SE VIVA QUESTIONS

1. Software Engineering:
    - An engineering branch associated with software system development.
    - A discipline concerned with all the aspects of software production.

2. Computer Software:
    - A complete package which includes a software program, its documentation and user's guide for how to use the software.

3. Software Re-Engineering:
    - It is a process to upgrade the technology on which the software is built without changing the functionality of the software.
    - This is done in order to keep the software tuned with the latest technology.

4. Software Process or Software Development Life Cycle:
    - It is the systematic development of a software by following its generic activities such as:
        a. Requirement Gathering
        b. System Analysis
        c. Design
        d. Coding
        e. Testing
        f. Deployment & Maintenance
        g. Documentation

5. SDLC Models:
    - Waterfall Model
    - V shape Model
    - Spiral Model
    - Big Bang Model
    - Agile Model

6. Waterfall Model:
    - Also called a linear sequential model.
    - It gives a systematic and sequential approach.
    - In this model, each phase depends on the previous stage's results.
    - Used when requirements are fixed and well known.
    - There are 5 phases:
        a. Communication: requirements gathered from the clients.
        b. System Design: planning of language to use and high level details.
        c. Implementation: coding stage.
        d. Deployment: deploy application in environment.
        e. Maintenance: ensures that the application is running smoothly and changes are made if necessary.

7. RAD Model (Rapid Application Development) :
    - Extremely short development cycle + component based construction.
    - The RAD model is based on prototyping and iterative development with no specific planning involved.

- It focuses on gathering customer requirements through workshops or focus groups, early testing of the prototypes by the customer using iterative concepts, reuse of the existing prototypes (components), continuous integration and rapid delivery.

8. Prototyping Model:
    - It is used when the customer does not know completely about how the end product should be and its requirements.
    - So in this model, a prototype of the end product is first developed by the developers and then tested and changes are made as per customer feedback until the customer is satisfied with the prototype.

9. Big Bang Model:
    - SDLC model which does not follow any procedure and very little planning is required.
    - Development starts with the required money and efforts as inputs.
    - Output may or may not be as per customer's requirements.
    - Used for small projects.

10. Agile Methodology:
    - Refers to software development methodologies centered around the idea of iterative development where requirements and solutions evolve through collaboration between different teams.
    - It involves iterative development and incremental release of product.
    - Software development is divided in small periods of time: at most 2 weeks
    - This helps reduce risks and allows the product to change quickly.
    - Set of guidelines and requirements are specified before starting new iterations.
    - At the end of each iteration, the product is presented to the customer who reviews it and gives its feedback.
    - If any changes are again requested, they are incorporated before the release of the final product.

11. Need of Agile:
    - Since existing SDLC models are predefined and preplanned, it is difficult to make any changes. It is necessary to specify all the requirements of a project which is not really feasible.
    - Agile methodology provides an adaptive approach in which customers review the prototype which covers the subset of requirements at the end of each iteration.

12. Types of Agile Methods:
    - Scrum
    - Extreme Programming

13. Scrum:
    - Agile framework which focuses on how to manage tasks within a team based development environment.

- It is designed for teams, where the work is broken into activities that can be completed within the time based iterations called sprints.
- In this, each iteration phase has a duration decided by the team which is fixed. The deliverable product is created each sprint.
- A product backlog which includes a set of requirements is prepared at the beginning of each sprint.
- Expected requirements of products which are not fulfilled in the first sprint cycle are incorporated in the next sprint cycle.

14. Extreme Programming:
    - It is used to improve software quality and responsive to customer requirements.
    - The extreme programming model recommends taking the best practices that have worked well in the past in program development projects to extreme levels.
    - Refactoring in XP: a technique of improving code without changing functionality.
    - Refactoring is an ongoing process of simplification that applies to the code, design, testing and XP itself.
    - Principles of Extreme programming are Coding. Testing, Listening, Designing, Feedback, Simplicity.
    - Duration of each iteration is comparatively less than that of scrum.
    - Duration of each iteration is not fixed.
    - Duration can be extended if all requirements have not been covered.

