EASWARI ENGINEERING COLLEGE
DEPARTMENT OF INFORMATION TECHNOLOGY

CS6403 SOFTWARE ENGINEERING

QUESTION BANK

JAN 2015 TO APRIL 2015

YEAR: II **SEMESTER: IV**

PREPARED BY	**APPROVRED BY**
M.MOHANA	**HOD**
S. GNANAPRIYA	

CS6403 SOFTWARE ENGINEERING

UNIT I - SOFTWARE PROCESS AND PROJECT MANAGEMENT

PART A

1. What are the characteristics of the software?

Characteristics of the software are:

- Software is engineered, not manufactured.
- Software does not wear out.
- Most software is custom built rather than being assembled from components.

2. What are the various categories of software?

The various categories of software are:

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

3. List the challenges in software?

The challenges in software are:

- Copying with legacy systems.
- Heterogeneity challenge.
- Delivery times challenge.

4. What is computer software?

Computer software is

- A data structure that enables the programs to adequately manipulate information and
- Descriptive information in both hard copy and virtual form that describes the operation and use of the program.

5. **Define Software Engineering.**

Software Engineering is defined as the application of systematic, disciplined, quantified approach to the development, operations, and maintenance of software.

6. **Define Software process.**

Software process is defined as the structured set of activities that are required to develop the software system.

7. **What are the fundamental activities of a software process?**

The fundamental activities of a software process are:

- Specification
- Design and Implementation
- Validation
- Evolution

8. **What is a Process Framework?**

Process Framework establishes foundation for a complete software process by identifying a small number of framework activities that are applicable for all software projects regardless of their size and complexity

9. **What are the Generic Framework Activities?**

Generic Framework Activities are:

- Communication.
- Planning.
- Modeling.
- Construction.
- Deployment.

10. **What are the umbrella activities of a software process?**

The umbrella activities of a software process are:

- Software project tracking and control.
- Risk Management.
- Software Quality Assurance.
- Formal Technical Reviews.
- Software Configuration Management.

- Work product preparation and production.
- Reusability management, Measurement.

11. Write out the reasons for the Failure of Water Fall Model?

Reasons for the Failure of Water Fall Model are:

- Real project rarely follow sequential Flow. Iterations are made in indirect manner.
- Difficult for customer to state all requirements explicitly.
- Customer needs more patients as working products reach only at deployment phase.

12. Write the disadvantages of classic life cycle model.

Disadvantages of classic life cycle model are:

- Real projects rarely follow sequential flow. Iteration always occurs and creates problem.
- Difficult for the customer to state all requirements.
- Working version of the program is not available. So the customer must have patience.

13. What do you mean by task set in spiral Model?

Each of the regions in the spiral model is populated by a set of work tasks called a task set that are adopted to the characteristics of the project to be undertaken.

14. Which of the software engineering paradigms would be most effective? Why?

Incremental / Spiral model will be most effective.

Reasons:

- It combines linear sequential model with iterative nature of prototyping.
- Focuses on delivery of product at each increment.
- Can be planned to manage technical risks.

15. What are the merits of incremental model?

The merits of incremental model are:

- The incremental model can be adopted when there is 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.

16. List the task regions in the Spiral model.

Task regions in the Spiral model are:

- 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.
- Construct and Release: All the necessary tasks required to construct, test, and install the applications are conducted.
- Customer Evaluation: Customer's feedback is obtained and based on the customer evaluation required tasks are performed and implemented at installation stage.

17. What are the drawbacks of spiral model?

The drawbacks of spiral model are:

- 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.
- It demands considerable risk assessment. If the risk assessment is done properly then only the successful product can be obtained.

18. Name the Evolutionary process Models.

Evolutionary powers models are:

- Incremental model
- Spiral model
- WIN-WIN spiral model
- Concurrent Development

19. Define Software Prototyping.

Software prototyping is defined as a rapid software development for validating the requirements.

20. What are the benefits of prototyping?

The benefits of prototyping are :

- Prototype services as a basis for deriving system specification.
- Design quality can be improved.
- System can be maintained easily.
- Development efforts may get reduced.
- System usability can be improved.

21. What are the prototyping approaches in software process?

The prototyping approaches in software process are :

- Evolutionary prototyping : In this approach of system development, the initial prototype is prepared and it is then refined through number of stages to final stage.
- Throw-away prototyping : Using this approach a rough practical implementation of the system is produced. The requirement problems can be identified from this implementation. It is then discarded. System is then developed using some different engineering paradigm.

