Soundarya Educational Trust
Soundarya Institute of Management & Science
Affiliated to Bangalore University || NAAC Accredited Institution || Approved by Govt. of Karnataka

Department of Computer Science

BCA 502T SOFTWARE ENGINEERING

BCA502T : SOFTWARE ENGINEERING

Total Teaching Hours : 60 — No of Hours / Week : 04

Unit - I
Introduction: Software Products and Software process, Process models: Waterfall modal, Evolutionary Development, Bohemia's Spiral model, Overview of risk management, Process Visibility, Professional responsibility. Computer based System Engineering: Systems and their environment, System Procurement, System Engineering Process, System architecture modelling. Human Factors, System reliability Engineering. Requirements and Specification: The requirement Engineering Process, The Software requirement document, Validation of Evolution of requirements, Viewpoint – oriented & method based analysis , system contexts , Social 7 organizational factors . Data flow , Semantic, Objects, models , Requirement Specification, Non functional requirement.[12 Hours]

Unit - II
Software Prototyping: Prototyping in software process, Prototyping techniques, User interface prototyping. Software Design: Design Process, Design Strategies, Design Quality , System Structuring control models, Modular decomposition , Domain Specific architecture.[12 Hours]

Unit - III
Object Oriented& function oriented design: Objects, object Classes and inheritance Object identification, An object oriented design example, Concurrent Objects, Data flow design Structural decomposition, Detailed Design, A Comparison of design Strategies.User interface design: Design Principles, User System interaction, Information Presentation, User Guidance, Interface Evaluation. [12 Hours]

Unit - IV
Software Reliability and reusability : Software reliability metrics , Software reliability Specification , Statistical testing ,Reliability Growth modeling, Fault avoidance & tolerance, Exception handling & defensive programming , Software development with reuse, Software' development for reuse , Generator based reuse, Application System Portability.[12 Hours]

Unit - V
Software Verification and Validation : The testing Process , Test Planning & Strategies, Black Box , Structural, interface testing , Program inspections , Mathematically based verification, Static analysis tools, Clean room software development. Management Issues: Project management, Quality management, Software cost estimation, Softwaremaintenance. [12 Hours]

Text book

1. Ian Sommerville – Software Engineering, 9th Edition, Pearson Education Ltd, 2010.

Reference Books

1. Roger S. Pressman – Software Engineering, A Practitioner's approach, 7th Edition, McGRAW-HILL Publication, 2010.
2. Pankaj Jalote, "An integrated approach to Software Engineering", 3rd Edition, Narosa Publishing House, 2013.

BCA502T : SOFTWARE ENGINEERING

BLUE PRINT

Question paper pattern for theory has two sections :

Section – A :Contains 12 questions, out of which a student has to answer 10 questions. Each question carries 2 marks (10 x 2 = 20)

Section – B :Contains 8 full questions, out of which 5 question to be answered. Each full question carries 5 marks (5 x 5 = 25)

Section – C :Contains 5 full questions with sub questions(a) & (b), out of which 3 questions to be answered. Each full question carries 15 marks (3x 15 = 45)

Section – D :Contains 2 full questions, out of which 1 question to be answered. Each full question carries 10 marks (1 x 10 = 10)

UNIT	CHAPTER	SECTION A 2 MARKS	SECTION B 5 MARKS	SECTION B 15 MARKS	SECTION D 15 MARKS
I	Introduction: Software Products and Software process,	4	2	1	1
II	Software Prototyping	1	1	1	
III	Object Oriented& function oriented design	2	2	1	
IV	Software Reliability and reusability	2	1	1	
V	Software Verification and Validation	3	2	1	1
	TOTAL	12	8	5	2
		ANSWER ANY 10	ANSWER ANY 5	ANSWER ANY 3	ANSWER ANY 1
	TOTAL MARKS	**20**	**25**	**45**	**10**

SECTION – A (2 Marks)

UNIT-I

[Nov / Dec 2015]

1. What is software product ? Name two types of software product.
2. What is the difference between software engineering and system engineering?
3. What is system decommissioning ?
4. Define volatile requirement.
5. What are functional requirement? Give example.

[Nov / Dec 2016]

6. What is customized software product? Give an example.
7. What is COTS?
8. What is feasibility study?

[Nov / Dec 2017]

9. Define system.
10. What are the two types of software products?
11. Define SRS

[Nov / Dec 2018]

12. Define Software engineering?

[TMAQ - Important Tutor Mark Assignment Questions]

13. Define SDLC. List the different phases of SDLC.
14. What are the goals of software engineering?
15. What are the characteristics of software process?
16. Differentiate between generic product & customized product.
17. What are the characteristics of software?
18. Differentiate between product & process in software engineering.
19. Name different types of SDLC models.
20. What is system integration ? Explain the two types of system integration.
21. What are non-functional requirement? Give example.
22. Define DFD. Explain different symbols.

SECTION – A (2 Marks)

UNIT-II

[Nov / Dec 2015]

1. Define cohesion and coupling.
2. Define prototype

[Nov / Dec 2016]

Define coupling
3. Define object class.

[Nov / Dec 2017]

Define cohesion.
4. Define Evolutionary prototyping.

[Nov / Dec 2018]

What is coupling? Mention any two types.

[TMAQ - Important Tutor Mark Assignment Questions]

5. What is loose coupling?
6. What is tight coupling?
7. Difference between generic model and reference model.
8. What is software design?
9. What are the objectives of prototyping?
10. Why software design is important?
11. Define throw-away prototyping.

SECTION – A (2 Marks)

UNIT-III

[Nov / Dec 2015]

1. Write any two characteristics of GUI.
2. What is object oriented design.

[Nov / Dec 2016]

3. What are OOD and OOP?
4. What is user interface design?

[Nov / Dec 2017]

5. Define object and class.
What are the characteristics of GUI.

[Nov / Dec 2018]

What are OOD and OOP?
6. What are the advantages of GUI.

[TMAQ - Important Tutor Mark Assignment Questions]

7. Name the different styles of user-system interaction.
8. What are advantages and disadvantages of command languages?
9. Explain two types of interface evaluation.
10. Define object.
11. Define object class.

SECTION – A (2 Marks)

UNIT-IV

[Nov / Dec 2015]

1. Define reliability. Mention its types.

[Nov / Dec 2016]

2. Difference between fault and failure.

[Nov / Dec 2017]

3. Define reliability.

[Nov / Dec 2017]

4. What is fault detection and recovery?
5. Define risk.

[TMAQ - Important Tutor Mark Assignment Questions]

6. What is software product line?
7. Define software reliability.
8. Define hardware reliability.
9. Define software reliability metrics.
10. What is RGM?
11. What are the factors on which reliability depends?

SECTION – A (2 Marks)

UNIT-V

[Nov / Dec 2015]

1. What is test case ? Give one example for test case.
2. What is quality assurance? What is the purpose of quality assurance?
3. List different phases of project management.

