all the features required by the customer.

19. Differentiate between personal software process and team software process model

Personal software process model (PSP): PSP measures the work product that is produced and results the quality of the product. The psp model defines 5 framework activities

Planning ---> High level design ---> High level design review ---> Development --->Postmortem

Team software process model (TSP) : The goal of TSP is to build a "self-directed" project team that organizes itself to produce high quality software.TSP has following framework activities:

Project launch--->High level design--->Implementation--->Integration & Test--->Postmortem

UNIT – 2

1. State Characteristics of SRS document

Software requirement specification (SRS) is a document that completely describes what the proposed software should do without describing how software will do it. The basic goal of the requirement phase is to produce the SRS, Which describes the complete behavior of the proposed software. SRS is also helping the clients to understand their own needs.

Characteristics of an SRS:

1. Correct
2. Complete
3. Unambiguous
4. Verifiable
5. Consistent
6. Ranked for importance and/or stability
7. Modifiable
8. Traceable

2. Discuss about class based modeling

Class-based elements

i. Implied by scenarios – each usage scenario implies a set of objects that are manipulated as an actor interacts with the system. These objects are categorized into classes.
ii. Class diagrams depicting classes, their attributes and relationships between classes are developed.
iii. Collaboration diagrams are developed to depict the collaboration of classes

3. Discuss the role of developer in negotiating requirements of the system to be developed.

The objective of this phase is to agree on a deliverable system that is realistic for developers and customers.

i. Conflicting requirements of customers, end-users and stake holders are reconciled.
ii. Iterative process for – requirements prioritization, risk analysis, cost estimation etc.

Roles are:

i. Identify the key stakeholders
   - These are the people who will be involved in the negotiation
ii. Determine each of the stakeholders "win conditions"
   - Win conditions are not always obvious
iii. Negotiate
   - Work toward a set of requirements that lead to "win-win"

4. Why scenario based modeling is getting popular in the field of requirements modeling

Scenario-based elements

i. Using scenario-based approach the system is described from the user's point of view.
ii. Use-case—descriptions of the interaction between an "actor" and the system are developed.
iii. Activity diagrams and swim-lane diagrams can be developed to complement use-case diagrams.

5. Discuss analysis patterns of requirement engineering?

i. Certain problems reoccur across all projects within a specific application domain. These analysis patterns suggest solutions within the application domain that can be reused when modeling many applications.
ii. Analysis patterns are integrated into the analysis model by reference to the pattern name.
iii. They are also stored in a repository so that requirements engineers can uses search facilities to find and apply them.

6. Identify goals of elicitation phase?

The goal of elicitation phase is:

i. to identify the problem
ii. propose elements of the solution
iii. negotiate different approaches, and
iv. specify a preliminary set of solution requirements

7. Explain the process of validating requirements?

In this phase a review mechanism adapted that looks for

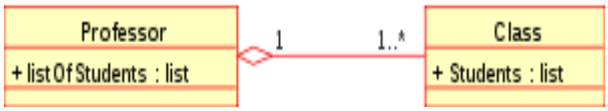
i. errors in content or interpretation
ii. areas where clarification may be required
iii. missing information
iv. inconsistencies (a major problem when large products or systems are engineered)
v. conflicting or unrealistic (unachievable) requirements.

8. **Illustrate 'dependency' relationship of class diagrams?**

A *dependency* is a semantic connection between dependent and independent model elements. It exists between two elements if changes to the definition of one element (the server or target) may cause changes to the other (the client or source). This association is uni-directional.