22. What are the advantages of evolutionary prototyping ?

The advantages of evolutionary prototyping are :

- Fast delivery of the working system.
- User is involved while developing the system.
- More useful system can be delivered.
- Specification, design and implementation work in co-ordinated manner.

23. What is software project management?

Software project management is process of managing all activities like time, cost and quality management involved in software development.

24. Define Stakeholder.

Stakeholder is anyone who has stake in successful outcome of project such as:

- Project Managers,
- Senior Managers
- Customers
- Practitioners
- End Users

25. What is software scope?

Software scope is a well-defined boundary, which encompasses all the activities that are done to develop and deliver the software product.

The software scope clearly defines all functionalities and artifacts to be delivered as a part of the software. The scope identifies what the product will do and what it will not do, what the end product will contain and what it will not contain.

26. What is project estimation?

It is a process to estimate various aspects of software product in order to calculate the cost of development in terms of efforts, time and resources. This estimation can be derived from past experience, by consulting experts or by using pre-defined formulas.

27. How can we derive the size of software product?

Size of software product can be calculated using either of two methods:

- Counting the lines of delivered code
- Counting delivered function points

28. What are function points?

Function points are the various features provided by the software product. It is considered as a unit of measurement for software size.

29. What are software project estimation techniques available?

There are many estimation techniques available. The most widely used are -

- Decomposition technique (Counting Lines of Code and Function Points)
- Empirical technique (Putnam and COCOMO).

30. What are the Decomposition Techniques?

Decomposition Techniques are:

- Software Sizing
- Problem — Based Estimation
- Process — Based Estimation
- Estimation with Use Cases.
- Reconciling Estimates.

31. How do we compute the "Expected Value" for Software Size?

Expected value for estimation variable (size), S, can be compute as Weighted Average of Optimistic (opt) most likely (m), and Pessimistic (pess) estimates.

$$S = (opt + 4\ m + pess\ ) / 6$$

32. What is an Object Point?

Object Point means: Count is determined by multiplying original number of object instances by weighting factor and summing to obtain total object point count.

33. List out the basic principles of software project scheduling?

Basic Principles of Software Project Scheduling are:

- Compartmentalization
- Interdependency
- Time Allocation
- Effort Validation
- Defined Responsibilities Defined Outcomes
- Defined Milestones.

34. 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.

35. Write the objective of project planning?

It is to provide a framework that enables the manager to make reasonable estimates of resources, cost and schedule.

36. What is FP? How it is used for project estimation?

Function Point: It Function Point is used as the estimation variable to size each element of the software. It requires considerably less details. It is estimated indirectly by estimating the number of inputs, outputs, data files, external interfaces.

37. What is LOC? How it is used for project estimation?

LOC: Lines of Code. It is used as estimation variable to size each element of the software. It requires considerable level of detail.

38. What is 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 the software project. It acts as a measure for software project progress

39. What Is Risk?

Risks are events that are usually beyond the planner's control.

40. What is Risk management?

Risk management is the process of anticipating hurdles in carrying out the original plan and providing alternate plans so that the impact on the originally anticipated final outcome is minimal.

41. Give the two important characteristics of the risk management?

The two important characteristics of risk management are:

- It is proactive.
- It strives to reduce the impact of uncertainty.

42. What are the three phases of Risk management?

The three phases of risk management are:

- Risk identification,
- Risk Quantification,
- and Risk mitigation.

43. What are the ways of identifying the potential risks?

The ways of identifying the potential risks are:

Examining organizational history, preparing checklists, information buying, framework based risk categorization, simulation, Decision trees.

44. What are the factors that lead to Risk?

The factors that lead to Risk are:

- Estimation errors.
- Planning assumptions.
- Business risks.

45. What are the various steps under risk analysis?

The various steps under risk analysis are:

- Risk Estimation.
- Risk identification.
- Risk evaluation.

