**1. What is software re-engineering?**

Software reengineering is the process of scanning, modifying, and reconfiguring a system in a new way. The principle of reengineering applied to the software development process is called software reengineering. It has a positive impact on software cost, quality, customer service, and shipping speed. Software reengineering improves software to create it more efficiently and effectively.

For more details please refer to [What Is Software Re-Engineering?.](https://www.geeksforgeeks.org/software-re-engineering/)

**2. What are the characteristics of Software?**

There are various characteristics of software:

* **Software is developed or engineered; it is not manufactured in the classical sense:**
  + Although some similarities exist between software development and hardware manufacturing, few activities are fundamentally different.
  + In both activities, high quality is achieved through good design, but the manufacturing phase for hardware can introduce quality problems than software.
* **The software doesn’t “wear out.”:**
  + Hardware components suffer from the growing effects of many other environmental factors. Stated simply, the hardware begins to wear out.
  + Software is not susceptible to the environmental maladies that cause hardware to wear out.
  + When a hardware component wears out, it is replaced by a spare part.
  + There are no software spare parts.
  + Every software failure indicates an error in design or in the process through which design was translated into machine-executable code. Therefore, the software maintenance tasks that accommodate requests for change involve considerably more complexity than hardware maintenance. However, the implication is clear—the software doesn’t wear out. But it does deteriorate.
* **The software continues to be custom-built:**
  + A software part should be planned and carried out with the goal that it tends to be reused in various projects.
  + Current reusable segments encapsulate the two information and the preparation that is applied to the information, empowering the programmer to make new applications from reusable parts.
  + In the hardware world, component reuse is a natural part of the engineering process

For more details please refer to the following article[Software Engineering Characteristics](https://www.geeksforgeeks.org/software-engineering-software-characteristics/).

**3. What activities come under the umbrella** **activities?**

The activities of the software engineering process framework are complemented by a variety of higher-level activities. Umbrella activities typically apply to the entire software project and help the software team manage and control progress, quality, changes, and risks. Common top activities include Software Project Tracking and Control Risk Management, Software Quality Assurance Technical Review Measurement Software Configuration Management Reusability Management Work Product Preparation and Production, etc.

**4. What is Cohesion and Coupling?**

Cohesion indicates the relative functional capacity of the module. Aggregation modules need to interact less with other sections of other parts of the program to perform a single task. It can be said that only one coagulation module (ideally) needs to be run. Cohesion is a measurement of the functional strength of a module. A module with high cohesion and low coupling is functionally independent of other modules. Here, functional independence means that a cohesive module performs a single operation or function. The coupling means the overall association between the modules.

Coupling relies on the information delivered through the interface with the complexity of the interface between the modules in which the reference to the section or module was created. High coupling support Low coupling modules assume that there are virtually no other modules. It is exceptionally relevant when both modules exchange a lot of information. The level of coupling between two modules depends on the complexity of the interface.

For more details, please refer to the following article [Coupling and cohesion.](https://www.geeksforgeeks.org/software-engineering-coupling-and-cohesion/)

**5. What are the various phases of SDLC?**

**SDLC phases:**

* Requirement gathering & analysis
* Design
* Implementation & coding
* Testing
* Deployment
* Maintenance

For more details, please refer to the following article [Software Development Life Cycle.](https://www.geeksforgeeks.org/software-development-life-cycle-sdlc/)

**6. What is the name of various CASE tools?**

* Requirement Analysis Tool
* Structure Analysis Tool
* Software Design Tool
* Code Generation Tool
* Test Case Generation Tool
* Document Production Tool
* Reverse Engineering Tool

For more details, please refer to the following article[Computer-Aided Software Engineering(CASE).](https://www.geeksforgeeks.org/computer-aided-software-engineering-case/)

**7. What is Black box testing?**

The black box test (also known as the conducted test closed box test opaque box test) is centered around software useful prerequisites. In other words, it is possible to guess a set of information conditions that help the program through an attempt to discover and fulfill all the necessities perfectly. There is no choice of black-box testing white box procedures. Maybe it’s a complementary methodology, perhaps the white box method will reveal the errors of other classes.

