St. Xavier's Catholic College of Engineering
Department of Computer Science and Engineering
CS6403 Software Engineering
Important Questions from Previous Year Question Papers

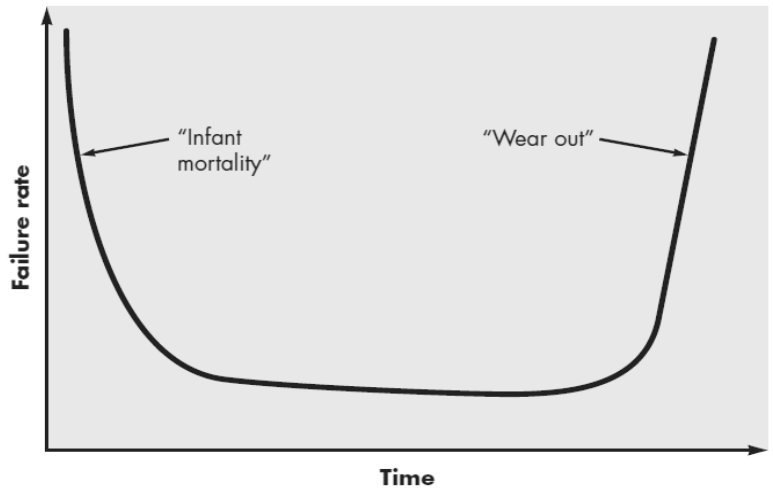
UNIT 1

PART - A

1. **What are two types of software products?**
   (1) Generic – These products are developed and to be sold to the range of different customers.
   (2) Custom – These type of products are developed and sold to the specific group of customers and developed as per their requirements.
2. **What is software engineering?**
   The application of a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software is called software engineering. (ie) The application of Engineering to Software.
3. **Software doesn't wear out. Justify.**

In the early stage of hardware development process the failure rate is very high because of manufacturing defects. The relationship, often called the "bathtub curve". But after correcting such defects the failure rate gets reduced. The failure rate remains constant for some period of time and again it starts increasing because of environmental maladies (extreme temperature, dusts, and vibrations).

On the other hand software does not get affected from such environmental maladies. Hence ideally it should have an "idealized curve". But due to some undiscovered errors the failure rate is high and drops down as soon as the errors get corrected. Hence in failure rating of software the "actual curve".



4. What is software process?

A structured set of activities required to develop a software system

- Specification
- Design
- Validation
- Evolution

A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

5. Write down the generic process framework activities that are applicable to any software project.

- ✓ Communication

  By communicating customer requirement gathering is done.
- ✓ Planning

  Establishes engineering work plan, describes technical risks, lists resource requirements, work products produced and defines work schedule
- ✓ Modelling

  The software model is prepared by:
  - Analysis of requirements
  - Design
- ✓ Construction

  The software design is mapped into a code by:
  - Code generation
  - Testing
- ✓ Deployment

  The software delivered for customer evaluation and feedback is obtained.

6. What is software process model? On what basis it is chosen?

The software process model can be defined as abstract representation of process. It is based on nature of software project.

7. How does "Project Risk" factor affect the spiral model of software development?

The Spiral model demands considerable risk assessment because if a major risk is not uncovered and managed, problems will occur in the project and then it will not be acceptable by end user.

8. State the benefits of waterfall life cycle model for software development.

- ❖ The waterfall model is simple to implement.
- ❖ For implementation of small systems the waterfall model is used.

9. Define the terms product and process in software engineering.

The product in software engineering is a standalone entity that can be produced by development organization and sold on the open market to any customer who is able to buy them. The software product consists of computer programs, procedures, and associated documentation (documentation can be in hard copy form or it may be in visual form). Some of the examples of software product are databases, word processors, drawing tools.

The process in software engineering can be defined as the structured set of activities that are required to develop software system. Various activities under software process are,

- ✓ Specification
- ✓ Design and implementation
- ✓ Validation
- ✓ Evolution

10. Identify in which phase of the software life cycle the following documents are delivered.

a) Architectural Design b) Test Plan c) Cost estimate d) Source code document

	Document	**Phase**
a)	Architectural design	Design
b)	Test Plan	Testing
c)	Cost estimate	Project management and planning
d)	Source code document	Coding

12. What is Software?

Software is nothing but a collection of computer programs that are related documents that are indented to provide desired features, functionalities and better performance.