PART B

1. Explain iterative waterfall and spiral model for software life cycle and various activities in each phase. (16)
2. Explain about the incremental model. (16)
3. Explain in detail about the software process. (16)
4. Explain in detail about the life cycle process. (16)
5. Explain Spiral model and win-win spiral model in detail? (16)
6. Explain about rapid prototyping techniques. (16)
7. Explain the prototyping approaches in software process. (16)

UNIT – II

REQUIREMENTS ANALYSIS AND SPECIFICATION

PART A

1. **What are the Objectives of Requirement Analysis?**

   Objectives of Requirement Analysis are :

   - It describes what customer requires.
   - It establishes a basis for creation of software design.
   - It defines a set of requirements that can be validated once the software design is built.

2. **What is Requirement Engineering?**

   Requirement engineering is the process of defining, analyzing and managing the system requirement is called requirement engineering

3. **Define System Context Diagram (SCD)?**

   System Context Diagram (SCD):

   - Establish information boundary between System being implemented and Environment in which system operates.
   - Defines all external producers, external consumers and entities that communicate through user interface.

4. **What are functional and non-functional requirements?**

   Functional

   - How the system should react to the particular inputs
   - How the system should behave to the particular situations
   - What the system should not do

   Non functional

   - Constraints on the services or functions
   - Time constraints
   - Constraints on the development process

5. **Define System Flow Diagram (SFD)?**

System Flow Diagram (SFD):

- Indicates Information flow across System Context Diagram region.
- Used to guide system engineer in developing system.

6. What are the Requirements Engineering Process Functions?

Requirements Engineering Process Functions are:

- Inception
- Elaboration
- Specification
- Management

Elicitation
Negotiation
Validation

7. What are the Difficulties in Elicitations?

Difficulties in Elicitation are:

- Problem of Scope
- Problem of Volatility
- Problem of Understanding

8. Define Quality Function Deployment (QFD)?

Quality Function Deployment (QFD) is a technique that translates needs of customer into technical requirement. It concentrates on maximizing customer satisfaction from the software engineering process.

9. What are the characteristics of SRS?

The characteristics of SRS are as follows:

Correct: The SRS should be made up the date when appropriate requirements are identified.

Unambiguous: When the requirements are correctly understood then only it is possible to write unambiguous software.

Complete: To make SRS complete, its hold 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.

10. What are the objectives of Analysis modeling ?

The objectives of analysis modeling are:

- 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 build.

11. What are the elements of analysis model?

The elements of analysis model are:

(i) Data Dictionary	(ii) Entity Relationship Diagram
(iii) Data flow Diagram	(iv) State Transition Diagram
(v) Control Specification	(vi) Process Specification

12. What is ERD?

Entity Relationship Diagram is the graphical representation of the object relationship pair. It is mainly used in database application.

13. What is DFD?

Data Flow Diagram depicts the information flow and the transforms that are applied on the data as it moves from input to output.

14. What does Level 0 DFD represent?

Level-0 DFD is called as fundamental system model or context model. In the context model the entire software system is represented by a single bubble with input and output indicated by incoming and outgoing arrows.

15. What is a state transition diagram?

State transition diagram is basically a collection of states and events. The events cause the system to change its state. It also represents what actions are to be taken on the occurrence of particular events.

16. Define Data Dictionary.

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.

17. What are the dimensions of requirements gathering?

The dimensions of requirements gathering are:

- Responsibilities: Commitments on either side Requirement form the basis for the success of further in a project.
- Current system requirements
    - ➢ Functionality requirements
    - ➢ Performance requirements
    - ➢ Availability needs
    - ➢ Security
    - ➢ Environmental definitions
- Targets
- Acceptance criteria
- Ongoing needs:
    - ➢ Documentation
    - ➢ Training

- Ongoing support

18. List the skill sets required during the requirements phase.

The skill sets required during the requirements phase are:

Availability to look the requirements	Technology awareness
domain expertise	Strong negotiation skills
Storing interpersonal skills	Strong communication skills
Ability to tolerate ambiguity	

19. What are the dimensions of requirements gathering?

The dimensions of requirements gathering are:

Responsibilities	Current system needs
Targets	Ongoing needs

20. Give the classifications of system requirements.

The classification of system requirements are:

Functionality Requirements	Performance requirements
Availability needs	Security
Environmental definition	

21. What does P-CMM model stand for?

P-CMM stand for people CMM.

22. What are the components of the Cost of Quality?

Components of the Cost of Quality are:

Quality Costs.	Prevention Costs.

Appraisal Costs.

23. What is the use of CMM?

Software Quality means Conformance to explicitly stated functional and performance requirements, explicitly documented development standards, Implicit characteristics expected for professionally developed software.