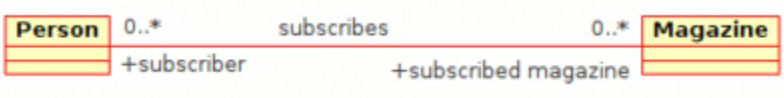
9. **Explain 'aggregation' relationship of class diagrams with the help of an example?**

Aggregation is a variant of the "has a" association relationship; aggregation is more specific than association. It is an association that represents a part-whole or part-of relationship. As shown in the image, a Professor 'has a' class to teach. As a type of association, an aggregation can be named and have the same adornments that an association can



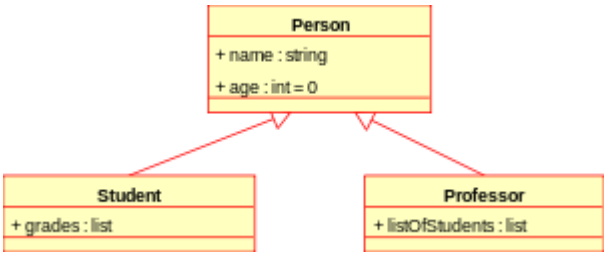
10. **Explain 'Association' relationship of class diagrams with an example?**

An *association* represents a family of links. A binary association (with two ends) is normally represented as a line. An association can link any number of classes. An association with three links is called a ternary association. An association can be named, and the ends of an association can be adorned with role names, ownership indicators, multiplicity, visibility, and other properties.



11. **Illustrate 'Generalization' relationship of class diagrams?**

The generalization relationship is also known as the *inheritance* or *"is a"* relationship. It indicates that one of the two related classes (the *subclass*) is considered to be a specialized form of the other (the *super type*) and the superclass is considered a Generalization of the subclass. In practice, this means that any instance of the subtype is also an instance of the superclass.



12. **What is Requirement Engineering and Software Requirement Engineering?**

Requirements Engineering: Requirements Engineering builds a bridge to design and construction.

The broad spectrum of tasks and techniques that lead to an understanding of requirements is called Requirements Engineering.

Software Requirement Engineering: Requirements analysis, also called **requirements engineering**, is the process of determining user expectations for a new or modified product.

These features, called requirements, must be quantifiable, relevant and detailed. In **software engineering**, such **requirements** are often called functional specifications.

13. Discuss the role of use cases in UML

Use Case Diagrams: Use case diagrams are usually referred to as behavior diagrams used to describe a set of actions (use cases) that some system or systems (subject) should or can perform in collaboration with one or more external users of the system (actors).

Actors are the different people (or devices) play as the system operates. Every actor has one or more goals when using the system.

UNIT – 3

1. List out software quality attributes

a. ***Functionality***: is assessed by evaluating the features set, the generality of the functions that are delivered, and the security of the overall system.
b. ***Usability***: is assessed by considering human factors, overall aesthetics, consistency, and documentation.
c. ***Reliability***: is evaluated by measuring the frequency and severity of failure, the accuracy of output results, the mean-time-to-failure, the ability to recover from failure, and the predictability of the program.
d. ***Performance***: is measured by processing speed, response time, resource consumption, throughput, and efficiency.
e. ***Supportability***: combines the ability to extend the program extensibility, adaptability, serviceability ➔ maintainability. In addition, testability, compatibility, configurability, etc.

2. List out the components of a Software Design Model?

The components of a software design model are :

i. Data/Class Design Model
ii. Architectural Design Model
iii. Interface Design Model
iv. Component Level Design Model

3. What are the properties that should be exhibited by a good software design according to Mitch Kapor?

i. Firmness: A program should not have any bugs that inhibit its function.
ii. Commodity: A program should be suitable for the purposes for which it was intended.
iii. Delight: The experience of using the program should be pleasurable one.

4. Define Abstraction ant types of abstraction?

Abstraction is the act of representing essential features without including the background details or explanations.

A *data abstraction* is collection of data that describes a data object.

A *procedural abstraction* refers to a sequence of instructions that have a specific and limited function. An example of a procedural abstraction would be the word *open* for a door.

5. Define Cohesion & Coupling (OR) What is Coupling? How it differs from cohesion (OR)
What is Cohesion? How it differs from coupling?