13. Write out the reasons for the Failure of Water Fall Model.

- ✓ It is based on customer communication.
- ✓ It demands considerable risk assessment

14. What are the characteristics of the software?

- ✓ Software is engineered or developed, it is not manufactured in the classical sense.
- ✓ Software doesn't wear out.
- ✓ Although the industry is moving toward component based assembly, most software continues to be custom built.

15. What are the various categories of software?

- ✓ System software
- ✓ Application software
- ✓ Engineering/Scientific software
- ✓ Embedded software
- ✓ Web Applications
- ✓ Artificial Intelligence software

16. What are the challenges in software?

Software is dependent upon the programming language. This method is well designed but shorter program may get suffered. It does not accommodate non procedural languages. In early stage of development it is difficult to estimate LOC.

17. Define software process

A structured set of activities required to develop a software system

- Specification
- Design
- Validation
- Evolution

A software process model is an abstract representation of a process. It presents a description of a process from some particular perspective.

18. What are the major phases in the water fall model and spiral model? Where is spiral model beneficial?

- Requirement gathering phase in which all requirements are identified.
- The design phase is responsible for creating architectural view of the software.
- The implementation phase in which the software design is transformed into coding.
- Testing is a kind of phase in which the developed software component is fully tested.
- Maintenance is an activity by which the software product can be maintained.
- Requirements
- Design
- Implementation
- Testing
- Maintenance

19. What are the umbrella activities of a software process?

- Software project tracking and control
- Risk management
- Software quality assurance
- Formal technical reviews
- Software configuration management
- Work product preparation and production
- Reusability management.
- Measurement

20. What are the merits of incremental model?

- The incremental model can be adopted when tere are less number of people involved in the project.
- Technical risks can be managed with each increment.

- For a very small time span,at least core product can be delivered to the customer.

21. List the task regions in the Spiral model.

- Customer communication – In this region it is suggested to establish customer communication.
- Planning – All planning activities are carried out in order to define resources timeline and other project related activities.
- Risk analysis – The tasks required to calculate technical and management risks.
- Engineering – In this the task region, tasks required to build one or more representations of applications are carried out.
- Construct and release – All the necessary tasks required to construct, test and install the applications are conducted. Customer feedback is obtained and based on the customer evaluation required tasks are performed and implemented at installation stage.

22. What are the drawbacks of spiral model?

i. It is based on customer communication. If the communication is not proper then the software product that gets developed will not be the up to the mark.
ii. It demands considerable risk assessment. If the risk assessment is done properly then only the successful product can be obtained.

23. List out the available software risk.

- Performance risk—the degree of uncertainty that the product will meet its requirements and be fit for its intended use.
- Cost risk—the degree of uncertainty that the project budget will be maintained.
- Support risk—the degree of uncertainty that the resultant software will be easy to correct, adapt, and enhance.
- Schedule risk—the degree of uncertainty that the project schedule will be maintained and that the product will be delivered on time.

24. List the process maturity levels in SEIs CMM.

- 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.
- Level 3: Defined – The process is standardized, documented and followed.
- 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.

25. What is COCOMO model?

COnstructive COst MOdel is a cost model, which gives the estimate of number of man- months it will take to develop the software product. An empirical model based on

project experience. Well-documented, independent model which is not tied to a specific software vendor.

26. Define Reactive and proactive risk strategies.

A reactive strategy monitors the project for likely risks. the software team does nothing about risks until something goes wrong. Then, the team flies into action in an attempt to correct the problem rapidly. This is often called a fire-fighting mode.

A proactive risk strategy begins long before technical work is initiated. Potential risks are identified, their probability and impact are assessed, and they are ranked by importance. Then, the software team establishes a plan for managing risk. The primary objective is to avoid risk.

27. What is meant by LOC and FP based estimation?

LOC and FP estimation are distinct estimation techniques. A bounded statement of software scope is decomposed into problem functions that can each be estimated individually. LOC or FP (the estimation variable) is then estimated for each function. Baseline productivity metrics (e.g., LOC/pm or FP/pm) are then applied to the appropriate estimation variable, and cost are effort for the function derived. Function estimates are combined to produce and overall estimate for the entire project.

28. Define earned value analysis.

Earned Value Analysis is a technique of performing quantitative analysis of the software Project. It provides a common value scale for every task of software project. It acts as a measure for software project progress.