24. Distinguish between verification and validation.

Verification refers to the set of activities that ensure that software correctly implements a specific function.

Validation refers to a different set of activities that ensure that the software that has been built is traceable to the customer requirements. According to Boehm,

- Verification: Are we building the product right?
- Validation: Are we building the right product?

25. What are the conditions that exist after performing validation testing?

After performing the validation testing their exists two conditions:

- The function or performance characteristics are according to the specifications and are accepted.
- The requirement specifications are derived and the deficiency list is created. The deficiencies then can be resolved by establishing the proper communication with the customer.

26. Explain the Capability Maturity Model.

CMM model strives to achieve predictability and consistency as a precursor to continuous improvements by following a set of process in a well defined framework.

- Level 1 is Initial Level

- Level 2 is repeatable which helps in achieving repeatability of performance and quality should the organizations undertake a similar project again.
- Level 3 is defined level.
- Level four is measured level.
- Level 5 is optimistic level, here people always work towards a target.

27. Name any two process models.

- The ISO-9001 Model.
- The capability maturity model.

PART B

1. What is the need of feasibility study? Explain various types of feasibility study in brief. (8)

2. Short notes on

a. Feasibility study

b. Types of feasibility study

c. Data Dictionary

3. What is requirement engineering? Discuss different types of requirement elicitation techniques in brief. (10)

4. What is software requirement specification (SRS)? State its principles and characteristics. (10)

5. What is the difference between SRS document and design document? What are the contents we should contain in the SRS document and design document

UNIT – III

SOFTWARE DESIGN

PART A

1. **Define design process.**

   Design process is a sequence of steps carried through which the requirements are translated into a system or software model.

2. **List the principles of a software design.**

   i. The design process should not suffer from “tunnel vision”.

   ii. The design should be traceable to the analysis model.

   iii. The design should exhibit uniformity and integration.

   iv. Design is not coding.

   v. The design should not reinvent the wheel.

3. **List out the characteristics of good software design**

   ➢ The design must implement all of the explicit requirements contained in the analysis model, and it must accommodate all of the implicit requirements desired by the customer.

   ➢ The design must be a readable, understandable guide for those who generate code and for those who test and subsequently support the software.

   ➢ The design should provide a complete picture of the software, addressing the data, functional, and behavioral domains from an implementation perspective.

4. **List out the Fundamental software design concepts**

   - Abstraction
   - Refinement
   - Modularity
   - Software architecture
   - Control hierarchy
   - Structural portioning
   - Data structure
   - Software procedure

- Information hiding

5. **Explain abstraction with respect to software design concepts. List the types**

   Abstraction permits one to concentrate on a problem at some level of generalization without regard to irrelevant low level details; use of abstraction also permits one to work with concepts and terms that are familiar in the problem environment without having to transform them to an unfamiliar structure

   - Procedural Abstraction
   - Data Abstraction

6. **What is refinement in design, according to Wirth?**

   In each step of refinement, one or several instructions of the given program are decomposed into more detailed instructions. This successive decomposition terminates when all instructions are expressed in terms of any programming

7. **Define software architecture**.

   Software architecture alludes to "overall structure of the software and the ways in which the structure provides conceptual integrity for a system". Architecture is the hierarchical structure of program components, the manner in which the components interact and structure of data that are used by these components.

8. **What is called factoring or vertical partitioning?**

   Factoring suggests that control and work should be distributed top-down in the program structure. Top level modules should perform control functions. Low level modules include workers, performing all input, computation and output tasks.

9. **What is information hiding?**

   The principle of information hiding suggests that modules be characterized by design a decision that hides from all others. The modules should be specified and designed so that information within a module is inaccessible to other modules that have no need for such information.

10. **What is cohesion? List out the various types.**

A cohesive module performs a single task within a s/w procedure, requiring little interactions with procedures being performed in other parts of a program. The ypes of cohesion include:

- Coincidental cohesion.
- Logical cohesion.
- Temporal cohesion.
- Communication cohesion.
- Sequential cohesion.
- Functional cohesion.
- Informational cohesion.

11. What are the elements of design model

The elements of design model are;

- Data Design
- Architectural design
- Interface design
- Component-level design.

12. What is the benefit of modular design?

Changes made during testing and maintenance becomes manageable and they do not affect other modules.