Cohesion is a measure that defines the degree of intra-dependability within elements of a module. The greater the cohesion, the better is the program design.

Coupling is a measure that defines the level of inter-dependability among modules of a program. It tells at what level the modules interfere and interact with each other. The lower the coupling, the better the program.

6. Define Refactoring and Aspect.

Refactoring – process of changing a software system in such a way internal structure is improved without altering the external behavior or code design

"Refactoring is the process of changing a software system in such a way that it does not alter the external behavior of the code [design] yet improves its internal structure."

Aspects – a representation of a cross-cutting concern that must be accommodated as refinement and modularization occur

7. What are the object oriented design concepts.

Object Oriented Design Concepts

i. Design classes
   a. Entity classes (business classes)
   b. Boundary classes (interfaces)
   c. Controller classes (controller)
ii. Inheritance—all responsibilities of a super class is immediately inherited by all subclasses
iii. Messages—stimulate some behavior to occur in the receiving object
iv. Polymorphism—a characteristic that greatly reduces the effort required to extend the design

8. What are the advantages of Modularization

i. Smaller components are easier to maintain
ii. Program can be divided based on functional aspects
iii. Desired level of abstraction can be brought in the program
iv. Components with high cohesion can be re-used again
v. Concurrent execution can be made possible
vi. Desired from security aspect

9. Define Design pattern concept

a. A design pattern "conveys the essence of a proven design solution to a recurring problem within a certain context of computing concerns."
b. The intent of each design pattern is to provide a description that enables a designer to determine:
   i. whether the pattern is applicable to the current work,
   ii. whether the pattern can be reused, and
   iii. whether the pattern can serve as a guide for developing a similar, but functionally or structurally different pattern.

10. What is Software Architecture?

The software architecture of a system is the structure or structures of the system which comprise of

i. *Software components*
ii. *External visible properties of those components*
iii. *Relationships among the components*

11. List out the different Architectural styles

Some of the architectural styles are:

i. Data-centered architectures
ii. Data flow architectures
iii. Call and return architectures
iv. Object-oriented architectures
v. Layered architectures

12. What is a Software Component

Definition: A software component is a modular, deployable, and replaceable part of a system that encapsulates implementation and exposes a set of interfaces.

Functions of a component: A component communicates and collaborates with

a. Other components
b. Entities outside the boundaries of the system

Three different views of a component

c. An object-oriented view
d. A conventional view
e. A process-related view

13. What is Domain Engineering?

i. **Intent is to identify, construct, catalog, and disseminate a set of software components** applicable to existing and future products in a particular application domain.
ii. **Overall goal** is to establish mechanisms that allow for component sharing and reuse during work on both new and existing systems
iii. Includes three major activities: **analysis, construction, dissemination**

14. Expand Design Concepts

List of design concepts are:

Abstraction	Architecture
Patterns	Separation of Concerns
Modularity	Information Hiding
Functional Independence	Refinement
Aspects	Refactoring
Object Oriented Design concepts	Design classes

15. Describe the role of software architecture in project development

The architecture of a system is a comprehensive framework that describes its form and structure. Good Software developers have often adopted one or several architectural patterns as strategies for system organization, but they use these patterns informally and have no means to make them explicit in the resulting system.

This would include articulating the architectural vision, conceptualizing and experimenting with alternative architectural approaches, creating models and component and interface specification documents, and validating the architecture against requirements and assumptions.

16. Define Software architecture with IEEE definition and its types.

The IEEE Standard defines an *architectural description* (AD) as a "a collection of products to document an architecture."

- The description itself is represented using multiple views, where each *view* is "a representation of a whole system from the perspective of a related set of [stakeholder] concerns."