29. What is the difference between the "Known Risks" and Predictable Risks"?

Known risks are those that can be uncovered after careful evaluation of the project plan, the business and technical environment in which the project is being developed, and other reliable information sources (e.g., unrealistic delivery date, lack of documented requirements or software scope, poor development environment).

Predictable risks are extrapolated from past project experience (e.g., staff turnover, poor communication with the customer, dilution of staff effort as on-going maintenance requests are serviced).

30. What is CASE?

Software engineering tools provide automated or semi-automated support for the process and the methods. When tools are integrated so that information created by one tool can be used by another, a system for the support of software development, called **Computer-Aided Software Engineering**, is established.

PART - B

1. Explain in detail the project structure and programming team structure of a software organisation. (16)
2. Discuss any four process models with suitable applications. (16)
3. Discuss in detail various evolutionary process models. (16)
4. a) Discuss about the classic waterfall process model. (8)
   b) Explain the prototype paradigm in process models. (8)
5. Compare the following life cycle models based on their distinguishing factors, strengths and weaknesses - Waterfall Model, and Spiral Model.
6. Elaborate COCOMO model?
7. Explain in detail about the software project management.
8. Explain in detail about risk projection.
9. Narrate project scheduling in detail.
10. Explain in detail about LOC and FP based estimation

UNIT 2

PART - A

1. List the notations used in data flow models.

The notations used in data flow models are given below

a) Level 0 DFD i.e. Context level DFD should depict the system as a single bubble
b) Primary input and primary output should be carefully identified.
c) While doing the refinement isolate processes, data objects and data stores represent the next level.
d) All the bubbles (processes) and arrows should be appropriately named
e) One bubble at a time should be refined.
f) Information flow continuity must be maintained from level to level

2. List two advantages of traceability tables in the requirements management phase.

The two advantages of traceability tables are as given below,

[1] The traceability matrix ensures the coverage between different types of requirements
[2] The quick change impact analysis can be made using traceability matrix.

3. What does requirement process involve?

Requirements engineering process involves,

1) Elicitation
2) Analysis
3) Validation

4. Define functional and non-functional requirements.

Functional requirements:

Functional requirements should describe all the required functionality or system services. Functional requirements are heavily dependent upon the type of software, expected users and the type of system where software is used. Functional user

requirements may be high-level statements of what the system should do but functional system requirements should describe the system services in detail.

Non Functional requirements:

The non-functional requirements define system properties and constraints. Various properties of a system can be: Reliability, response time, storage requirements. And constraints of the system can be: Input and output device capability, system requirements etc. Process requirements may also specify programming language or development method.

5. List out requirements engineering.

Various activities in requirement engineering are,

- ✓ Requirements elicitation
- ✓ Requirements analysis
- ✓ Requirements validation
- ✓ Requirements management

6. How do we use the models that we create during requirement analysis?

Various models and their use during requirement analysis phase are given as below,

[1] Data model – The entity relationship model is created during the data modelling. The Entity Relationship Diagram defines the relationship between data objects.

[2] Functional model – The data flow diagrams are created in this modelling. The DFD depicts the information flow in three phases input, output and process.

[3] Behavioural model – The behavioural models are used to describe the overall behaviour of a system. The state transition diagrams are used to represent the behavioural model.

7. Identify ambiguities and omissions in the functional requirements. What questions would you ask to clarify these functional requirements?

Following are the questions that must be asked to clarify the functional requirements,

(1) Is the requirement specification clear? Can it be misinterpreted?
(2) What are the sources of requirements? Is the final requirement specification as per the original source?
(3) Is there any requirement violation for domain constructs?
(4) What are the other related requirements of the current requirement? And are they cross verified?
(5) For tracing the requirement is there any system model?
(6) Is the requirement testable?
(7) Has an index specification created for the requirements?
(8) Is the requirement satisfying the system objective?

8. What is the major distinction between user requirements and system requirements?

The user requirements describe both the functional and non-functional requirements in such way that they are understandable by the users who do not have detailed technical knowledge. On the other hand the system requirements are more detailed specifications of system functional, services and constraints than user requirements.

The user requirements are specified using natural languages tables and diagrams because these requirements can be understood by all users.

The system requirements can be expressed using system models.