[Nov / Dec 2016]

4. What do you mean by cyclomatic complexity?
5. Define quality planning.
6. What is software maintenance?

[Nov / Dec 2017]

7. Differentiate between verification and validation/
8. What is test case?

[Nov / Dec 2018]

9. Define equivalence class portioning.
Define quality assurance.
10. Define project management
Differentiate between verification and validation.

[TMAQ - Important Tutor Mark Assignment Questions]

1. Define Black box testing.
2. What are the advantages and disadvantages of V-model.
3. Define quality assurance.
4. Explain the importance of standards.

SECTION – B(5 Marks)

UNIT-I

[Nov / Dec 2015]

1. Discuss the challenges of software engineer.
2. Explain system procurement process in detail.

[Nov / Dec 2016]

Discuss the challenges of software engineer.
3. Explain the phases of requirement elicitation and analysis process.

[Nov / Dec 2017]

4. Explain water fall model with its advantages and disadvantages.
5. What are volatile requirements? Explain the classification of volatile requirements.

[Nov / Dec 2018]

6. Describe system procurement process.
7. Explain the IEEE structure of SRS document.

[TMAQ - Important Tutor Mark Assignment Questions]

1. Explain spiral model with advantages and disadvantages.
2. What is data flow diagram? Explain various notations using in data flow model with an example.
3. Explain different requirement validation check.
4. What is SDLC ? Explain the different phases of SDLC with neat diagram.
5. Describe system architectural modeling with an example.
6. What is software process? What are the activities of software process?

SECTION – B(5 Marks)

UNIT-II

[Nov / Dec 2015]

1. Explain Prototyping model.

[Nov / Dec 2016]

2. Describe in detail the design principle

[Nov / Dec 2017]

3. Explain two types of prototyping with advantages and disadvantages

[Nov / Dec 2018]

4. Describe design principles

[TMAQ - Important Tutor Mark Assignment Questions]

SECTION – B(5 Marks)

UNIT-III

[Nov / Dec 2015]

1. Describe any two styles of use interaction

[Nov / Dec 2016]

2. What are the methods of object identification with an example

[Nov / Dec 2017]

3. Explain different phases of user system interaction

[Nov / Dec 2018]

4. Write a short note on Data flow design, structural decomposition.

[TMAQ - Important Tutor Mark Assignment Questions]

SECTION – B(5 Marks)

UNIT-IV

[Nov / Dec 2015]

1. Explain different types of software reliability metrics.
2. What are the different types of interface errors

[Nov / Dec 2016]

3. Write a note on system reliability engineering.
4. Discuss hardware and software reliability metrics.

[Nov / Dec 2017]

5. Give the classification of failures with examples
6. What is fault tolerance? Explain the two approaches of software fault tolerance.

[Nov / Dec 2018]

7. Write a short note on reliability growth modeling.

SECTION – B(5 Marks)

UNIT-V

[Nov / Dec 2015]

1. Write a note on black box testing

[Nov / Dec 2016]

2. Explain thread testing with a diagram

[Nov / Dec 2017]

3. Difference between Black box and white box testing

[Nov / Dec 2018]

4. Explain the content of test plan

SECTION –C , D & E

UNIT-I

[Nov / Dec 2015]

1. Explain Spiral model with neat diagram. Discuss advantages and disadvantages.
2. Explain requirement elicitation and analysis process.
3. Explain IEEE structure of SRS
4. Write SRS for library system.

[Nov / Dec 2016]

5. Explain water fall model with a neat diagram. Mention its merits and demerits

[Nov / Dec 2017]

6. Explain system engineering process with a neat diagram.

[Nov / Dec 2018]

7. Explain the fundamental process activities involved in SDLC with neat diagram.

UNIT-IV & UNIT V

[Nov / Dec 2015]

1. Explain quality control in detail.

[Nov / Dec 2016]

2. Write a note on Risk management
3. Write a note on COCOMO model.

[Nov / Dec 2017]

4. Explain different types of software maintenance.
5. Explain the contents of test plan template.

[Nov / Dec 2018]

1. Explain clean room software development process.
2. Explain software reuse.
3. Explain types of cohesion.
4. Explain function oriented design.

UN – 322

V Semester B.C.A. Degree Examination, Nov./Dec. 2015
(Y2K8 Scheme) (F + R)
BCA 501 : SOFTWARE ENGINEERING
(100 – 2013-14 and Onwards) (90 – Prior to 2013-14)

Time : 3 Hours — Max. Marks : 90/100

Instructions : Section – A, B, C is common to all. Section – D is applicable to the students who have admission in 100 marks.

SECTION – A

Answer **any ten** questions. **Each** question carries **2** marks. **(10×2=20)**

1. What is software product ? Name two types of software product.
2. What is the difference between software engineering and system engineering ?
3. What is system decommissioning ?
4. What are functional requirements ? Give one example.
5. Define cohesion and coupling.
6. What is test case ? Give one example for test case.
7. Define volatile requirement.
8. List different phases of project management.
9. What is quality assurance ? What is the purpose of quality assurance ?
10. Define reliability. Mention its types.
11. Write any two characteristics of GUI.
12. What is fault detection and recovery ?

SECTION – B

Answer **any five** questions. **Each** question carries **5** marks. **(5×5=25)**

13. Discuss the challenges of software engineer.
14. Explain system procurement process in detail.

P.T.O.

UN – 322

15. Explain prototyping model.
16. Describe any two styles of user system interaction.
17. What is risk identification ? Explain its techniques.
18. Write a short note on black box testing.
19. Explain different types of interface errors.
20. Explain different types of software reliability metrics.

SECTION – C

Answer **any 3** questions : **(3×15=45)**

21. Explain spiral model with neat diagram. Discuss advantages and disadvantages. 15
22. a) Explain requirement elicitation and analysis process. 8
b) Discuss object oriented design process in detail. 7
23. a) Explain IEEE structure of SRS. 10
b) Write SRS for library system. 5
24. a) Explain the contents of test plan. 8
b) Explain different levels of testing. 7
25. a) Explain quality control in detail. 8
b) Write a short note on software productivity. 7

SECTION – D

Answer **any 1** question. **Each** question carries **ten** marks. **(1×10=10)**

26. Explain the fundamental process activities involved in SDLC with neat diagram. 10
27. Write a short note on :
a) Context model. 5
b) COCOMO model. 5

NS – 612

V Semester B.C.A. Degree Examination, Nov./Dec. 2016
(CBCS) (2016-17 and Onwards)
COMPUTER SCIENCE
BCA 502 : Software Engineering

Time : 3 Hours — Max. Marks : 100

***Instruction :** Answer **all** Sections.*

SECTION – A

Answer **any ten** questions. **Each** question carries **two** marks : **(10×2=20)**

1. What is customized software product ? Give an example.
2. What is COTS ?
3. What is feasibility study ?
4. What is 4GL ?
5. Define coupling.
6. What are OOD and OOP ?
7. What is user interface prototyping ?
8. Difference between fault and failure.
9. What do you mean by cyclomatic complexity ?
10. What is interface testing ?
11. Define quality planning.
12. What is software maintenance ?

SECTION – B

Answer **any five** questions. **Each** question carries **five** marks : **(5×5=25)**

13. Discuss the challenges of software engineer.
14. Write a note on system realibility engineering.
15. Explain the phases of requirement elicitation and analysis process.