The IEEE Computer Society has proposed IEEE-Std, *Recommended Practice for Architectural Description of Software-Intensive System,*

- to establish a conceptual framework and vocabulary for use during the design of software architecture,

17. Define Software design. Write different types of software design.

Software Design is the last software engineering action within the modeling activity and sets the stage for construction(code generation and testing). The types of software design are:

- ➔ Data/class Design
- ➔ Architectural Design
- ➔ Interface Design
- ➔ Component Level Design

UNIT – 4

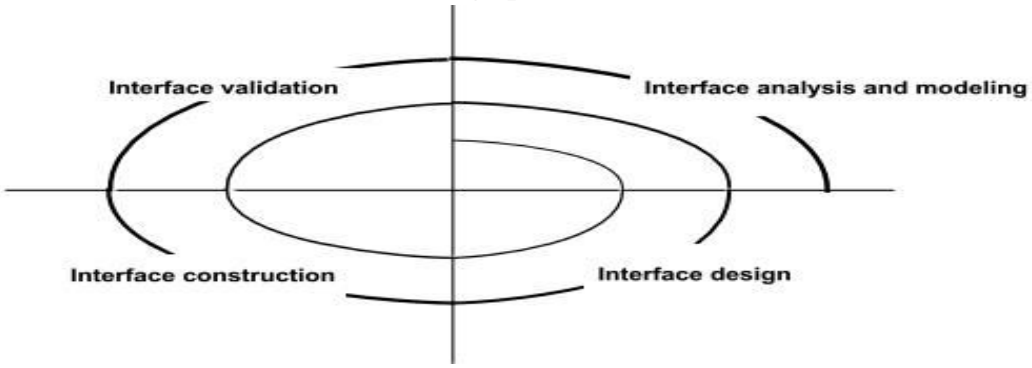
1. **State Golden rules of user interface design**
   - ➔ Place the user in control
   - ➔ Reduce the user's memory load
   - ➔ Make the interface consistent

2. **Why interface analysis is critical in UI development**
   - ➔ Allow the user to put the current task into a meaningful context.
   - ➔ Maintain consistency across a family of applications.
   - ➔ If past interactive models have created user expectations, do not make changes unless there is a compelling reason to do so.

3. **What are the steps for user interface analysis and design**

User model — a profile of all end users of the system
Design model — a design realization of the user model
Mental model (system perception) — the user's mental image of what the interface is
Implementation model — the interface "look and feel" coupled with supporting information that describe interface syntax and semantics.

4. **Draw the architecture for user interface design process**



5. **What is interface analysis**

Interface analysis means understanding

a. the people (end-users) who will interact with the system through the interface;
b. the tasks that end-users must perform to do their work,
c. the content that is presented as part of the interface
d. the environment in which these tasks will be conducted.

6. **What are the steps required for interface design steps**

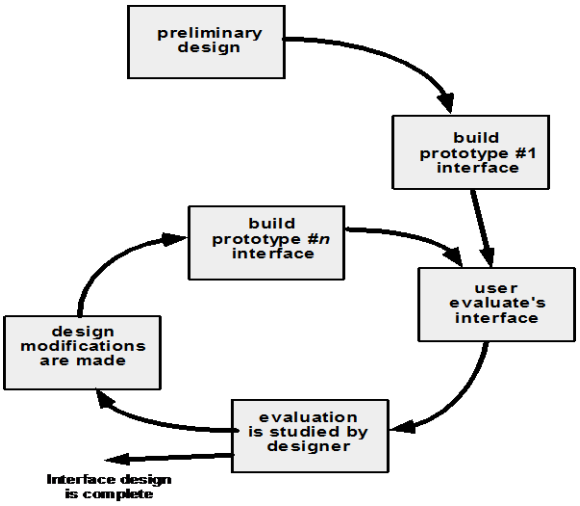
Using information developed during interface analysis, define interface objects and actions (operations).
Define events (user actions) that will cause the state of the user interface to change. Model this behavior.
Depict each interface state as it will actually look to the end-user.
Indicate how the user interprets the state of the system from information provided through the interface.