13. What is a cohesive module?

A cohesive module performs only "one task" in software procedure with little interaction with other modules. In other words cohesive module performs only one thing.

14. What are the different types of Cohesion?

- Coincidentally cohesive –The modules in which the set of tasks are related with each other loosely then such modules are called coincidentally cohesive.
- Logically cohesive – A module that performs the tasks that are logically related with each other is called logically cohesive.

- Temporal cohesion – The module in which the tasks need to be executed in some specific time span is called temporal cohesive.
- Procedural cohesion – When processing elements of a module are related with one another and must be executed in some specific order then such module is called procedural cohesive.
- Communicational cohesion – When the processing elements of a module share the data then such module is called communicational cohesive.

15. What is Coupling?

Coupling is the measure of interconnection among modules in a program structure. It depends on the interface complexity between modules.

16. What are the various types of coupling?

- Data coupling – The data coupling is possible by parameter passing or data interaction.
- Control coupling – The modules share related control data in control coupling.
- Common coupling – The common data or a global data is shared among modules.
- Content coupling – Content coupling occurs when one module makes use of data or control information maintained in another module.

17. What are the common activities in design process?

- System structuring – The system is subdivided into principle subsystems components and communications between these subsystems are identified.
- Control modeling – A model of control relationships between different parts of the system is established.
- Modular decomposition – The identified subsystems are decomposed into modules.

18. What are the various elements of data design?

- Data object – The data objects are identified and relationship among various data objects can be represented using ERD or data dictionaries.
- Databases – Using software design model, the data models are translated into data structures and data bases at the application level.

- Data warehouses – At the business level useful information is identified from various databases and the data warehouses are created.

19. List the guidelines for data design.

- Apply systematic analysis on data.
- Identify data structures and related operations.
- Establish data dictionary.
- Use information hiding in the design of data structure.
- Apply a library of useful data structures and operations.

20. What is an architectural style?

Architectural style describes system category that encompasses a set of components that performs a function required by a system. Set of connectors that enable communication, coordination among components

PART- B

1. What are the advantages of modular design? State the design heuristics for effective modularity. (10)
2. What are the design issues to be considered in a User Interface design? (6)
3. What are the characteristics of good design? Describe the different types of coupling and cohesion. How is design evaluation performed? (3 + 4 + 3)
4. State the effects of coupling and cohesion in software quality. (6)
5. Why is UID critical for highly interactive software?
6. What is the difference between Level-0 and Level-1 DFD? Draw a Level-0 and Level-1 DFD for safe Home Security System. (8)
7. Write short notes on

a. User interface design

b. Architectural design

8. Using the architecture of college building as metaphor draw comparisons with software architecture. How the disciplines of classical architecture and software architecture are are similar? How do they differ? Explain.
9. What are the major design issues in User Interface Design? Explain with neat ketches wherever necessary.

UNIT-IV

TESTING AND IMPLEMENTATION

PART-A

1. **Define software testing.**

   Software testing is generally described as a group of procedures carried out to evaluate some aspect of a piece of software.

   Software testing can be described as a process used for revealing defects in software and for establishing that the software has attained a specified degree of quality with respect to selected attributes.

2. **2. List out the difference between verification and validation.**

Verification	Validation
Verification is the process of evaluating a software System or component to determine whether the products of a given development phase satisfy the conditions imposed at the start of that phase	Validation is the process of evaluating a software system Or component during or at the end of the development Cycle in order to determine whether it satisfies Specified requirements
It refers to the set of activities that ensure that software correctly implements a specific function.	It refers to a different set of activities that ensure that the software that has been built is traceable to customers requirements.
Verification is usually associated with activities such as Inspections and reviews of software deliverables.	Validation is usually associated with traditional execution based testing that is exercising the code with test cases.

3. **List out the two basic testing strategies.**

   The two basic testing strategies are

   Black box approach – A tester considers the software under test to be an opaque box and there is no knowledge of its inner structure .

White box approach – It focuses on the inner structure of the software to be tested. To design the test cases using this strategy the tester must have knowledge of that structure

4. **What are the advantages of equivalence class partitioning?**

The advantages of equivalence class partitioning are,

- It eliminates the need for exhaustive testing, which is not feasible.
- It guides a tester in selecting a subset of test inputs with a high probability of detecting a defect
- It allows a tester to cover a larger domain of inputs/outputs with a smaller subset selected from an equivalence class.