P.T.O.

NS – 612

16. Explain the methods for object identification.
17. Write a short note on user interface design.
18. Explain reliability growth modeling with its advantages.
19. Explain thread testing with a diagram.
20. Explain quality assurance in brief.

SECTION – C

Answer **any three** questions. **Each** question carries **fifteen** marks : (3x15=45)

21. Explain spiral model with a neat diagram. Discuss its advantages and disadvantages. 15

22. a) Explain various requirement validation techniques. 9
    b) Explain evolutionary prototyping with a diagram. 6

23. a) Explain different types of cohesion with example. 9
    b) Explain functional oriented design with example. 6

24. a) Describe the five types of user system interaction. 8
    b) Explain four types of software realibility matrices. 7

25. a) Explain any two types of software testing. 8
    b) Explain quality control in brief. 7

SECTION – D

Answer **any one** question. **Each** question carries **ten** marks : (1x10=10)

26. Explain waterfall model with a neat diagram. Mention its merits and demerits. 10

27. Write short note on :
    a) Risk Management 5
    b) COCOMO model. 5

SN – 663

V Semester B.C.A. Degree Examination, Nov./Dec. 2017
(2016-17 and Onwards) (CBCS) (F + R)
BCA 502 : SOFTWARE ENGINEERING

Time : 3 Hours Max. Marks : 100

Instruction : *Answer all Sections.*

SECTION – A

I. Answer **any ten** questions. **Each** question carries **two** marks. (10x2=20)

1) Define system.
2) What are the two types of software products ?
3) What is system decommissioning ?
4) Mention two advantages of prototype model.
5) Define cohesion.
6) Define object and class.
7) What are the characteristics of GUI ?
8) Define SRS.
9) Define Risk.
10) Differentiate between verification and validation.
11) Define reliability.
12) What is a test case ?

SECTION – B

II. Answer **any five** questions. **Each** carries **five** marks. (5x5=25)

13) Explain waterfall model with its advantages and disadvantages.
14) What are volatile requirements ? Explain the classification of volatile requirements.
15) Explain the different phases of system design process with a diagram.
16) What is fault tolerance ? Explain the two approaches to software fault tolerance.
17) Differentiate between black box and white box testing.

P.T.O.

SN – 663

18) Explain the quality characteristics of design.
19) Describe different requirement validation checks.
20) Explain types of software maintenance.

SECTION – C

III. Answer **any three** questions. **Each** question carries **fifteen** marks. (3x15=45)

21) a) Explain requirement elicitation and analysis process of requirement engineering with diagram.
    b) Explain IEEE structure of SRS document. (8+7)
22) a) Explain design principles in detail.
    b) Explain two types of prototyping with advantages and disadvantages. (8+7)
23) a) Explain different reliability metrics.
    b) Explain reliability growth modeling. (7+8)
24) a) Write a note on object oriented design concept.
    b) Explain different styles of user system interaction. (7+8)
25) a) Explain various levels of testing.
    b) Explain the contents of test plan template. (6+9)

SECTION – D

IV. Answer **any one** question. **Each** carries **ten** marks. (1x10=10)

26) Explain COCOMO model in detail.
27) Explain system engineering process with a neat diagram.

SS – 683

V Semester B.C.A. Degree Examination, November/December 2018
(CBCS) (F + R) (2016-17 and Onwards)
COMPUTER SCIENCE
BCA 502 : Software Engineering

Time : 3 Hours Max. Marks : 100

Instruction : *Answer all Sections.*

SECTION – A

I. Answer **any ten** questions. (10x2=20)

1) What is software product ? Name two types of software product.
2) Define system engineering.
3) What is feasibility study ?
4) Define prototype model.
5) What is coupling ? Name two types of coupling.
6) What are OOD and OOP ?
7) What are the advantages of GUI ?
8) Define Test case.
9) Differentiate between verification and validation.
10) Define equivalence class partitioning.
11) Define quality assurance.
12) Define project management.

SECTION – B

II. Answer **any five** questions. (5x5=25)

13) Write a note on risk management.
14) Describe system procurement prooess.
15) Explain the IEEE structure of SRS document.
16) Explain evolutionary and throw-away prototyping.
17) Describe design principles.
18) Write a note on reliability growth modeling.
19) Explain the contents of test plan.
20) Write a note on quality control.

P.T.O.

SS – 683

SECTION – C

III. Answer **any three** questions. (3x15=45)

21) a) Explain the different phases of SDLC.
    b) Explain system design process with a diagram. (8+7)
22) Explain the requirement engineering process. 15
23) a) Explain function oriented design.
    b) Explain different styles of user system interaction. (8+7)
24) a) Explain different types of cohesion.
    b) Explain software reuse. (8+7)
25) a) Describe clean room software development process.
    b) Explain different types of software maintenance. (8+7)

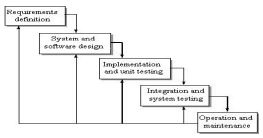
SECTION – D

IV. Answer **any one** question. (1x10=10)

26) Explain spiral model with a neat diagram. Mention its merits and demerits.
27) Explain COCOMO model in detail.

1.Explain waterfall model with its advantages and disadvantages?

Waterfall model was developed by Royce in 1970. this model is also referred as linear sequential model or classical life cycle model. This model suggests a systematic, sequential approach to software development that begins at system level and progresses through analysis, design, development, coding, testing and maintenance.



Advantages of waterfall model:

- **Easy to understand and implement.**
- **Widely used and known**
- **Identifies deliverables and milestones**
- **The waterfall model redeems the software development process of the code and fix problems**
- **It allows for communication between customer and developer and specifies what will be delivered when and at what cost.**

Disadvantages:

- **The waterfall model requires the user to define system requirements early in the project.**
- **This model is rigid because it assumes that a phase is fully complete before another one commences.in reality two or more phases may proceed in parallel.**
- **Interaction with the user takes place right in the beginning while firming up requirements and then at the time of implementation. this leaves a huge gap in-between phases and does not in any way build a method of cross checking user requirements.**

2. Explain IEEE structure of SRS document.

The IEEE standards recognize the fact that different projects may require their requirements to be organized differently, that is no method that is suitable for all projects. It provides different ways of structuring the SRS. The first two sections of SRS are the same in all of them. IEEE suggests the following structure for requirements document.

1.Introduction
- **1.1 Purpose of the requirement's document**
- **1.2 Scope of the product**
- **1.3 Definitions, acronyms and abbreviations**
- **1.4 References to supporting documents**
- **1.5 Overview of rest of SRS**

2.General description
- **2.1 Product perspective**
- **2.2 Product functions**
- **2.3 User characteristics**
- **2.4 General constraints**
- **2.5 Assumptions and dependencies.**

3.Functional requirements
4.Non-functional requirements
5.System architecture
6.System models
7.Appendices

3. Explain the phases of requirement elicitation and analysis process.

The phases of requirement elicitation and analysis include:

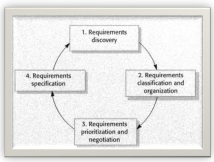
- **Requirements discovery**
- **Requirement classification and organization**
- **Requirement prioritization and negotiation**
- **Requirements specification**

Requirement discovery: This is the process of interacting with stakeholders of the system to discover their requirements. Domain requirements from stakeholders and documentation are also discover during this activity.