7. **List out four command design issues**

The 4 Command design issues are

1. Response time
2. Help Facilities
3. Error Handling
4. Menu and Command Labeling**.**

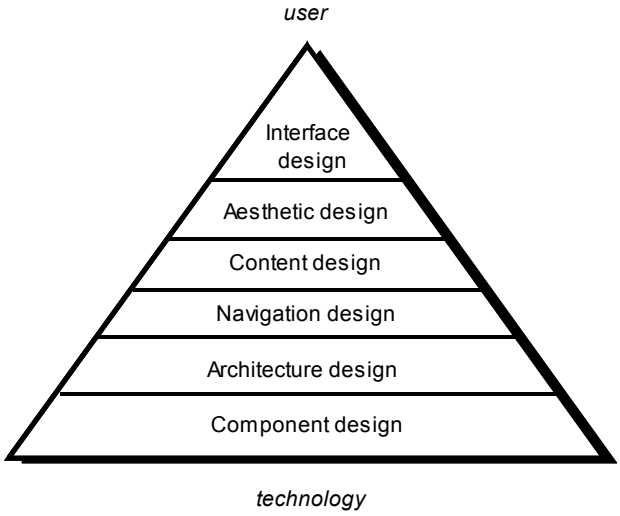
8. Design evaluation cycle for interface design



9. List the web application quality attributes

a. Usability
b. Functionality
c. Reliability
d. Efficiency
e. Maintainability

10. List various designs used in web application



11. What are the objectives of webapp design

➔ Establish a consistent window into the content and functionality provided by the interface.
➔ Guide the user through a series of interactions with the webapp
➔ Organize the navigation options and content available to the user.To implement navigation options, you can select from one of a number of interaction mechanisms:

a. *Navigation Menus*
b. *Graphic icons*
c. *Graphic images*

12. What is content architecture

➔ *Content architecture* focuses on the manner in which content objects (or composite objects such as Web pages) are structured for presentation and navigation.

a. The term information architecture is also used to connote structures that lead to better organization, labeling, navigation, and searching of content objects.

13. What is webapp architecture

-> *WebApp architecture* addresses the manner in which the application is structured to manage user interaction, handle internal processing tasks, effect navigation, and present content.

-> Architecture design is conducted in parallel with interface design, aesthetic design and content design.

14. Define MVC architecture

The *model* contains all application specific content and processing logic, including

a. all content objects
b. access to external data/information sources,
c. all processing functionality that are application specific

The *view* contains all interface specific functions and enables

a. the presentation of content and processing logic
b. access to external data/information sources,
c. all processing functionality required by the end-user.

The *controller* manages access to the model and the view and coordinates the flow of data between them.

15. Define Navigation Semantic Unit(NSU)

NSU—"a set of information and related navigation structures that collaborate in the fulfillment of a subset of related user requirements".

Navigation semantic unit

- Ways of navigation (WoN)—represents the best navigation way or path for users with certain profiles to achieve their desired goal or sub-goal.
    - Navigation nodes (NN) connected by Navigation links

16. What are the designs included in object oriented hypermedia design method

a. Conceptual design
b. Navigational design
c. Abstract Interface design
d. Implementation

UNIT – 5

1. **Who does the software testing and need of it.**
   - ➔ A strategy for software testing is developed by the project manager, software engineers, and testing specialists.
   - ➔ Testing often accounts for more project effort than any other software engineering action. If it is conducted haphazardly, time is wasted, unnecessary effort is expended, and even worse, errors sneak through undetected. It would therefore seem reasonable to establish a systematic strategy for testing software.

2. **Differentiate between Validation and Verification of a Software Product?**

   Verification refers to the set of tasks that ensure that software correctly implements a specific function.
   Validation refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements.
   Boehm states this another way:
   a. *Verification:* "Are we building the product right?"
   b. *Validation:* "Are we building the right product?"