For more details, please refer to [the following article Software Engineering – Black Box Testing.](https://www.geeksforgeeks.org/software-engineering-black-box-testing/)

**8.  What is White box testing?**

White Box Testing is a method of analyzing the internal structure, data structures used, internal design, code structure, and behavior of software, as well as functions such as black-box testing. Also called glass-box test or clear box test or structural test.

For more details, please refer to the following article[Software Engineering – White Box Testing.](https://www.geeksforgeeks.org/software-engineering-white-box-testing/)

**9.  What is a Feasibility Study?**

The Feasibility Study in Software Engineering is a study to assess the adequacy of proposed projects and systems. A feasibility study is a measure of a software product on how product development can benefit an organization from a validity analysis or practical point of view. Feasibility studies are conducted for multiple purposes to analyze the correctness of a software product in terms of development, porting, the contribution of an organization’s projects, and so on.

For more details, please refer to the following article [Types of Feasibility Study in Software Project Development article.](https://www.geeksforgeeks.org/types-of-feasibility-study-in-software-project-development/)

**10.  What is the Difference Between Quality Assurance and Quality Control?**

| **Quality Assurance (QA)** | **Quality Control (QC)** |
| --- | --- |
| It focuses on providing assurance that the quality requested will be achieved. | It focuses on fulfilling the quality requested. |
| It is the technique of managing quality. | It is the technique to verify quality. |
| It does not include the execution of the program. | It always includes the execution of the program. |
| It is a managerial tool. | It is a corrective tool. |
| It is process-oriented. | It is product-oriented. |
| The aim of quality assurance is to prevent defects. | The aim of quality control is to identify and improve the defects. |
| It is a preventive technique. | It is a corrective technique. |
| It is a proactive measure. | It is a reactive measure. |
| It is responsible for the full software development life cycle. | It is responsible for the software testing life cycle. |
| Example: Verification | Example: Validation |

**11.  What is the difference between Verification and Validation?**

| **Verification** | **Validation** |
| --- | --- |
| Verification is a static practice of verifying documents, design, code, black-box, and programs human-based. | Validation is a dynamic mechanism of validation and testing the actual product. |
| It does not involve executing the code. | It always involves executing the code. |
| It is human-based checking of documents and files. | It is computer-based execution of the program. |
| Verification uses methods like inspections, reviews, walkthroughs, and Desk-checking, etc. | Validation uses methods like black box (functional) testing, gray box testing, and white box (structural) testing, etc. |
| Verification is to check whether the software conforms to specifications. | Validation is to check whether the software meets the customer’s expectations and requirements. |
| It can catch errors that validation cannot catch. | It can catch errors that verification cannot catch. |
| Target is requirements specification, application and software architecture, high level, complete design, and database design, etc. | Target is an actual product-a unit, a module, a bent of integrated modules, and an effective final product. |
| Verification is done by QA team to ensure that the software is as per the specifications in the SRS document. | Validation is carried out with the involvement of the testing team |
| It generally comes first done before validation. | It generally follows after verification. |
| It is low-level exercise. | It is a High-Level Exercise. |

For more details, please refer to the following article [Software Engineering – Verification and Validation.](https://www.geeksforgeeks.org/software-engineering-verification-and-validation/)

**12. What is reverse engineering?**

**Software Reverse Engineering** is a process of recovering the design, requirement specifications, and functions of a product from an analysis of its code. It builds a program database and generates information from this. The purpose of reverse engineering is to facilitate maintenance work by improving the understandability of a system and producing the necessary documents for a legacy system.

**Reverse Engineering Goals:**

* Cope with Complexity.
* Recover lost information.
* Detect side effects.
* Synthesize higher abstraction.
* Facilitate Reuse.