15. Functional Testing:
    - It is a type of testing that validates the software system against the functional/ expected requirements.
    - Purpose of functional testing is to test each function of an application by providing appropriate input and verifying the output.
    - Examples: Unit Testing, Smoke Testing, Integration Testing, White Box Testing, Black Box Testing, Regression Testing.

16. Non-Functional Testing:
    - It is a type of software testing to check the non functional aspects of the software application (performance, usability).
    - It is designed to test the working of a system as per non functional parameters.Examples: Performance, Endurance, Load, Volume, Scalability, etc.

17. Testing:
    - It is a method to check whether the product meets the requirements set by the customer and that it is defect free.
    - It is a process of executing a program with the intention of finding errors.

    a. Unit Testing:
        - Unit Test Plans are developed during the module design phase.
        - These Unit Test Plans are executed to eliminate bugs at code or unit level.

b. Integration testing:
    - After completion of unit testing, Integration testing is performed.
    - In integration testing, the modules are integrated and the system is tested.
    - Integration testing is performed on the Architecture design phase.
    - This test verifies the communication of modules among themselves.
    - Types:
        a. Big Bang Approach :
        b. Incremental Approach:
            - Top Down Approach
            - Bottom Up Approach
            - Sandwich Approach - Combination of Top Down and Bottom Up

c. System Testing:
    - System testing tests the complete application with its functionality, inter dependency, and communication.
    - It tests the functional and non-functional requirements of the developed application.

d. User Acceptance Testing (UAT):
    - UAT verifies that the delivered system meets the user's requirement and the system is ready for use in the real world.

e. Smoke testing:
    - It checks if the software build is stable and can be used by the Quality Assurance team for further testing.

f. Regression testing:
    - It checks for any small changes in the code and whether any other existing functionalities are affected due to the newly developed code.

18. Capability maturity model (CMM):
    - It is a framework which is used to analyse the approach and techniques followed by any organization to develop software products.
    - It also provides guidelines to further enhance the maturity of the process used to develop those software products.
    - It is based on profound feedback and development practices adopted by the most successful organizations worldwide.
    - Steps of CMM:
    - Level 1 (Initial): processes are disorganized and success is likely to be on individual efforts as the processes are not well defined.
    - Level 2 (Repeatable): basic management projects are established and successes could be repeated because the required processes are well defined and documented in order to replicate them.
    - Level 3 (Defined): an organization has developed its own standard software process through greater attention to documentation and integration.
    - Level 4 (Managed): the organization controls and monitors its own processes through data collection and analysis.
    - Level 5 (Optimizing): processes are constantly being improved through monitoring and feedback from current processes and introducing innovative processes.

19. Software Requirement Specification:
    - It is a detailed description of a software system to be developed with its functional and non-functional requirements.
    - It is developed based on the agreement between customer and contractors.
    - It may include the use cases of how the user is going to interact with the software system.
    - This document consists of all necessary requirements required for project development.

20. Functional requirements:
    - Functional requirements are functional features and specifications expected by users from the proposed software product.

21. Non-Functional requirements:
    - Non-functional requirements are implicit and are related to security, performance, look and feel of user interface, interoperability, cost etc.

22. Requirement gathering techniques:
    - One-on-one interviews
    - Group interviews
    - Facilitated sessions
    - Questionnaires
    - Prototyping
    - Use cases
    - Brainstorming

23. Verification:
    - Verification checks if proper steps are followed to develop the product.
    - Verification confirms if the product is built in the right way.
    - Verification is followed by Validation.

24. Validation:
    - Validation checks if the product is made as per user requirements.
    - Validation confirms the right product

25. Data Flow Diagram (DFD):
    - It is a visual representation of the information flow within a system.
    - It shows how data enters and leaves the system, what changes the information, and where data is stored.