Requirements classification and organization:This activity takes the unstructured collection of requirements, groups related requirements, and organizes them into coherent clusters.

Requirements prioritization and negotiation: Inevitably, when multiple stakeholders are involved, requirements will conflict. This activity is concerned with prioritizing requirements and finding and resolving requirements conflicts through negotiation.

Requirements specification:The requirements are documented and input into the next round of the spiral. Formal or informal requirements documents may be produced.



4. Explain prototyping model.

- **The prototyping model is a systems development method in which a prototype is built, tested and then reworked as necessary until an acceptable outcome is achieved from which the complete system or product can be developed.**

Step 1: Requirements gathering and analysis : A prototyping model starts with requirement analysis. In this phase, the requirements of the system are defined in detail. During the process, the users of the system are interviewed to know what is their expectation from the system.

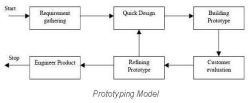
Step 2: Quick design. The second phase is a preliminary design or a quick design. In this stage, a simple However, it is not a complete design. It gives a brief idea of the system to the user. The quick design helps in developing the prototype .design of the system is created.

Step 3:. Building Prototype an actual prototype is designed based on the information gathered from quick design. It is a small working model of the required system.

Step 4: Engineer Product. In this stage, the proposed system is presented to the client for an initial evaluation. It helps to find out the strength and weakness of the working model. Comment and suggestion are collected from the customer and provided to the developer.

Step 5: Refining prototype.If the user is not happy with the current prototype, you need to refine the prototype according to the user's feedback and suggestions.

Step 6: Customer evaluation. Once the final system is developed based on the final prototype, it is thoroughly tested and deployed to production. The system undergoes routine maintenance for minimizing downtime and prevent large-scale failures.



Prototyping Model

5. Explain evolutionary prototyping with an example.

Evolutionary Prototyping –
In this method, the prototype developed initially is incrementally refined on the basis of customer feedback till it finally gets accepted. In comparison to Rapid Throwaway Prototyping, it offers a better approach which saves time as well as effort. This is because developing a prototype from scratch for every iteration of the process can sometimes be very frustrating for the developers.

Advantages –

- **The customers get to see the partial product early in the life cycle. This ensures a greater level of customer satisfaction and comfort.**
- **New requirements can be easily accommodated as there is scope for refinement.**
- **Missing functionalities can be easily figured out.**
- **Flexibility in design.**

Disadvantages –

- **Costly w.r.t time as well as money.**
- **There may be too much variation in requirements each time the prototype is evaluated by the customer.**
- **It is very difficult for the developers to accommodate all the changes demanded by the customer.**
- **Developers in a hurry to build prototypes may end up with sub-optimal solutions.**

6. Explain the principles of software design.

Software design is both a process and a model. The design process is a sequence of steps that enable the designer to describe all aspects of the software to be build.

- **Top-Down or Bottom-up**
  - **Problem Partitioning**
  - **Abstraction**
  - **Modularity**

1. **Problem Partitioning: Problem partitioning is a method of adopting the principle of divide and conquer to get the solution to the problem.**

2. **Abstraction: Abstraction is the method of describing a program function. High level of abstraction states the solution to the problem. Low level abstraction deals with procedural details. Types of abstraction are : Data abstraction, Procedural Abstraction andControl Abstraction**

3. **Modularity: Software is divide into separately named and addressable components, often called modules that are integrated to satisfy problem requirements. Module contains instructions, processing logic and data structures.**

- **Top-Down and Bottom-Up Strategies: Top-Down takes the whole software system as one entity and then decomposes it to achieve more than one sub system or component based on some characteristics until the lowest level of system in the top-down hierarchy is achieved.**

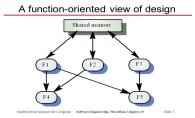
The bottom-up design model starts with most specific and basic components. It keeps creating higher level components until the desired system is not evolved as one single component.

7. Explain function oriented design with example.

A function-oriented design strategy relies on decomposing the system into a set of interacting functions with a centralized system state shared by these functions.

General Procedure :

1. **Start with a high level description of what the software / program does. Refine each part of the description one by one by specifying in greater details the functionality of each part. These points lead to Top-Down Structure.**
2. **Problem in Top-down design method**
3. **Solution to the problem**



The various activities in a function oriented design process are :

- **(b) Data-flow design : This shows how data passes through the system and is transformed by each system functions.**
- **(c) Structural decomposition : Model illustrates how functions are decomposed into sub-functions using graphical structure charts.**
- **(d) Detailed data description : Describe the entities in the design and their interfaces. These descriptions may be recorded in a data dictionary.**

8. Explain different types of cohesion with example.

Cohesion measures the semantic strength of relationship between components within a functional unit.

Several levels of cohesion can be identified in necessary order of strength, these are :

- **(a) Coincidental cohesion : The parts of a component are not related but simply bundled into a single component.**
  Ex : Module miscellaneous functions such as : use customer record, display customer record, read transaction record.
- **(b) Logical cohesion : Components that perform similar functions such as input, error handling and so on are put together in a single document.**
  Ex : if record-type is student then
  display student record
  else if record-type is staff then
  display staff record
- **(c) Temporal cohesion : A temporally cohesive module is one whose elements are functions that are related in time.**
  Ex :Set counter to 0, Open student file, Clear error message variable, initialize array.
- **(d) Communicational Cohesion : All of the elements of a component operate on the same input data or produce the same output data.**
  Ex : Use customer account, find customer name, find customer loan balance etc.
- **(e) Procedural cohesion : A procedurally cohesive module is on whose elements are involved m different activities but the activities are sequential.**
  Ex : Use out record, write out record, read in record, pad numeric fields to zero.
- **(f) Functional cohesion :Each part of the component is necessary for the execution of a single function.**
  Ex: Compute cosine of angle, Read transaction record, Assign seat to airline passanger.

9. Explain five types of user system interaction

A user interface, also sometimes called a human-computer interface, comprises both hardware and software components. It handles the interaction between the user and the system.There are different ways of interacting with computer systems which have evolved over the years. There are five main types of user interface:

- **command line (cli)**
- **graphical user interface (GUI)**
- **menu driven (mdi)**
- **form based (fbi)**
- **natural language (nli)**

Command Line Interface

Command line interfaces are the oldest of the interfaces,It involves the computer responding to commands typed by the operator. This type of interface has the drawback that it requires the operator to remember a range of different commands and is not ideal for novice users.

Graphical UI

Graphical user interfaces (GUI) are sometimes also referred to as WIMP because they use *Windows*, *Icons*, *Menus* and *Pointers*. Operators use a pointing device (such as a mouse, touchpad or trackball) to control a pointer on the screen which then interacts with other on-screen elements.