3. **Discuss testing strategy for small and large software testing**
   - ➔ A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against customer requirements.
   - ➔ Testing is conducted by the developer of the software and (for large projects) an independent test group.
   - ➔ To perform effective testing, you should conduct effective technical reviews. By doing this, many errors will be eliminated before testing commences.

4. **Define Software Testing? List out the advantages of Software Testing?**

   Software Testing: Testing is the process of exercising a program with the specific intent of finding errors prior to delivery to the end user.

   Advantages of software testing:

   Software testing shows
   1. errors
   2. requirements conformance
   3. performance
   4. an indication of quality

5. **Explain how Unit testing of a Software System is performed?**

   Definition: Unit testing is a method by which individual units of source code are tested to determine if they are fit for use.

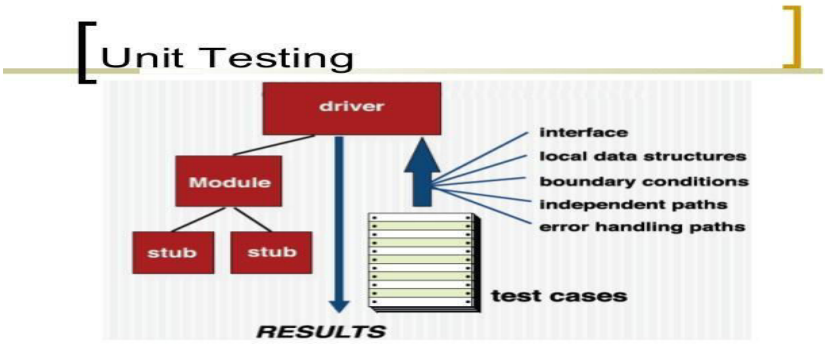
GOAL: The goal of unit testing is to segregate each part of the program and test that the individual parts are working correctly.

The following are activities are performed in unit testing:

i. Module interfaces are tested for proper information flow.
ii. Local data are examined to ensure that integrity is maintained.
iii. Boundary conditions are tested.
iv. Basis (independent) path are tested.
v. All error handling paths should be tested.

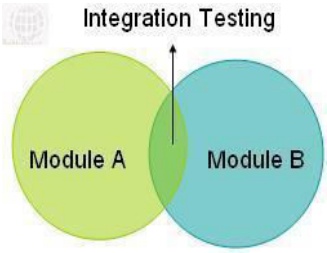
6. List out the outcome of unit testing

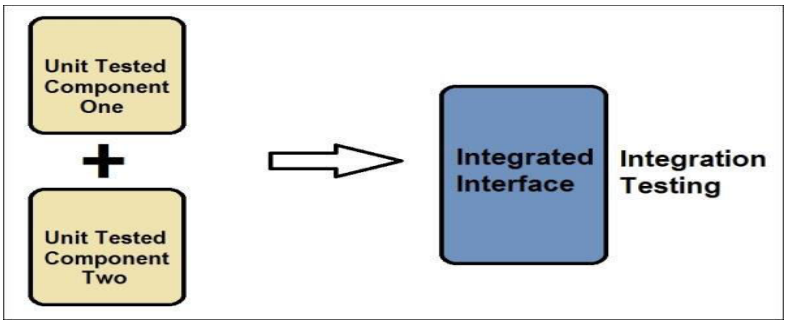
Unit testing focuses verification effort on the smallest unit of software design—the software component or module



7. What is Integration Testing?

Definition: Integration Testing is a type of software testing where individual units are combined and tested as a group. Integration Testing exposes defects in the interfaces and in the interactions between integrated components or systems.





8. Explain Smoke Testing?

Smoke testing: Smoke Testing, also known as "Build Verification Testing", is a type of software testing that comprises of a non-exhaustive set of tests that aim at ensuring that the most important functions work.

Features of smoke testing:

1. Identifying the business critical functionalities that a product must satisfy.
2. Designing and executing the basic functionalities of the application.