For more details, please refer to the following article [Software Engineering – Reverse Engineering.](https://www.geeksforgeeks.org/software-engineering-reverse-engineering/)

**13. What is SRS?**

**Software Requirement Specification (SRS) Format** is a complete specification and description of requirements of the software that needs to be fulfilled for successful development of software system. These requirements can be functional as well as non-requirements depending upon the type of requirement. The interaction between different customers and contractors is done because it’s necessary to fully understand the needs of customers. For more details please refer[software requirement specification format article.](https://www.geeksforgeeks.org/software-requirement-specification-srs-format/)

**14. Distinguish between Alpha and Beta testing.**

| **Alpha Testing** | **Beta Testing** |
| --- | --- |
| Alpha testing involves both white box and black box testing. | Beta testing commonly uses black-box testing. |
| Alpha testing is performed by testers who are usually black,it -box internal employees of the organization. | Beta testing is performed by clients who are not part of the organization. |
| Alpha testing is performed at the developer’s site. | Beta testing is performed at the end-user,  the of the product. |
| Reliability and security testing are not checked in alpha testing. | Reliability, security, and robustness are checked during beta testing. |
| Alpha testing ensures the quality of the product before forwarding it to beta testing. | Beta testing also concentrates on the quality of the product but collects the user’s time-long input on the product and ensures that the product is ready for real-time users. |
| Alpha testing requires a testing environment or a lab. | Beta testing doesn’t require a testing environment or lab. |
| Alpha testing may require a real-time long execution cycle. | Beta testing requires only a few weeks of execution. |
| Developers can immediately address the critical issues or fixes in alpha testing. | Most of the issues or feedback collected from the beta testing will be implemented in future versions of the product |

For more details, please refer to the following article[Alpha Testing](https://www.geeksforgeeks.org/alpha-testing-software-testing/) and [Beta Testing](https://www.geeksforgeeks.org/beta-testing-software-testing/).

**15.  What are the elements to be considered in the System Model Construction?**

The type and size of the software, the experience of use for reference to predecessors, difficulty level to obtain users’ needs, development techniques and tools, the situation of the development team, development risks, the software development methods should be kept in mind. It is an important prerequisite to ensure the success of software development that designing a reasonable and suitable software development plan.

**16. What are CASE tools?**

CASE stands for Computer-Aided Software Engineering. CASE tools are a set of automated software application programs, which are used to support, accelerate and smoothen the SDLC activities.

**17.  What is the limitation of the RAD Model?**

* For large but scalable projects RAD requires sufficient human resources.
* Projects fail if developers and customers are not committed in a much-shortened time frame.
* Problematic if a system cannot be modularized

For more details, please refer to the following article [Software Engineering – Rapid Application Development Model (RAD)](https://www.geeksforgeeks.org/software-engineering-rapid-application-development-model-rad/).

**18.  What is the disadvantage of the spiral model?**

* Can be a costly model to use.
* Risk analysis requires highly specific expertise.
* The project’s success is highly dependent on the risk analysis phase.
* Doesn’t work well for smaller projects

For more details, please refer to the following article [Software Engineering – Spiral Model](https://www.geeksforgeeks.org/software-engineering-spiral-model/).

**19.  What is COCOMO model?**

A COCOMO model stands for Constructive Cost Model. As with all estimation models, it requires sizing information and accepts it in three forms:

* Object points
* Function points
* Lines of source code

For more details, please refer to the following article [Software Engineering – COCOMO Model](https://www.geeksforgeeks.org/software-engineering-cocomo-model/).

**20. Define an estimation of software development effort for organic software in the basic COCOMO model?**

Estimation of software development effort for organic software in the basic COCOMO model is defined as

Organic: Effort = 2.4(KLOC) 1.05 PM

**21. What is the Agile software development model?**

The agile SDLC model is a combination of iterative and incremental process models with a focus on process adaptability and customer satisfaction by rapid delivery of working software products. Agile Methods break the product into small incremental builds. Every iteration involves cross-functional teams working simultaneously on various areas like planning, requirements analysis, design, coding, unit testing, and acceptance testing.

**Advantages:**

* Customer satisfaction by rapid, continuous delivery of useful software.
* Customers, developers, and testers constantly interact with each other.
* Close, daily cooperation between business people and developers.
* Continuous attention to technical excellence and good design.
* Regular adaptation to changing circumstances.
* Even late changes in requirements are welcomed.