9. Differentiate data flow diagram and state transition diagram.

Date flow diagram	State transition diagram
Data flow diagram is a graphical representation for representing the information flow and the transforms that are applied as data move from input to output.	State transition diagram is a graphical representation for representing the behaviour of a system by depicting its state and the events that cause the system to change state.
The data flow diagram is a collection of process, data store, flow of data (transitions) and external entity.	The state transition diagram is a collection of states and events.
The data flow diagrams are used to represent the system at any level of abstraction, and the increasing levels are used to expose more and more functionalities in the system.	The state transition diagrams are typically drawn at single level. They are intended to expose the overall behaviour of the system.

10. What is data dictionary? Where it is used in software engineering?

The data dictionary can be defined as an organized collection of all the data elements of the system with precise and rigorous definitions so that user and system analyst will have a common understanding of inputs, outputs, components of stores and intermediate calculations.

11. What is requirement elicitation?

Requirements elicitation is the practice of collecting the requirements of a system from users, customers and other stakeholders. The practice is also sometimes referred to as "requirement gathering". It is the definition of the system in terms understood by the customer. Requirements specification uses natural language and it is the communication with clients and users.

12. What is requirement engineering?

Requirement engineering is the process of establishing the services that the customer requires from the system and the constraints under which it operates and is developed.

13. What are the various types of traceability in software engineering?

i. Source traceability – These are basically the links from requirement to stakeholders

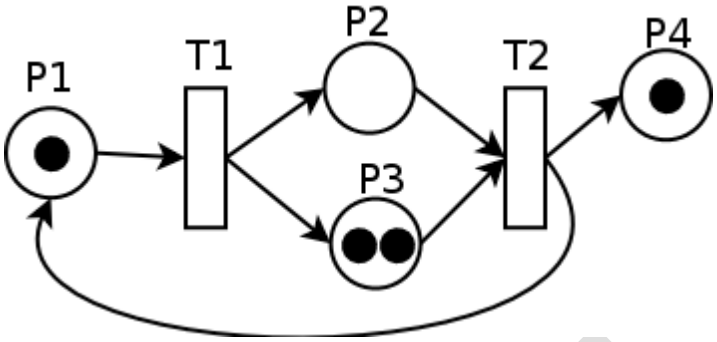
ii. Requirements traceability – These are links between dependant requirements.

iii. Design traceability – These are links from requirements to design.

14. Define Petri Net.

A Petri net (also known as a place/transition net or P/T net) is one of several mathematical modelling for the description of distributed systems. A Petri net is a directed bipartite graph, in which the nodes represent transitions (i.e. events that may occur, signified

by bars) and places (i.e. conditions, signified by circles). The directed arcs describe which places are pre- and/or postconditions for which transitions (signified by arrows).



15. What are the Difficulties in Elicitation?

- ❖ Requirements elicitation is a complex and imprecise process that varies greatly for different projects.
- ❖ Many technical problems are related to this process:
- ❖ Scope problems
- ❖ Problems related to the nature of computer science
- ❖ Problems related to the process itself
- ❖ Many problems pertaining to human nature are related to this process:
- ❖ Articulation problems
- ❖ Communication barriers
- ❖ Knowledge and cognitive limitations
- ❖ Human behaviour issues

16. What is System Modelling?

System modelling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system. System modelling helps the analyst to understand the functionality of the system and models are used to communicate with customers.

17. What are the characteristics of SRS?

- ➢ Correct – The SRS should be made up to date when appropriate requirements are identified.
- ➢ Unambiguous – When the requirements are correctly understood then only it is possible to write an unambiguous software.
- ➢ Complete – To make SRS complete,it shold be specified what a software designer wants to create software.
- ➢ Consistent – It should be consistent with reference to the functionalities identified.
- ➢ Specific – The requirements should be mentioned specifically.
- ➢ Traceable – What is the need for mentioned requirement?This should be correctly identified.

18. What are the objectives of Analysis modeling?

- To describe what the customer requires.
- To establish a basis for the creation of software design.
- To devise a set of valid requirements after which the software can be built.

19. What is data modeling?

Data modeling is the basic step in the analysis modeling. In data modeling the data objects are examined independently of processing. The data model represents how data are related with one another.

20. What is a data object?

Data object is a collection of attributes that act as an aspect, characteristic, quality, or descriptor of the object.

21. What are attributes?

Attributes are the one, which defines the properties of data object.