5. **Difference between the white box and black box testing strategies**

White box testing	**Black box testing**
It focuses on the inner structure of the software to be Tested.	There is no knowledge of its inner structure
To design test cases using this strategy the tester must have Knowledge of that structure	The tester only has knowledge of what it does
The tester selects test cases to exercise specific internal Structural elements to determine if they are working Properly	The tester provides the specified inputs to the software Under test, runs the test and then determines if the outputs Produced are equivalent to those in the specification
The methods used in the white box testing are statement Testing, branch testing, path testing, data flow testing, Mutation testing and loop testing	The methods used in the black testing are Equivalence Class, partitioning, boundary value analysis, state Transition testing, cause and effect graphing and error Guessing

6. What is unit testing?

Unit testing focuses verification effort on the smallest unit of software design—the software component or module. It is a white box approach.

7. What are the things to be considered while performing unit test

- The module interface is tested to ensure that information properly flows into and out of the program unit under test.
- The local data structure is examined to ensure that data stored temporarily maintains its integrity during all steps in an algorithm's execution.
- Boundary conditions are tested to ensure that the module operates properly at boundaries established to limit or restrict processing.
- All independent paths (basis paths) through the control structure are exercised to ensure that all statements in a module have been executed at least once.

8. What is integration testing

Integration testing is a systematic technique for constructing the program structure while at the same time conducting tests to uncover errors associated with interfacing. The objective is to take unit tested components and build a program structure that has been dictated by design.

9. Write the difference between driver and stub?

- Stubs and drivers are dummy programs written while integration testing.
- Stubs are used during top down testing. In this type highest level components are created first. To test if component written will function correctly when integrated with lower level components a dummy program for lower level component is written as a substitute of actual code which is called Stubs.
- Stubs will contain only functionality needed to be successfully called by a higher level component. It will simulate the behavior of a lower level

component. In bottom up approach, lower level components are created first.

- Temporary components called 'drivers' are written as substitutes for the missing code. Then the lowest level components can be tested using the test driver. For eg. a empty block of code (driver) can be created which only calls a function.

10. Write the difference between top-down integration and bottom up integration

Top-down integration testing is an incremental approach to construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main control module (main program). Modules subordinate (and ultimately subordinate) to the main control module are incorporated into the structure in either a depth-first or breadth-first manner.

Bottom-up integration testing, as its name implies, begins construction and testing with atomic modules (i.e., components at the lowest levels in the program structure). Because components are integrated from the bottom up, processing required for components subordinate to a given level is always available and the need for stubs is eliminated.

11. What is Regression Testing?

It is not a level of testing, but it is the retesting of software that occurs when changes are made to ensure that the new version of the software has retained the capabilities of the old version and that no new defects have been introduced due to the changes.

12. What is Recovery Testing?

It subjects a system to losses of resources in order to determine if it can recover properly from these losses. This type of testing is especially important for transaction system.

13. Define alpha testing.

This test takes place at the developer‘s site. A cross section of potential users and members of the developer’s organization are invited to use the software. Developers observe the users and note problems.

14. Define acceptance testing.

When software is being developed for a specific client, acceptance tests are carried out after system testing, the test cases are based on requirements. Acceptance tests are a very important milestone for the developers.

15. Define security testing

Security testing attempts to verify that protection mechanisms built into a system will, in fact, protect it from improper penetration. "The system's security must, of course, be tested for invulnerability from frontal attack—but must also be tested for invulnerability from flank or rear attack

16. Define stress testing

Stress testing executes a system in a manner that demands resources in abnormal quantity, frequency, or volume. For example,

(1) special tests may be designed that generate ten interrupts per second, when one or two is the average rate,

(2) input data rates may be increased by an order of magnitude to determine how input functions will respond,

(3) test cases that require maximum memory or other resources are executed,

(4) test cases that may cause thrashing in a virtual operating system are designed,

(5) test cases that may cause excessive hunting for disk-resident data are created. Essentially, the tester attempts to break the program.

A variation of stress testing is a technique called sensitivity testing.

17. Define performance testing

Performance tests are often coupled with stress testing and usually require both hardware and software instrumentation. That is, it is often necessary to measure resource utilization (e.g., processor cycles) in an exacting fashion. External instrumentation can monitor execution intervals, log events (e.g., interrupts) as they occur, and sample machine states on a regular basis. By instrumenting a system, the tester can uncover situations that lead to degradation and possible system failure.