For more details, please refer to the following article [Software Engineering – Agile Development Models](https://www.geeksforgeeks.org/software-engineering-agile-development-models/).

**22. Which model can be selected if the user is involved in all the phases of SDLC?**

RAD model can be selected if the user is involved in all the phases of SDLC.

**23. What are software project estimation techniques available?**

There are some software project estimation techniques available:

* PERT
* WBS
* Delphi method
* User case point

**24. What is level-0 DFD?**

The highest abstraction level is called Level 0 of DFD. It is also called context-level DFD. It portrays the entire information system as one diagram.

For more details, please refer to the following article [DFD](https://www.geeksforgeeks.org/what-is-dfddata-flow-diagram/).

**25. What is physical DFD?**

Physical DFD focuses on how the system is implemented. The next diagram to draw after creating a logical DFD is physical DFD. It explains the best method to implement the business activities of the system. Moreover, it involves the physical implementation of devices and files required for the business processes. In other words, physical DFD contains the implantation-related details such as hardware, people, and other external components required to run the business processes.

**26. What is the black hole concept in DFD?**

A block hole concept in the data flow diagram can be defined as “A processing step may have input flows but no output flows”.In a black hole, data can only store inbound flows.

**27. Mention the formula to calculate the Cyclomatic complexity of a program?**

The formula to calculate the cyclomatic complexity of a program is:

e = number of edges

n = number of vertices

p = predicates

For more details, please refer to the following article [Cyclomatic Complexity](https://www.geeksforgeeks.org/cyclomatic-complexity/).

**28. What is software re-engineering?**

It is a process of software development that is done to improve the maintainability of a software system.

**29. How to find the size of a software product?**

Estimation of the size of the software is an essential part of Software Project Management. It helps the project manager to further predict the effort and time which will be needed to build the project. Various measures are used in project size estimation. Some of these are:

* Lines of Code
* Number of entities in ER diagram
* Total number of processes in detailed data flow diagram
* Function points

**30. Mentions some software analysis & design tools?**

* Data Flow Diagrams
* Structured Charts
* Structured English
* Data Dictionary
* Hierarchical Input Process Output diagrams
* Entity Relationship Diagrams and Decision tables

**31. What is the difference between Bug and Error?**

Bug: An Error found in the development environment before the product is shipped to the customer.  
Error: Deviation for actual and the expected/theoretical value.

**32. What is the difference between Risk and Uncertainty?**

* Risk is able to be measured while uncertainty is not able to be measured.
* Risk can be calculated while uncertainty can never be counted.
* You are capable of make earlier plans in order to avoid risk. It is impossible to make prior plans for the uncertainty.
* Certain sorts of empirical observations can help to understand the risk but on the other hand, the uncertainty can never be based on empirical observations.
* After making efforts, the risk is able to be converted into certainty. On the contrary, you can’t convert uncertainty into certainty.
* After making an estimate of the risk factor, a decision can be made but as the calculation of the uncertainty is not possible, hence no decision can be made.

**33. What is a use case diagram?**

A use case diagram is a behavior diagram and visualizes the observable interactions between actors and the system under development. The diagram consists of the system, the related use cases, and actors and relates these to each other:

* **System**: What is being described?
* **Actor**: Who is using the system?
* **Use Case**: What are the actors doing?

**34. Which model is used to check software reliability?**

A Rayleigh model is used to check software reliability. The Rayleigh model is a parametric model in the sense that it is based on a specific statistical distribution. When the parameters of the statistical distribution are estimated based on the data from a software project, projections about the defect rate of the project can be made based on the model.

**35. What is CMM?**

To determine an organization’s current state of process maturity, the SEI uses an assessment that results in a five-point grading scheme. The grading scheme determines compliance with a capability maturity model (CMM) that deﬁnes key activities required at different levels of process maturity. The SEI approach provides a measure of the global effectiveness of a company’s software engineering practices and establishes ﬁve process maturity levels that are deﬁned in the following manner:

* Level 1: Initial
* Level 2: Repeatable
* Level 3: Defined
* Level 4: Managed
* Level 5: Optimizing

**36. Define adaptive maintenance?**

Adaptive maintenance defines as modifications and updations when the customers need the product to run on new platforms, on new operating systems, or when they need the product to interface with new hardware and software.