PART - B

1. Explain in detail about,
   i) Functional requirements
   ii) Non-Functional requirements
   iii) User requirements
   iv) System requirements
2. Write short note on SRS ii) Explain the metrics used for specifying non-functional requirements. (8)
3. Name the importance of software specification of requirements. Explain typical SRS structure and its parts. (16)
4. Explain i) Requirement Elicitation and analysis ii) Requirement Validation and iii) Requirement Management. (16)
5. Explain Classical analysis : Structured system analysis
6. Explain
   i. Petri nets
   ii. Data Dictionary

UNIT 3

PART - A

1. What is software architecture?

Software architecture is a structure of systems which consists of various components, externally visible properties of these components and inter-relationships among these components

2. Define data abstraction.

Data abstraction is a kind of representation of data in which the implementation details are hidden.

3. What is design quality attributes 'FURPS' meant?

The design attributes FURPS stands for,

- **F**unctionality: Functionality can be checked by assessing the set of features and capabilities of the functions.
- **U**sability: The usability can be assessed by knowing the usefulness of the system.
- **R**eliability: Reliability is a measure of frequency and severity of failure.
- **P**erformance: Performance is a measure that represents the response of the system.
- **S**upportability: Supportability is also called maintainability. It ia ability to adopt the enhancement or changes made in the software..

4. List out design methods.

The two software design methods are,

1. Object oriented design
2. Function oriented design

5. List the architectural models that can be developed.

Following are the architectural models that can be developed,

[1] Data cantered architectural
[2] Data flow architectures
[3] Call and return architecture
[4] Object oriented architecture
[5] Layered architectures

6. **Define interface design**.

Interface design is an iterative process in which each design process occurs more than once. Each time the design step gets elaborated in more detail.

7. What is system design?

System design is process of translating customer's requirements into a software product layout. In this process a system design model is created.

8. If a module has logical cohesion what kind of coupling is this module likely to have with others?

When a module that performs a tasks that are logically related with each other is called logically cohesive. For such module content coupling can be suitable for coupling with other modules. The content coupling is a kind of coupling when one module makes use of data or control information maintained in other module.

9. Differentiate cohesion and coupling.

Coupling	Cohesion
Coupling represents how the modules are connected with other modules or with the outside world	In cohesion, the cohesive module performs only one thing.
With coupling interface complexity is decided	With cohesion data hiding can be done
The goal of coupling is to achieve lowest coupling	The goal of cohesion is to achieve high cohesion
Various types of coupling are: Data coupling, Control coupling, Common coupling, and Content coupling.	Various types of cohesion are: Coincidental cohesion, Logical cohesion, Temporal cohesion, Procedural cohesion, and Communicational cohesion.

10. Why it is necessary to design the system architecture before specifications are completed?

The system architecture defines the role of hardware, software, people, database procedures and other system elements. Designing of system architecture will help the developer to understand the system as a whole. Hence system architecture must be built before specifications are completed.

11. **What are the elements of Analysis model?**

i. Data Dictionary
ii. Data flow diagram
iii. Entity relationship diagram
iv. State transition diagram
v. Control specification
vi. Process specification

12. What are the elements of design model?

i. Data design elements
ii. Architectural design elements
iii. Interface design elements
iv. Component level design elements
v. Deployment level design elements

13. How the Architecture Design can be represented?

Architecture Design can be represented as,

i. Analyze the effectiveness of the design in meeting its stated requirements,
ii. Consider architectural alternatives at a stage when making design changes is still relatively easy
iii. Reduce the risks associated with the construction of the software.

14. Define design process.

Software design is an iterative process through which requirements are translated into a "blueprint" for constructing the software. The design is represented at a high level of abstraction, a level that can be directly traced to the specific system objective and more detailed data, functional, and behavioral requirements.

15. What is the benefit of modular design?

The benefits of modular design are,

i. Easily planned
ii. Software increments can be defined and delivered
iii. Changes can be more easily accommodated
iv. Testing and debugging can be conducted more efficiently
v. Long-term maintenance can be conducted without serious side effects.

16. What is a cohesive module?

A cohesive module performs a single task, requiring little interaction with other components in other parts of a program.

17. What are the different types of Cohesion?

i. Functional
ii. Layer
iii. Communicational

18. What is coupling?

Coupling is a qualitative measure of the degree to which classes are connected to one another. As classes and components become more interdependent, coupling increases. An important objective in component-level design is to keep coupling as low as is possible.