Menu Driven

A menu driven interface is commonly used on cash machines (also known as automated teller machines ('ATM's), ticket machines and information kiosks (for example in a museum). They provide a simple and easy to use interface comprised of a series of menus and sub-menus which the user accesses by pressing buttons, often on a touch-screen device.

Form Based

A form-based interface uses text-boxes, drop-down menus, text areas, check boxes, radio boxes and buttons to create an electronic form which a user completes in order to enter data into a system. This is commonly used on websites to gather data from a user, or in call centres to allow operators to quickly enter information gathered over the phone.

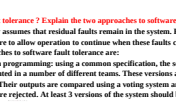
Natural language

A natural language interface is a spoken interface where the user interacts with the computer by talking to it. Sometimes referred to as a 'conversational interface', This is the kind of interface used by the popular iPhone application called Siri and Cortana in Windows

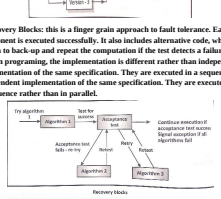
10. What is fault tolerance ? Explain the two approaches to software fault tolerance.

This strategy assumes that residual faults remain in the system. Facilities are provided in the software to allow operation to continue when these faults cause system failures. Two approaches to software fault tolerance are:

1) **N-version programming: using a common specification, the software system is implemented in a number of different teams. These versions are executed in parallel. Their outputs are compared using a voting system and inconsistent outputs are rejected. At least 3 versions of the system should be available.**



2)Recovery Blocks: this is a finger grain approach to fault tolerance. Each program component is executed successfully. It also includes alternative code, which allows the system to back-up and repeat the computation if the test detects a failure. Unlike N-version programing, the implementation is different rather than independent implementation of the same specification. They are executed in a sequence rather than independent implementation of the same specification. They are executed in sequence in sequence rather than in parallel.



11. Explain the different types of software reliability matrices.

Software reliability metrics are units of measure for system reliability. System reliability is measured by counting the number of operational failures and relating these to demands made on the system at the time of failure.
The metrics are :

- **(a) Rate of occurrence of failures (ROCOF) –This is a measure of the frequency of occurrence in which unexpected behaviour is likely to occur.**
- **(b) Mean Time to Failure (MTTF) – MTTF is the average time between two successive failures observed over a large number of failures.**

- **(c) Mean Time to Repair(MTTR) –Once the failure occurs, some time is required to fix the error that is nothing but mean time to repair.**
- **(d) Mean Time between Failure(MTBF) – This metrics is the combination of MTTF and MTTR. Thus as MTBF of 300 hours indicates that once the failure occurs the next failure is expected to occur only after 300 hours.**
- **(e) Probability of Failure on Demand(POFOD) – Unlike the other metrics explained above this metrics does not explicitly involved in the time measurements. POFOD metrics measures the systems falling when a service request is made.**
- **(f) AVAIL :The availability of system is a measure of how well the system is available for the use over a given time.**

12. What is reliability growth modelling ? Explain its types.

A GRM is a mathematical model. It explains how software reliability improves as the errors are detected and repaired. The model can be used to predict when a particular level of reliability can't be attained. Thus RGM can be used to determine when to stop testing to attain a given reliability.

Software reliability growth models have been grouped into two classes of models - concaved and S-shaped. These two model types are shown in Figure 2-2. The most important thing about both models is that they have the same asymptotic behavior, i.e., the defect detection rate decreases as the number of defects detected (and repaired) increases

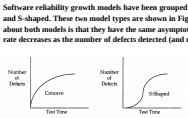


Figure 2-2. Concave and S-Shaped Models

Although several different growth models have been proposed.

There are two types of steps function model as follows:

- **Equal step Function**
- **Random step function**

1.Equal Step Function: The model has been designed by Jelinki and Moranda. In this model it is assumed that the reliability increases by a constant-increment each time an error is detected and repaired.

The demerit of this model is that it assumes that all defects contribute equally to the reliability growth. However in reliability some defects are simple and some are more complex. Hence the reliability also varies.

2. Random step Function: This model has been designed by little wood and Verall. The model allows for negative reliability growth to reflect the fact that when a repair is carried out it may introduce additional errors.

The model explains that as the errors are repaired the average improvement in reliability per repair decreases. Therefore, contribution of errors in reliability improvement is random variable.

13. Explain COCOMO model in detail.

Cocomo (Constructive Cost Model) is a regression model based on LOC, i.e number of Lines of Code. It is a procedural cost estimate model for software projects and often used as a process of reliably predicting the various parameters associated with making a project such as size, effort, cost, time and quality. It was proposed by Barry Boehm in 1970 and is based on the study of 63 projects, which make it one of the best-documented models.

Bohem'sCocomo model takes three forms :

- **Model 1 : Basic Cocomo model : This model is the starting point for project estimation which computes software development effort & cost as function of program size expressed in estimated lines of code.**
  Basic cocomo model provides an approximate estimation of software costs and is given by

$$E = a\,(KLOC)^{\,b}$$

PRODUCT COMPLEXITY	A	B
Simple	2.4	1.05
Moderate	3.0	1.12
Embedded	3.6	1.20

- **Model 2: Intermediate Cocomomodel :Intermediate cocomo makes use of cost drives and their multiples to estimate the cost.Model utilizes 15 drives such as product attribute, hardware attribute, personal attribute, project attribute etc for cost estimation. Ex: Computers, skilled professional, administrative staff etc.**

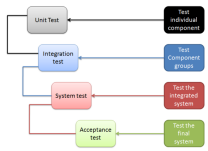
$$E = (a\,(KLOC)^{\,b}) * EAF$$

- **Model 3: Complete Cocomo model :Complex systems are made up of sub-systems, each parameter of a module must be summed up to get complete cost estimation. There are six phases in this model they are :**
  - **Planning and requirements**
  - **System design**

- Detailed design
- Module code and test
- Integration and test
- Cast Constructive model

14. Explain various levels of testing

Testing process is the creation of a test strategy or plan device to test the system. The stages/levels of testing process are :



(a) **Unit Testing : Individual components are tested to ensure that they operate correctly. Each component is tested independently without referring other system.**

(b) **Module Testing : A module is a collection of dependent components such as an object class, an abstract data type or some collection of procedures and functions.**

(c) **Integration Testing : This phase involves collection of modules which have been integrated into sub-systems. Sub-systems may be independently designed and implemented.**

(d) **System Testing : The sub-systems are integrated to make-up the entire system. It is also concerned with validating that the system meets its functional and non-functional requirements.**

(e) **Acceptance Testing : This is the final stage in the testing process before the system is accepted for operational use.**

15. Differentiate between Black box and White box testing

Software Testing can be majorly classified into two categories:

1. **Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester**
2. **White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is not known to the tester.**

BLACK BOX TESTING	WHITE BOX TESTING
• It is a way of software testing in which the internal structure or the program or the code is hidden and nothing is known about it.	• It is a way of testing the software in which the tester has knowledge about the internal structure r the code or the program of the software.
• It is mostly done by software testers.	• It is mostly done by software developers.
• It is functional test of the software.	• It is structural test of the software.
• This testing can be initiated on the basis of requirement specifications document.	• This type of testing of software is started after detail design document.
• No knowledge of programming is required.	• It is mandatory to have knowledge of programming.
• It is the behavior testing of the software.	• It is the logic testing of the software.
• It is applicable to the higher levels of testing of software.	• It is generally applicable to the lower levels of software testing.
• It is also called closed testing.	• It is also called as clear box testing.
• It is least time consuming.	• It is most time consuming.
Example: search something on google by using keywords	Example: by input to check and verify loops

16. Write a note on risk management.

Risk management is a process that is used to minimize or eradicate risk before it can harm the productivity of a software project. With only 28% of software projects finishing on time and within budget. Risk and management of risk play an important role in software development.