**37. In the context of modular software design, which of the combination is considered for cohesion and coupling?**

In the context of modular software design, high cohesion, and low coupling is considered.

**38. What is regression testing?**

Regression testing is defined as a type of software testing that is used to confirm that recent changes to the program or code have not adversely affected existing functionality. Regression testing is just a selection of all or part of the test cases that have been run. These test cases are rerun to ensure that the existing functions work correctly. This test is performed to ensure that new code changes do not have side effects on existing functions. Ensures that after the last code changes are completed, the above code is still valid.

**39. Black box testing always focuses on which requirement of software?**

Black box testing always focuses on the functional requirements of the software.

**40. Which of the testing is used for fault simulation?**

With increased expectations for software component quality and the complexity of components, software developers are expected to perform effective testing. In today’s scenario, mutation testing has been used as a fault injection technique to measure test adequacy. Mutation Testing adopts “fault simulation mode”.

**41. What is a function point?**

Function point metrics provide a standardized method for measuring the various functions of a software application. Function point metrics, measure functionality from the user’s point of view, that is, on the basis of what the user requests and receives in return.

**42. What is a baseline?**

A baseline is a measurement that defines the completeness of a phase. After all activities associated with a particular phase are accomplished, the phase is complete and acts as a baseline for next phase.

**43. What is the cyclomatic complexity of a module that has 17 edges and 13 nodes?**

The cyclomatic complexity of a module that has seventeen edges and thirteen nodes = E – N + 2

E = Number of edges, N = Number of nodes

Cyclomatic complexity = 17 – 13 + 2 = 6

**44. A software does not wear out in the traditional sense of the term, but the software does tend to deteriorate as it evolves, why?**

The software does not wear out in the traditional sense of the term, but the software does tend to deteriorate as it evolves because  Multiple change requests introduce errors in component interactions.

**45. A cohesion is an extension of which concept?**

Cohesion refers to the degree to which Cohesion the elements inside a module belong together. is an extension of the information hiding concept.

**46. What are the three essential components of a software project plan?**

* Team structure,
* Quality assurance plans,
* Cost estimation.

**47. The testing of software against SRS is known as ….?**

The testing of software against SRS is known as acceptance testing.

**48. How to measure the complexity of software?**

To measure the complexity of software there are some methods in software engineering:

* Line of codes
* Cyclomatic complexity
* Class coupling
* Depth of inheritance

**49. Define the term WBS?**

The full form of WBS is Work Breakdown Structure. Its **Work Breakdown Structure** includes dividing a large and complex project into simpler, manageable, and independent tasks. For constructing a work breakdown structure, each node is recursively decomposed into smaller sub-activities, until at the leaf level, the activities become undividable and independent. A WBS works on a top-down approach. For more detail please refer [Work breakdown structure](https://www.geeksforgeeks.org/software-engineering-work-breakdown-structure/) article.

**50. A regression testing primarily related to which testing?**

Regression testing is primarily related to Maintenance testing.

Software Engineering Interview Questions

A list of frequently asked **Software Engineering Interview Questions** and Answers are given below.

1) What is Software Engineering?

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

[Click here for more information](https://www.javatpoint.com/software-engineering-tutorial)

2) What are the elements to be considered in the System Model Construction?

Elements to be considered in the System Model Construction are:

* Assumption
* Simplification
* Limitation
* Constraints
* Preferences

3) What does a System Engineering Model accomplish?

System Engineering Model accomplishes the following:

* Define Processes that serve needs of view
* Represent behavior of process and assumption
* Explicitly define Exogenous and Endogenous Input
* It represents all Linkages that enable an engineer to understand aspect better.

4) Define Framework.

A framework is the Code Skeleton that can be fleshed out with particular classes or functionality and designed to address the specific problem at hand.

5) 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.

6) 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.

7) What are the challenges in software?

The challenges in the software are:

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

8) Define Software process.

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

[Click here for more information](https://www.javatpoint.com/software-processes)

9) What are the internal milestones?