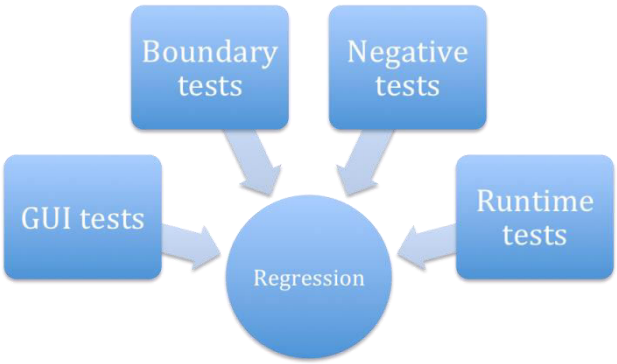
3. Ensuring that the smoke test passes each and every build in order to proceed with the testing.
4. Smoke Tests enables uncovering obvious errors which saves time and effort of test team.
5. Smoke Tests can be manual or automated.

9. Discuss Regression box testing?

Regression testing: it is used to check for defects propagated to other modules by changes made to existing program. Regression means retesting the unchanged parts of the application.

Test cases are re-executed in order to check whether previous functionality of application is working fine and new changes have not introduced any new bugs.

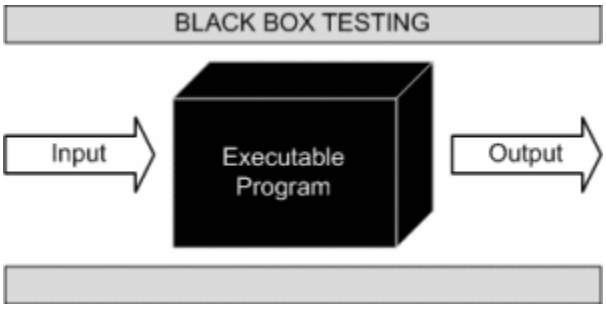
This test can be performed on a new build when there is significant change in original functionality or even a single bug fix.



10. Explain black box testing

Black Box Testing is a type of software Testing, either functional or non-functional, without reference to the internal structure of the component or system.

It also known as Behavioral Testing, is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester. These tests can be functional or non-functional, though usually functional.



11. Explain White box testing

White Box Testing is Testing based on an analysis of the internal structure of the component or system.

It is also known as Clear Box Testing, Open Box Testing, Glass Box Testing, Transparent Box Testing, Code-Based Testing or Structural Testing, it is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.

WHITE BOX TESTING ADVANTAGES

i. Testing can be commenced at an earlier stage. One need not wait for the GUI to be available.
ii. Testing is more thorough, with the possibility of covering most paths.

12. List out various system testing

a. Recovery Testing
b. Security Testing
c. Stress Testing
d. Performance Testing
e. Deployment Testing

13. What is the general Test criteria

Interface integrity – internal and external module interfaces are tested as each module or cluster is added to the software

Functional validity – test to uncover functional defects in the software

Information content – test for errors in local or global data structures

Performance – verify specified performance bounds are tested

14. Explain Acceptance testing

Making sure the software works correctly for intended user in his or her normal work environment.

There are two types of acceptance testing:

a. Alpha test
b. Beta test

Alpha test – version of the complete software is tested by customer under the supervision of the developer at the developer's site.
Beta test – version of the complete software is tested by customer at his or her own site without the developer being present.

15. Explain Debugging Strategies

Debugging is the process of finding and resolving of defects that prevent correct operation of computer software
Debugging (removal of a defect) occurs as a consequence of successful testing.
Common approaches (may be partially automated with debugging tools):

a. Brute force – memory dumps and run-time traces are examined for clues to error causes
b. Backtracking – source code is examined by looking backwards from symptom to potential causes of errors
c. Cause elimination – uses binary partitioning to reduce the number of locations potential where errors can exist)