26. Elements of Data Flow Diagram:
    a. Data Store: A data store is a holding place for information within the system
    b. Process: A process performs transformation or manipulation on input data and produces output data.
    c. Entity: An entity is anything that interacts with a system and is a source or destination of a data.
    d. Data Flow: A data flow shows the flow of information from its source to its destination.

27. UML (Unified Modeling Language):
   - The UML is a standard for specifying, visualizing, constructing, and documenting the artifacts of software systems, as well as for business modeling and other non-software systems.
   - The UML is a very important part of developing Object Oriented software and the software development process.
   - The UML uses mostly graphical notations to express the design of software projects.

28. UML diagrams:
   a. Use Case Diagram
   b. Class Diagram
   c. Sequence Diagram
   d. Collaboration Diagram
   e. Activity diagram
   f. State Chart Diagram
   g. Component Diagram
   h. Deployment Diagram

29. <<includes>>:
   - Specifies that source use case explicitly interprets the behavior of another use case.

30. <<extends>>:
   - Specifies that target use cases extend the behaviour of the source.

31. Software project estimation techniques:
   a. Decomposition technique (Counting Lines of Code and Function Points).
   b. Empirical technique (Putnam and COCOMO).

32. Function points:
   - Various features provided by the software product.
   - It is considered as a unit of measurement for software size.

33. COCOMO Model (Constructive Cost Model):
   - Cocomo 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 predicts the efforts and schedule of a software product based on the size of the software.

34. Earned Value Analysis:
   - It is an approach to measuring progress in a software project is to calculate how much has been accomplished.
   - It is basically the percentage of the estimated time that has been completed.

35. Cohesion:

- Cohesion represents the relationship within a module.
- Cohesion is the concept of intra intra-module.
- Increasing cohesion is good for software.
- Cohesion represents the functional strength of modules.

36. Coupling:

- Coupling represents the relationships between modules.
- Coupling is the concept of inter module.
- Increasing in coupling is avoided for software.
- Coupling represents the independence among modules.

37. Formal Technical Review (FTR):

- It is a software quality control activity performed by software engineers.
- The purpose of FTR is to verify that the software meets specified requirements.
- To ensure that software is represented according to predefined standards.

38. Software Configuration Management:

- It is a process to systematically manage, organize and control the changes in the documents, code and other entities during SDLC.
- Its main goal is to increase productivity with minimal mistakes.
- Steps of SCM:
    a. Identification of objects in software configuration:
        - It sets the basis for the subsequent tasks of the SCM process.
        - It includes defining the basis for identifying software configuration items (SCIs).
    b. Version Control:
        - combines procedures and tools to handle different versions of configuration objects that are generated during the software process.
    c. Change Control:
        - includes evaluating a change request, implementing the change, and verifying and releasing the change.
    d. Configuration Audit:
        - SCM audits to verify that the software product satisfies the baselines requirements and ensures that what is built and what is delivered.
        - It also ensures that traceability is maintained between all Continuous items and that all work requests are associated with one or more CI modifications.
    e. Status Reporting:
        - Provides accurate status and current configuration data to developers, testers, end users, customers and stakeholders through admin guides, user guides, FAQs, Release Notes, Installation Guide, Configuration Guide, etc.

39. RMMM:
- Risk management technique is usually seen in the software Project plan.
- It can be divided into Risk Mitigation, Monitoring, and Management Plan.
- In this plan, all work is done as part of risk analysis.
- Risk Mitigation: It is an activity used to avoid risks (Risk Avoidance).
- Risk Monitoring: It is the process of tracking risk management execution and continuing to identify and manage new risks.
- Risk Management and planning: It assumes that the mitigation activity failed and the risk is a reality. This task is done by the Project manager when risk becomes reality and causes severe problems. If the project manager effectively uses project mitigation to remove risks successfully then it is easier to manage the risks. This shows the response that will be taken for each risk by a manager.