There are several types of risk that can occur during a software development project. This include:

Generic Risks	**Generic threats across all projects. For ex., Requirements change, loss of team members, loss of funding.**
Product-Specific Risks	**High level risks associated with the type of product being developed. For ex., Availability of testing resources.**
Project Risk	**Affect project schedule of resources**
Product Risk	**Affect quality of performance of software**
Business Risk	**Affect the viability of the software**

There are also specific risks associated with team members, customers, tools, technology, time estimation and team size. Many of these risks can be minimized by the development methodology used for the project. There are many different tools that can be used to analyze the risk apparent in a project and that can help to choose the best way to minimize or eliminate the risk.

17. Explain evolutionary and throw-away prototyping.

EVOLUTIONARY PROTOTYPING: An approach to system development where an initial prototype is produced and refined through a number of stages to the final system.

Advantages:

- **Accelerated delivery of the system: rapid delivery and deployment are sometimes more important than functionality or long-term software maintainability.**
- **User engagement with the system: not only is the system more likely to meet user requirements, they are more likely to commit to the use of the system.**

Disadvantages:

- **Management problem: Existing management processes assume a waterfall model of development. Specialist skills are required which may not be available in all development teams.**
- **Maintenance problems: Continual change tends to corrupt system structure so long-term maintenance is expensive Contractual problems.**

THROW-AWAY PROTOTYPING: A prototype which is usually a practical implementation of the system is produced to help discover requirements problems and then discarded. The system is then developed using some other development process.

Advantages:

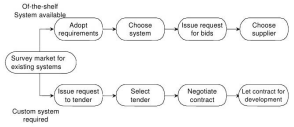
- **The speed with each prototype is put together.**
- **It also focuses the user on only one aspect of the system so keeping their feedback precise.**

Disadvantages:

- **One disadvantage with throw-away prototyping is that developers may be pressurized by the users to deliver it as a final system.**
- **Another issue is that in throw-away prototype, all the efforts put in one loft unlike evolutionary prototype.**

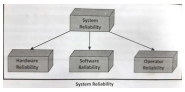
18. Describe system procurement process in detail.

- **System procurement is a process of acquiring a system for an organization to meet some identified need.**
- **The contractor or sub-contractor minimizes the number of organization which the procurer must deal with. The sub-contractor design and build parts of the system to a specification produced by the principal contractor. Once completed these different parts are integrated by the principal contractor. They are then delivered to the customer by buying the system. The procurement of large hardware or software system is usually based around some principal contractor.**
- **Depending on the contract the procurer may allow the principal contractor a free choice of sub-contractor or may require principal contractor to choice sub-contractor from an approved list.**



19. Write a note on reliability engineering.

- **Reliability is a complex concept which should always be considered at the level of systems rather than at the individual component level. The components in the system reliability namely, Hardware reliability, Software reliability, software reliable and operator reliably.**



A. Hardware reliability: What is the probability of a hardware component failing and how long does it take to replace/repair that component.

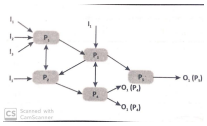
B. Software reliability: How likely is that a software component will produce an incorrect output? Software failure is usually distinct from hardware failure. In that software does not ware out. It can continue to work even after an incorrect result has been produced.

3.Operator reliability: How likely is that the operator of a system will make an error?

20. Explain thread testing with a diagram.

- **Thread testing is normally applied for testing real time application program. It is also called event based system. Since it is an event based approach; tests are based on the events which trigger system actions. Thread testing strategy which may be used after processed or objects have been individually tested and integrated into sub-systems.**

Consider the real time system made up of five interacting processes shown in the figure:



Some processes accept inputs from their own environment and generate output to that environment. These inputs may be from sensors, keyboards or some other computer systems. Similarly, outputs may be to control lines, other computer or user terminals. Inputs from the environment are labeled with an "I" and output with an "O". As part of the testing process, the system should be analyzed to identify as many threads as possible.

21 .Explain the methods for object identification?

Step 1	**BY MEANS OF NATURAL LANGUAGE GRAMMATICAL NOTATIONS: using this methodology we can refer objects and its supported attributes as "nouns" and finally various operations associated with these objects as "verbs"**
Step 2	**BY MEANS OF BEHAVIORAL APPROACH: in this mechanism considering the system's behavior is of prime focus. If possible all the participating entities of a given system are assigned with a specific behavior. Hence ,in this way the entities which are performing specific roles can be treated as objects of that system**
Step 3	**BY MEANS OF SCENARIO BASED ANALYSIS PROCESS: in this mechanism, a scenario of a given system is built and is analyzed deeply. Hence during analysis of this system ,various objects and its specifications can be easily determined**
Step 4	**BY USING FOLLOWING SPECIFICATION Usage of events such as request/reply Usage of interactions such as meetings Usage of tangible entities such as car etc Usage of locations such as offices Usage of organizational units such as companies**

22. What are volatile requirements? Explain the classification of volatile requirements.

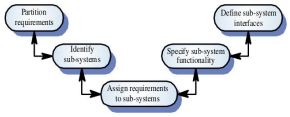
Volatile requirements are unstable requirements and are likely to change during the system development process or after the system has been put into (operational) use.

Classification of volatile requirements are:

Types	Description
Mutable requirements	**Requirements that changes due to the system environment in which the organization is operating**
Emergent requirements	**Requirements that emerge as the customers understanding of the system develops during the system development**
Consequential requirements	**Requirements that result from the introduction of the computer system. Introducing the computer system may change the organizations processes and open up new ways of working.**
Compatibility requirements	**Requirements that depend on other systems or business process within an organization.**

23. Explain the different phases of system design process with a diagram?

System design is concerned with how the system functionality is to be provided by the components of the system. The different phases are:



- **Partition requirements: analyze the requirement and organize them into related groups.**
- **Identify subsystems: identify sub system that can individually or collectively meet the requirements. group of requirements are usually related to sub systems so this activity and requirement partitioning may be carried out together.**
- **Assign requirements to sub system: assign the requirements to each identified sub systems .in principle this should be straight forward if the requirements partitioning is used to drive the sub system identification. in practice there is never a clean match between requirements partitions and identified sub systems. limitations of COTS sub system may mean that requirements have to be modified**
- **Specify subsystem functionality: the specific functions provided by each sub system are specified. This may be seen as a part of the system design phase or if the subsystem is a software system, part of the system requirement specification activity for that system. relationship between sub system should also be identified at this stage.**
- **Define sub system interface: define the interfaces that are provided and expected by each subsystems. once these interfaces have been agreed parallel development of the subsystem becomes possible.**