19. What are the various types of coupling?

i. Content coupling
ii. Common coupling
iii. Control coupling
iv. Stamp coupling
v. Data coupling
vi. Routine call coupling
vii. Type use coupling
viii. Inclusion or import coupling
ix. External coupling

20. What are the common activities in design process?

i. The data/class design
ii. The architectural design
iii. The interface design
iv. The component-level design

21. What are the benefits of horizontal partitioning?

The benefits of horizontal partitioning are,

i. Easy to test, maintain and extend.
ii. Side effects in change propagation or error propagation

22. What is vertical partitioning?

Vertical partitioning defines separate branches of the module hierarchy for each major function. It use control modules to co-ordinate communication between functions.

23. What are the advantages of vertical partitioning?

i. Easy to maintain the changes
ii. Reduce the change impact and error propagation

24. What are the various elements of data design?

The various elements of data design are,

i. Data object
ii. Databases
iii. Data Warehouses

25. List the guidelines for data design.

i. Apply systematic analysis on data
ii. Identify data structures and related operations
iii. Establish data dictionary
iv. Defer the low-level design decisions until late in the design process
v. Use information hiding in the design of data structures
vi. Apply a library of useful data structures and operations
vii. Use a software design and programming language to support data specification and abstraction.

26. Name the commonly used architectural styles.

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

27. What is Transform mapping?

Transform mapping is a set of design steps that allows a DFD with transform flow characteristics to be mapped into a specific architectural style.

28. Define product Engineering

Product engineering refers to the process of designing and developing a device, assembly, or system such that it is produced as an item for sale through some production manufacturing process.

PART - B

1. Explain in detail about any four architectural styles. (16)
2. Explain in detail about
    1) Design process
    2) Design Concepts
    3) Design model
3. Explain various modular decomposition and control styles commonly used in any organizational model. (16)
4. i) Explain the golden rules of interface design. (8)
   ii) Discuss the design heuristics for effective modularity design. (8)
5. Describe transform and transactional mapping by applying design steps to an example system. (16)
6. Describe the architectural design of software. (16)
7. Describe Component level Design:
    [1] Designing class based components
    [2] Traditional components
8. Describe User Interface design
    [1] Interface analysis
    [2] Interface Design

UNIT 4

PART - A

1. **Distinguish between verification and validation**.

Verification	Validation
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
Are we building the product right?	Are we building the right product?
After a valid and complete specification the verification starts	Validation begins as soon as project starts.
The verification is conducted to ensure whether software meets specification or not.	Validation is conducted to show that the user requirements are getting satisfied.

2. **Write down generic characteristics of software testing**.

Following are the characteristic s of software testing:

i. Testing should be conducted by the developer as well as by some independent group.

ii. The testing must begin at the component level and must work outwards towards the integration testing

iii. Different testing must be appropriate at appropriate times

3. What are the classes of loops that can be tested?

Various classes of loop testing are:

a) Simple loop
b) Nested loop
c) Concatenated loop
d) Unstructured loop

4. What are the levels at which the testing is done?

Testing can be done in the two levels as given below:

- ✓ Component level testing – In the component level testing the individual component is tested.
- ✓ System level testing – In system testing, the testing of group of components integrated to create a system or subsystem is done.

5. How are the software testing results related to the reliability of the software?

During the software testing the program is executed with the intension of finding as much errors as possible. The test cases are designed that are intended to discover yet-undiscovered errors. Thus after testing, majority of serious errors are removed from the system and it is turned into the model that satisfied user requirements.

6. Distinguish alpha testing and beta testing.

Alpha testing	Beta testing
Alpha testing is done by a developer or by a customer under the supervision of developer in company's premises.	Beta testing is done by the customer without any interference of developer and is done at customer's place.
Sometime full product is not tested using alpha testing and only core functionalities are tested	The complete product is tested under this testing. Such product is usually given as free trial version.

7. Distinguish between stress testing and load testing.

Load testing	Stress testing
Load testing is conducted to measure system performance based on volume of users.	Stress testing is conducted for measuring the breakpoint of the system when extreme load is applied.
In load testing the expected load is applied to measure the performance of the system.	In stress testing the extreme load is applied.

8. What is big bang approach?

Big-Bang approach is used in non-incremental integration testing. In this approach of integration testing, all the components are combined in advance and then entire program is tested as a whole.