18. Define use case.

A use case is a pattern, scenario or exemplar or usage. It describes a typical interaction between the software system under development and a user.

19. Define a path.

A path is a sequence of control flow nodes usually beginning from the entry node of a graph through to the exit node.

20. Define cyclomatic complexity.

Cyclomatic complexity is a measure of the number of so called independent paths in the graph. Independent path is a special kind of path in the flow graph.

$V(G)=E-N+2$

E- no of edges in a flow graph

N- no of nodes in a flow graph

21. Explain the process of debugging

Debugging is not testing but always occurs as a consequence of testing. The debugging process begins with the execution of a test case. Results are assessed and a lack of correspondence between expected and actual performance is encountered.

PART-B

1. Write short notes on

(i)StructuralTesting(4)

(ii)System Testingariddebugging(4)

(iii)IntegrationTesting(4)

(iv)BlackboxTesting. (4)

2. (i) Enumerate the various types of software test. Which type of testing is suitable for boundary condition? Justify. (2 + 6)

   (ii) How do you relate software testing results with reliability of the product? Explain. (8)

3. What is meant by control flow testing? ‘‘Is it always falling with data flow testing in case of software’’? Justify. (2 + 6)

4. What is the specific purpose of Black-box testing? Explain clearly any two Black-box testing methods?

5. Write short notes on the following :

   (i) Cyclomatic complexity

   (ii) Loop testing

   (iii) Basis path testing

   (iv) Acceptance testing

6. What are all the formulas for cyclomatic complexity? Calculate cyclomatic Complexity for greatest of three numbers. (8)

   (ii) How would you derive test cases for the given project? Explain in detail. (8)

7. (i) Narrate the path testing procedure in detail with a sample code.(8)

   (ii) Explain the different integration testing approaches. (8)

8. Is it beneficial to allow users to test the software before finally accepting it? If yes why? Explain the testing through which the users test the software. What are the various levels of testing that could be performed for a particular software

UNIT V

PROJECT MANAGEMENT

PART-A

1. What is make / buy decision? When this situation arise.

Software engineering managers are faced with a make/buy decision that can be further complicated by a number of acquisition options:

(1) software may be purchased (or licensed) off-the-shelf,

(2) “fullexperience” or “partial-experience” software components may
be acquired and then modified and integrated to meet specific needs, or

(3) software may be custom built by an outside contractor to meet the purchaser's

2. What is out sourcing?

Out sourcing is extremely simple. Software engineering activities are contracted to a third party who does the work at lower cost and, hopefully, higher quality.

3. Define Risk identification

Risk identification is a systematic attempt to specify threats to the project plan (estimates, schedule, resource loading, etc.). By identifying known and predictable risks, the project manager takes a first step toward avoiding them when possible and controlling them when necessary.

There are two distinct types of risks are

- generic risks
- product-specific risks.

4. What is RMMM?

Risk Mitigation, Monitoring and Management Plan. The RMMM plan documents all work performed as part of risk analysis and is used by the project manager as part of the overall project plan.

5. Explain Task set

A task set is a collection of software engineering work tasks, milestones, and deliverables that must be accomplished to complete a particular project. The task set to be chosen must provide enough discipline to achieve high software quality. But, at the same time, it must not burden the project team with unnecessary work.

6. Write the objective of software risk management

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, but because not all risks can be avoided, the team works to develop a contingency plan that will enable it to respond in a controlled and effective

7. Write the characteristics of software risk.

The 2 characteristics are

Uncertainty—the risk may or may not happen; that is, there are no 100% probable risks

Loss—if the risk becomes a reality, unwanted consequences or losses will occur.

8. List out the consequences of project risks

The project schedule will slip and that costs will increase. Project risks identify potential budgetary, schedule, personnel (staffing and organization), resource, customer, and requirements problems and their impact on a software project

9. Write the impacts of technical risks

Technical risks threaten the quality and timeliness of the software to be produced.

If a technical risk becomes a reality, implementation may become difficult or impossible. Technical risks identify potential design, implementation, interface, verification, and maintenance problems

10. List and describe the risk components

- 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.