24. Explain types of software maintenance.

In a software lifetime, type of maintenance may vary based on its nature. It may be just a routine maintenance tasks as some bug discovered by some user or it may be a large event in itself based on maintenance size or nature. Following are some types of maintenance based on their characteristics:

- **Corrective maintenance: this includes modifications and updations done in order to correct or fix problems, which are either discovered by user or concluded by user error reports.**
- **Adaptive maintenance: this includes modifications and updations done in order to correct or fix product up-to date and tuned to the ever changing world of technology and business environment.**
- **Perfective maintenance: this includes modifications and updates done in order to keep the software usable over long period of time. It includes new features, new user keep the software usable over long period of time. It includes new features, new user requirements for refining the software and improve its reliability and performance.**
- **Preventive maintenance: this includes modifications and updations to prevent future problems of the software. It aims to attend problems, which are not significant at this moment but may cause serious issues in future.**

	corrective	Enhancement
Proactive	**Preventive**	**Perfective**
Reactive	**Corrective**	**Adaptive**

25. Discuss the Challenges of Software Engineer.

The software engineering discipline has been faced with a number of challenges over the years, including those related to quality, management and cost estimation. Although numerous approaches, including methods and frameworks have been introduced and adopted as industry standards and/or best practices that have mitigated many of these issues. The discipline is still faced with a number of challenges, and future challenges are also bound to appear.

There are 3 challenges. They are :

1.The legacy Challenge

2.the heterogeneity challeage

3.The delivery challenge

4.Changing requirements

5.Schedule Optimism

- **The legacy challenge: The majority of software systems which are in use today were developed many years ago yet they perform critical business functions. The legacy challenge is the challenge of maintaining and updating this software in such a way that excessive costs are avoided and essential business services continue to be delivered.**
- **The heterogeneity challenge: Increasingly, systems are required to operate as distributed systems across networks that include different types of computers and with different kinds of support systems. The heterogeneity challenge is the challenge of developing techniques to build dependable software which is flexible enough to cope with this heterogeneity.**
- **The delivery challenge: Many traditional software engineering techniques are time-consuming to deliver a quality software. However business operation today change very frequently, so supporting software must also change rapidly. Software time should be reduced without compromising on quality of a software product.**

26. Write a shot note on user interface design.

System users often judge a system by its interface rather than its functionality. A poorly designed interface can cause a user to make catastrophic errors. The poor user interface design is the reason why many software systems are never used. Most users of business systems interact with these systems through graphical user interfaces (GUI's). In some cases, legacy text-based interfaces are still used.

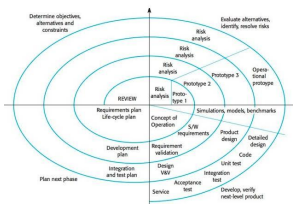
- **GUI characteristics**

Characteristics	Description
Windows	**Multiple windows allow different information to be displayed simultaneously on the user's screen.**
Icons	**Icons represents different types of information. On some systems, icons represent files; on others, icons represent processes.**
Menus	**Commands are selected from a menu rather than typed in a command language.**
Pointing	**A pointing device such as a mouse is used for selecting choices from a menu or indicating items of interest in a window.**
Graphics	**Graphical elements can be mixed with text on the same display.**

- **Advantages of GUI**
  - **They are easy to learn and use. Users without experience can learn to use the system quickly.**
  - **The user may switch quickly from one task to another and can interact with several different applications. Information remains visible in its own window when attention is switched.**
  - **Fast, full-screen interaction is possible anywhere on the screen.**

27. Explain Spiral Model in Detail.

- **The Spiral Life Cycle Model is a type of iterative software development model which is generally implemented in high risk projects. It was first proposed by Bohem in 1988.**
- **The spiral model is similar to the incremental model, with more emphasis placed on risk analysis. In this system development method, it combines the best features of both, waterfall model and prototype model. Each loop in the spiral represents a phase of the software process. Thus, the innermost loop might be concerned with system feasibility, the next loop with requirements definition, the next loop with system design and so on.**



Each phase of Spiral Model is divided into four quadrants as shown in the above figure. The functions of these four quadrants are :

- **Objectives determination and identify alternative solutions: Requirements are gathered from the customers and the objectives are identified, elaborated and analyzed at the start of every phase.**
- **Identify and resolve Risks: During the second quadrant all the possible solutions are evaluated to select the best possible solution. Then the risks associated with that solution is identified and the risks are resolved using the best possible strategy**
- **Develop next version of the Product: During the third quadrant, the identified features are developed and verified through testing. At the end of the third quadrant, the next version of the software is available.**
- **Review and plan for the next Phase: In the fourth quadrant, the Customers evaluate the so far developed version of the software. In the end, planning for the next phase is started.**

Advantages of Spiral Model:

- **Risk Handling: Spiral Model is the best development model to follow due to the risk analysis and risk handling at every phase.**
- **Good for large projects: It is recommended to use the Spiral Model in large and complex projects.**
- **Flexibility in Requirements: Change requests in the Requirements at later phase can be incorporated accurately by using this model.**
- **Customer Satisfaction: Customer can see the development of the product at the early phase of the software development and thus, they habituated with the system by using it before completion of the total product.**

Disadvantages of Spiral Model:

- **Complex: The Spiral Model is much more complex than other SDLC models.**
- **Expensive: Spiral Model is not suitable for small projects as it is expensive.**
- **Too much dependable on Risk Analysis: The successful completion of the project is very much dependent on Risk Analysis. Without very highly experienced expertise, it is going to be a failure to develop a project using this model.**
- **Difficulty in time management: As the number of phases is unknown at the start of the project, so time estimation is very difficult**

28. Explain the contents of test plan.

The test plan is the document which is created before the testing process. It includes the type of testing that will be performed, high level scope of the project, the environmental requirements of the testing process, what automated testing tools will be used the schedule of each test, when it will start and end etc.