9. State the guidelines for debugging.

Following are some debugging guideline suggested by Myers;

[1] Debugging can be effectively carried out by mental analysis.
[2] If an error cannot be located by yourself, then describe the problem to someone else.
[3] Use debugging tool only if no alternative is present.
[4] Avoid experimentation.
[5] If one bug occurs in one area of code then there are chances of occurrence of more bugs.
[6] Fix the error and not its symptoms.
[7] For correcting the existing program, if a new code is added to it then test it rigorously than the original program.
[8] There are chances of introducing new errors are correction of older ones.
[9] The process error repair must be conducted in parallel during design phase.
[10] Change the source code and not the object code/machine code.

10. Define software testing?

Software testing is a critical element of software quality assurance and represents the ultimate review of specification, design, and coding.

11. Define Smoke Testing?

Smoke testing is an integration testing approach that is commonly used when product software is developed. It is designed as a pacing mechanism for time-critical projects, allowing the software team to assess the project on a frequent basis.

12. What are the objectives of testing?

- ❖ Testing is a process of executing a program with the int end of finding an error.
- ❖ A good test case is one that has high probability of finding an undiscovered error.
- ❖ A successful test is one that uncovers as an-yet undiscovered error.

13. What are the testing principles must be applied while performing the software testing?

- ✓ All tests should be traceable to customer requirements.
- ✓ Tests should be planned long before testing begins.
- ✓ The pareto principle can be applied to software testing-80% of all errors uncovered during testing will likely be traceable to 20% of all program modules.
- ✓ Testing should begin "in the small" and progress toward testing "in the large". v. Exhaustive testing is not possible.
- ✓ To be most effective, an independent third party should conduct testing.

14. Define White Box Testing?

White-box testing of software is predicated on close examination of procedural detail. Logical paths through the software and collaborations between components are tested by exercising specific sets of conditions and/or loops.

15. What are the various testing activities?

- Test planning
- Test case design
- Test execution
- Data collection
- Effective evaluation

16. Write short note on black box testing.

The black box testing is also called as behavioral testing. This method fully focus on the functional requirements of the software. Tests are derived that fully exercise all functional requirements.

17. What is equivalence partitioning?

Equivalence partitioning is a black box technique that divides the input domain into classes of data. From this data test cases can be derived. Equivalence class represents a set of valid or invalid states for input conditions.

18. What is Regression Testing?

Regression testing is a type of software testing that seeks to uncover new software bugs, or regressions, in existing functional and non-functional areas of a system after changes such as enhancements, patches or configuration changes, have been made to them. The intent of regression testing is to ensure that changes such as those mentioned above have not introduced new faults. One of the main reasons for regression testing is to determine whether a change in one part of the software affects other parts of the software.

19. What is a boundary value analysis?

A boundary value analysis is a testing technique in which the elements at the edge of the domain are selected and tested. It is a test case design technique that complements equivalence partitioning technique.

20. What are the reasons behind to perform white box testing?

There are three main reasons behind performing the white box testing.

1. Programmers may have some incorrect assumptions while designing or implementing some functions.

2. Certain assumptions on flow of control and data may lead programmer to make design errors. To uncover the errors on logical path, white box testing is must.

3. There may be certain typographical errors that remain undetected even after syntax and type checking mechanisms. Such errors can be uncovered during white box testing.

21. What is cyclomatic complexity?

Cyclomatic complexity is software metric that gives the quantitative

Measure of logical complexity of the program.

The cyclomatic complexity can be computed by any one of the following ways.

[1] The numbers of regions of the flow graph correspond to the cyclomatic complexity.
[2] Cyclomatic complexity (G), for the flow graph G, is defined as: V(G)=E-N+2, E -- number of flow graph edges, N -- number of flow graph nodes
[3] V(G) = P+1 Where P is the number of predicate nodes contained in the flow graph..

22. What are the various testing strategies for conventional software?

a. Unit testing
b. Integration testing.
c. Validation testing.
d. System testing.

23. Write about drivers and stubs.

Drivers and stub software need to be developed to test incompatible software.

- ✓ The "**driver**" is a program that accepts the test data and prints the relevant results.
- ✓ The "**stub**" is a subprogram that uses the module interfaces and performs the minimal data manipulation if required.

24. What are the approaches of integration testing?

The integration testing can be carried out using two approaches.

a. The non-incremental testing.
b. Incremental testing.

25. What are the advantages and disadvantages of big-bang?

Advantage: This approach is simple.

Disadvantages: It is hard to debug.