11. What is risk projection? Write the 4 risk projection activities

Risk projection, also called risk estimation, attempts to rate each risk in two ways—the likelihood or probability that the risk is real and the consequences of the problems associated with the risk.

The project planner, along with other managers and technical staff, performs four risk projection activities:

(1) establish a scale that reflects the perceived likelihood of a risk,

(2) delineate the consequences of the risk,

(3) estimate the impact of the risk on the project and the product, and

(4) note the overall accuracy of the risk projection so that there will be no misunderstandings.

12. List out the root causes for late delivery of software

An unrealistic deadline established by someone outside the software development -group and forced on managers and practitioner's within the group.

• Changing customer requirements that are not reflected in schedule changes.

• An honest underestimate of the amount of effort and/or the number of resources that will be required to do the job.

• Predictable and/or unpredictable risks that were not considered when the project commenced.

• Technical difficulties that could not have been foreseen in advance.

• Human difficulties that could not have been foreseen in advance.

• Miscommunication among project staff that results in delays.

• A failure by project management to recognize that the project is falling behind schedule and a lack of action to correct the problem.

13. ”If we fall behind schedule, we can always add more programmers and catch up later in the project." Is the myth correct, if not justify

No its not correct, adding people late in a project often has a disruptive effect on the project, causing schedules to slip even further. The people who are added must learn the system, and the people who teach them are the same people who were doing the work. While teaching, no work is done, and the project falls further behind.

14. **Explain the software Process metric indicators**

Process indicators enable a software engineering organization to gain insight into the efficacy of an existing process (i.e., the paradigm, software engineering tasks, work products, a milestones). They enable managers and practitioners to assess what works and what doesn't. Process metrics are collected across all projects and over long periods of time. Their intent is to provide indicators that lead to long-term software process improvement.

15. **Explain the software Project metric indicators**

Project indicators enable a software project manager to

(1) assess the status of an ongoing project,

(2) track potential risks,

(3) uncover problem areas before they go "critical,"

(4) adjust work flow or tasks, and

(5) evaluate the project team's ability to control quality of software work products.

16. **List the basic principles that guide software project scheduling**

Compartmentalization.

Interdependency.

Time allocation

Effort validation

Defined responsibilities

Defined outcomes

Defined milestones

17. **What are task set? What are the different project types encountered by a software organization**

Task sets are designed to accommodate different types of projects and different degrees of rigor. Although it is difficult to develop a comprehensive taxonomy of software project types, most software organizations encounter the following projects:

1. Concept development projects that are initiated to explore some new business concept or application of some new technology.
2. New application development projects that are undertaken as a consequence of a specific customer request.
3. Application enhancement projects that occur when existing software undergoes major modifications to function, performance, or interfaces are observable by the end-user.
4. Application maintenance projects that correct, adapt, or extend existing software in ways that may not be immediately obvious to the end-user.
5. Reengineering projects that are undertaken with the intent of rebuilding an existing (legacy) system in whole or in part.

18. What are process indicators? Write its uses

Process indicators enable a software engineering organization to gain insight into the efficacy of an existing process (i.e., the paradigm, software engineering tasks, work products, and milestones). They enable managers and practitioners to assess what works and what doesn't. Process metrics are collected across all projects and over long periods of time. Their intent is to provide indicators that lead to long-term software process improvement.

Project indicators enable a software project manager to

(1) assess the status of an ongoing project,
(2) track potential risks,
(3) uncover problem areas before they go "critical,"
(4) adjust work flow or tasks, and
(5) evaluate the project team's ability to control quality of software work products.

19. What is personal software process?

The personal software process (PSP) is a structured set of process descriptions, measurements, and methods that can help engineers to improve their personal performance. It provides the forms, scripts, and standards that help them estimate and plan their work. It shows them how to define processes and how to measure their quality and productivity. A fundamental PSP principle is that everyone is different and that a method that is effective for one engineer may not be suitable for another. The PSP thus helps engineers to measure and track their own work so they can find the methods that are best for them.

PART B

1. Describe three types of software process metric that may be collected as part of process improvement process. Give example of each type of metric
2. Describe the details of project planning
3. Discuss in detail the risk management process.
4. Explain RMMM. Describe the steps taken to mitigate risk
5. Explain the constructive cost estimation model COCOMO II by addressing the areas addressed by it
6. Explain the basic concepts of project scheduling and tracking. Write the relationship between people and effort
7. Explain defining task set for a software project