Test plan identifier • **Provide a unique for the document.**
Introduction: • **Provide an overview of the test plan.** • **Specify the goals** • **Specify any constraints.**
References: • **List the related documents, with links to them if available, including the following:** **1. Project plan** **2. Configuration management Plan**
Test Items: • **List the test items and their versions.**
Features to be tested: • **List the features of the software/product to be tested.** • **Provide references to the requirements and/or Design specifications of the features to be tested.**
Features not to be Tested: • **List the features of the software/product which will not be tested.** • **Specify the reasons these features won't be tested.**
Approach: • **Mention the overall approach to testing.** • **Specify the testing levels, testing types, and the testing methods.**
Item pass/fail Criteria: • **Specify the criteria that will be used to determine whether each test item has passed or failed testing.**
Suspension Criteria and Resumption Requirements: • **Specify criteria to be used to suspend the testing activity.** • **Specify testing activities which must be redone when is resumed.**
Test Deliverables: • **List test deliverables, and links to them if available, including the following:** **1. Test Plan** **2. Test cases** **3. Test Scripts** **4. Defect /Enhancement logs** **5. Test Reports**
Test Environment: • **Specify the properties of test environment: Hardware, Software, network etc.** • **List any testing or related tools.**
Estimate:
• **Provide a summary of test estimates and/or provide a link to the detailed estimation.**
Schedule: • **Provide a summary of test schedule, specifying key test milestones, and/or provide a link to the detailed estimated.**
Staffing and Training Needs: • **Specify staffing needs by role and required skills.** • **Identify training that is necessary to provide those skills, if not already acquired.**
Responsibilities: • **List the responsibilities of each team/role/individual.**
Risks: • **List the risk that have been identified.** • **Specify the migration plan and the contingency plan for each risk.**
Assumptions: • **List the assumptions that have been made during the preparation of this plan.** • **List the dependencies.**
Approvals: • **Specify the names and roles of all persons who must approve the plan.** • **Provide space for signatures and dates.**

29. Explain the quality characteristics of design.

A good design might be a design that allows efficient code to be produced. It might be A minimal design where the implementation is as compact as possible or it might be the most maintainable design. Following are some of the quality characteristics that equally applicable to object-oriented and function-oriented design. These factors are:

1. **Correctness**
2. **Understandability**
3. **Efficiency**
4. **Maintainability**

- **Correctness : The design of any software is evaluated for its correctness. The evaluators check the software for every kind of input and action and observe the results that the software will produce according to the proposed design.**
- **Understandability : The software design should be understandable so that the developers do not find any difficulty to understand it. Good software design should be self-explanatory.**
- **Efficiency : The software design must be efficient. The efficiency of the software can be estimated from the design phase itself, because if the design is describing software that is not efficient and useful, then the developed software would also stand on the same level of efficiency.**
- **Maintainability : The software design must be in such a way that modifications can be easily made in it. This is because every software needs time to time modifications and maintenance. So, the design of the software must also be able to bear such changes.**

30. Explain the characteristics of a good SRS.

Following are the characteristics of a good SRS document:

1. **Correctness: SRS is said to be correct if it covers all the requirements that are actually expected from the system.**
2. **Completeness: Completeness of SRS indicates every sense of completion including the numbering of all the pages, resolving the to be determined parts to as much extent as possible as well as covering all the functional and non-functional requirements properly.**
3. **Consistency: Requirements in SRS are said to be consistent if there are no conflicts between any set of requirements.**
4. **Unambiguousness: An SRS is said to be unambiguous if all the requirements stated have only 1 interpretation.**
5. **Ranking for importance and stability: There should a criterion to classify the requirements as less or more important or more specifically as desirable or essential**
6. **Modifiability: SRS should be made as modifiable as possible and should be capable of easily accepting changes to the system to some extent.**

7. **Verifiability: An SRS is verifiable if there exists a specific technique to quantifiably measure the extent to which every requirement is met by the system.**
8. **Traceability: One should be able to trace a requirement to a design component and then to a code segment in the program.**
9. **Design Independence: There should be an option to choose from multiple design alternatives for the final system.**
10. **Testability: An SRS should be written in such a way that it is easy to generate test cases and test plans from the document.**
11. **Understandable by the customer: An end user maybe an expert in his/her specific domain but might not be an expert in computer science. Hence, the use of formal notations and symbols should be avoided to as much extent as possible. The language should be kept easy and clear.**
12. **Right level of abstraction: If the SRS is written for the requirements phase, the details should be explained explicitly. Whereas, for a feasibility study, fewer details can be used. Hence, the level of abstraction varies according to the purpose of the SRS.**

31. Write a short note on quality control?

- **Quality control means testing and it measures the quality of a product.the goal of quality control is to ensure that the products services of processes provided must specific requirements and are dependable and satisfactory.**

The QC system is designed to:

- **Provide routine and consistent checks to ensure data integrity,correctness and completeness.**
- **Identify and address errors and omissions.**
- **Document and archive inventory material and record all QC activities.**

Essentially, QC involves the examination of a product, service or process for certain minimum levels of quality. the goal of a quality control team is to identify products or services that do not meet a company's specified standards of quality.

QC concerns not just products ,services and processes ,but also people. if a company has employees that do not have adequate skills or training and knowledge then quality may be severely diminished.

There are seven primary quality control tools which include:

- **Checklists. At its most basic, quality control requires you to check off a list of items that are imperative to manufacture and sell your product.**
- **Fishbone diagram. ...**
- **Control chart. ...**
- **Stratification. ...**
- **Pareto chart. ...**
- **Histogram. ...**
- **Scatter Diagram.**

32. Write a short note on Software Quality Management?

Software Quality Management ensures that the required level of quality is achieved by submitting improvements to the product development process. SQA aims to develop a culture within the team and it is seen as everyone's responsibility.

Software Quality management should be independent of project management to ensure independence of cost and schedule adherences. It directly affects the process quality and indirectly affects the product quality.

Activities of Software Quality Management:

- **Quality Assurance - QA aims at developing Organizational procedures and standards for quality at Organizational level.**
- **Quality Planning - Select applicable procedures and standards for a particular project and modify as required to develop a quality plan.**
- **Quality Control - Ensure that best practices and standards are followed by the software development team to produce quality products.**

33. Explain quality assurance in brief?

- **Software Quality Assurance (SQA) is simply a way to assure quality in the software. It is the set of activities which ensure processes, procedures as well as standards suitable for the project and implemented correctly.**

Software Quality Assurance have:

- ✓ **A quality management approach**
- ✓ **Formal technical reviews**
- ✓ **Multi testing strategy**
- ✓ **Effective software engineering technology**
- ✓ **Measurement and reporting mechanism**

Major Software Quality Assurance Activities:

- ✓ **SQA Management Plan**
- ✓ **Set The Check Points**
- ✓ **Multi testing Strategy**
- ✓ **Measure Change Impact**
- ✓ **Manage Good Relations:**

Benefits of Software Quality Assurance (SQA):

- ✓ **SQA produce high quality software.**
- ✓ **High quality application saves time and cost.**
- ✓ **SQA is beneficial for better reliability.**
- ✓ **SQA is beneficial in the condition of no maintenance for long time.**
- ✓ **High quality commercial software increase market share of company.**
- ✓ **Improving the process of creating software.**
- ✓ **Improves the quality of the software.**

34. Describe different requirements validation checks?

During the requirements validation process,checks should be carried out on the requirements in the requirements document.these checks include:

REQUIREMENT VALIDITY CHECKS	DESCRIPTION
1.Validity checks	**These checks aims to ensure that the system meets all functional, behavioral and performance requirements**
2.Consistency checks	**Requirements collected must be consistent and should not lead to conflict ie there should not be contradictory Constraints or different descriptions of the same system function**
3.Realism checks	**Using knowledge of existing technology the requirement should be checked to ensure that they can actually be implemented. these checks also take account of the budget and schedule for the system development**
4.Verifiability	**At the completion of the system, it must be possible to demonstrate that the delivered system meets all the requirements**