It is not easy

PART - B

1. i) Explain in detail about basic path testing. (8)
   ii) What is equivalence class partitioning? List the rules used to define valid and invalid equivalence classes. Explain the technique using examples. (8)
2. i) Explain in detail about Integration testing. (8)
   ii) What is boundary value analysis? Explain the technique specifying the rules and its usage with the help of an example. (8)
3. Illustrate black box testing with an example. Write a short note on Unit testing and debugging. (16)
4. Explain in detail regression testing.
5. Explain system testing and art of debugging in detail.
6. Explain refactoring
7. Explain in detail about integration testing and validation testing? With an example. (16)
8. i) Explain white box testing strategies. (8)
   ii) How do you test boundary conditions? (8)

UNIT 5

PART - A

1. **What is error tracking?**
   Error tracking is a process of finding out and correcting the errors that may occur during the software development process at various stages such as software design, coding or documenting.
2. **What is scheduling?**

Scheduling is the activity that distributes estimated effort across the planned project duration. These efforts are related to software engineering tasks.

3. **What is risk? Give example of risk.**
   The software risk can be defined as that can cause some loss or then threaten the success of the project. Example: Experienced and skilled staff leaving the organization in between.
4. **Define software quality.**
   The software quality can be defined as the degree to which a system component or process meets specific requirements.
5. **What do you mean by estimation risk?**
   Estimation risk is measure by the degree of uncertainty in the quantitative estimates for cost, schedule and resources.
6. **List out the different approaches to size of the software.**
   The two approachs used for estimating the size of the software are:
   i. LOC – computing the lines of code.
   ii. FP – computing function point of the program.

7. Differentiate between size oriented and function oriented metrics.

Size oriented metrics	Function oriented metrics
Size oriented software metrics is by considering the size of the software that has been produced.	Function oriented metrics use a measure of functionality delivered by the software.
For a size oriented metric the software organization maintains simple records in tabular form. The typical table entries are: Project name, LOC, effort, Pages of documents, errors, defects, total number of people working on project.	Most widely used function oriented metric is the function point (FP) computation of the function point is based on characteristics of software's information domain and complexity.

8. What are project indicators and how do they help a project manager?

Project indicators mean combination of metrics that provides insight into the software process, project or product. An indicator is a tool that helps you and your organization know how far your project is form achieving your goals and whether you are headed in the right direction.

9. What are software planning activities?

The project planning activities can be carried out in following steps:

[1] Identify the constraints in the project.
[2] Make initial assessment of the project.
[3] Define project milestones and deadlines.
[4] While developing the project,
  i) Prepare project schedule.
  ii) Initiate project activities as per the project schedule.
  iii) Review progress of the project.
  iv) Update project schedule if required.
  v) Revise project constraints and deliverables
  vi) If any problem arises then perform technical review.

[5] Repeat the step [4] until the project is ready.

10. Define debugging.

11. What are the common approaches in debugging?

12. Write about the types of project plan.

13. Define measure.

14. Define metrics.

15. What is meant by Make/Buy decision?

16. What are the advantages and disadvantages of size measure?

17. Write short note on the various estimation techniques.

18. What is the Objective of Formal Technical Reviews?

19. What is COCOMO II model?

20. Give the procedure of the Delphi method.

21. What is the purpose of timeline chart?

22. What is EVA?

23. What are the metrics computed during error tracking activity?

24. Why software change occurs?

25. Write about software change strategies.

26. Define RFP risk Management.

27. What is risk scheduling and tracking?

28. Define RMMM.

29. What are the types of software maintenance?

PART - B

1. i) Write short notes on (a) COCOMO estimation criteria and (b) Software metrics (8)
   ii) Describe function point analysis with a neat example. (8)
2. i) Explain the scheduling of the software project. (8)
   ii) How is earned value computed to assess the progress? (8)
3. Using COCOMO, estimate time required for the following: (16)
   i) A semi-detached model of software project of 2000lines.
   ii) An embedded model of software of 30,000 lines.
   iii) An organic model of software of one lakh lines.
   iv) An organic model of software of 10 lakh lines.
4. Describe two metrics which have been used to measure the software. Discuss clearly the advantages and disadvantages. (16)
5. Explain the need for software measures and describe various metrics. (16)
6. Explain about software cost estimation.
7. Explain in detail about COCOMO II model.
8. Explain in detail about Delphi Method.
9. Explain in detail about software Maintenance.
10. Explain about RMMM.
11. Elaborate FP and Loc based estimation.
12. Explain scheduling and error tracking
13. Explain about project planning
14. Explain about risk management
15. Explain on software configuration management