40. Requirement Modelling:
- Requirements modeling is the process used in software development projects where requirements and solutions constantly evolve through collaborative efforts and teamwork.
- It comprises several stages, or 'patterns': scenario-based modeling, data modeling, flow-oriented modeling, class-based modeling and behavioral modeling.
- Each of these stages/patterns examines the same problem from a different perspective.
    a. Scenario based modelling: Using a scenario-based approach, the system is described from the user's point of view.
    b. Class-based modelling: It identifies classes, attributes and relationships that the system will use. The elements of the class based model consist of classes and object, attributes, operations, class – responsibility - collaborator (CRS) models.
    c. Behavior based modelling: It is specially designed to make us understand behavior and factors that influence behavior of a system. Behavior of a system is explained and represented with the help of a diagram. This diagram is known as the State Transition Diagram. It is a collection of states and events.

41. Black Box Testing:
- Black box testing is a high level of testing that focuses on the behavior of the software. It involves testing from an external or end-user perspective.
- It can be applied to virtually every level of software testing: unit, integration, system, and acceptance.
- In this, a tester doesn't have any information about the internal working of the software system.
- Testers perform testing only on the functional part of an application to make sure the behavior of the software is as expected. It is also known as input-output testing.
- Techniques for: Equivalence partitioning, Boundary Value Analysis, Decision table, State transition.

42. White Box Testing:

- White-Box testing is considered as low-level testing which checks the internal functioning of the system.
- In this system the tester has some idea of the internal working of the system.
- In this method, testing is based on coverage of code statements, branches, paths or conditions.
- It is also called glass box, transparent box, clear box or code base testing.
- This testing method assumes that the path of the logic in a unit or program is known.

43. Alpha Testing:

- Alpha Testing is a type of software testing performed to identify bugs before releasing the product to real users or to the public.
- Alpha Testing is one of the user acceptance testing.

44. Beta Testing:

- Beta Testing is performed by real users of the software application in a real environment.
- Beta testing is one type of User Acceptance Testing.

45. Difference between function oriented and object oriented design:

- Function-oriented design consists of many smaller sub-systems known as functions. Each function is capable of performing significant tasks in the system.
- Object oriented design works around the real world objects (entities), their classes (categories) and methods operating on objects (functions).

46. Joining:

- Joint vertices serve to merge several transitions coming from source vertices in different orthogonal regions. The transitions entering a join vertex cannot have guards or triggers.

47. Forking:

- Fork vertices in the UML Statechart Diagram serve to split an incoming transition into two or more transitions terminating on orthogonal target vertices. The segments outgoing from a fork vertex must not have guards or triggers.

48. Cyclomatic complexity of a program:

- Cyclomatic complexity uses graph theory's formula: $V(G) = e - n + 2$, where e: no of edges, n: no. of nodes

49. Risk:

- It is a problem that could cause some loss or threaten the progress of the project, but which has not happened yet.

50. Types of Risks:

a. Schedule Risk :
    - Schedule related risks refers to time related risks or project delivery related planning risks. The wrong schedule affects the project development and delivery.
    - If not managed properly it can lead to project failure and at last, it affects the organization/company economy very badly.

b. Budget Risk :
    - Budget related risks refers to the monetary risks and mainly occurs due to budget overruns.
    - So proper finance distribution and management are required for the success of a project, otherwise it may lead to project failure.

c. Operational Risks :
    - Operational risk refers to the procedural risks which are the risks which happen in day-to-day operational activities during project development due to improper process implementation or some external operational risks.

d. Technical Risks :
    - Technical risk refers to the functional risk or performance risk which means this technical risk is mainly associated with functionality of the product or performance part of the software product.

e. Programmatic Risks :
    - Programmatic risks refers to the external risk or other unavoidable risks.
    - These are the external risks which are unavoidable in nature and come from outside and it is out of control of programs.