They are the significant and quantifiable attributes of progress. They are the standard methods in the project which provide that we are on the right track. They are under the authority of the project manager.

10) What is the limitation of RAD Model?

Limitation of RAD Model are:

* It requires a sufficient number of Human Resources to create enough number of teams.
* Developers and Users are not committed,the system fails.
* It is not Properly Modularized building component may be Problematic.
* It is not applicable when there is more possibility for Technical Risk.

[Click here for more information](https://www.javatpoint.com/software-engineering-rapid-application-development-model)

11) What are the disadvantages of classic life cycle model?

Disadvantages of the classic life cycle model are:

* Real projects rarely follow the sequential flow. Iteration always occurs and creates a problem.
* Challenging for the customer to state all requirements.
* The working version of the program is not available. So the customer must have patience.

12) What are the merits of the incremental model?

The merits of the incremental model are:

* The incremental model can be accepted when there is less number of people include in the project.
* Technical risks can be handle with each increment.
* For a minimal period, at least the core product can be delivered to the user.

13) What is the disadvantage of the spiral model?

The disadvantage of the spiral model are:

1. It is based on user communication. If the interface is not proper, then the software product which gets created will not be the up to the mark.
2. It demands a vast risk assessment. If the risk assessment is completed correctly, then only the successful product can be obtained.

14) Name the Evolutionary process Models.

Evolutionary powers models are:

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

15) Define Software Prototyping.

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

16) What are the benefits of prototyping?

The benefits of prototyping are:

* Prototype services as a basis for developing system specification.
* Design quality can be revised.
* The system can be managed easily.
* Development efforts may get decreased.
* System usability can be upgraded.

17) What are the prototyping methods in software process?

The prototyping methods in the software process are:

* **Evolutionary prototyping:** In this method of system development, the initial prototype is arranged, and it is then precise through the number of phases to the final stage.
* **Throw-away prototyping:** Using this method, a rough practical implementation of the system is produced. The requirement issues can be identified from this implementation. It is then rejected. System is then developed using some various engineering paradigm.

18) What are the advantages of evolutionary prototyping?

The advantages of evolutionary prototyping are:

* Fast delivery of the working system.
* User is contained while developing the system.
* The more useful system can be delivered.
* Specification, design and implementation work in equivalent manner.

19) What are the various Rapid prototyping techniques?

The various rapid prototyping techniques are:

* Dynamic high-level language development.
* Database programming.
* Component and application assembly.

20) What are the uses of User-Interface Prototyping?

This prototyping is used to pre-specify the looks and effectively feel of customer interface.

21) What is the principle of the prototype model?

A prototype is built to quickly determine to the user what the product would look like. The only minimal functionality of the actual product is supported during the prototyping phase.

22) Define System Context Diagram (SCD)?

System Context Diagram (SCD):

* Establish data boundary between System being implemented and Environment in which system operates.
* Describes all external producers, external consumers, and entities that communicate through the customer interface.

23) Define Quality Function Deployment (QFD)?

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

24) What is Requirement Engineering?

Requirement engineering is the process of establishing services which the user required from the system and constraint under which it operates and is developed.

[Click here for more information](https://www.javatpoint.com/software-engineering-requirement-engineering)

25) What is ERD?

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

26) What is DFD?

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

27) What is a state transition diagram?

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

28) What is Software Quality Assurance?

Software Quality Assurance is a set of auditing and documenting functions that assess the effectiveness and completeness of quality control activities.

[Click here for more information](https://www.javatpoint.com/software-quality-assurance)

29) What is the use of CMM?

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

30) What is coupling?

Coupling is the significant measure of the degree to which classes are linked to one another. Coupling should be kept as low as possible.

31) What is cohesion?

Cohesion is the indication of the relative functional strength of a module. It is a natural extension of Information Hiding and Performs a single task, requiring little integration with other components.

[Click here for more information](https://www.javatpoint.com/software-engineering-coupling-and-cohesion)

32) Define Refactoring.

Refactoring means changing a software system in a way that does not alter the external behavior of code.

33) What is Software Architecture?

Software Architecture means the overall structure of the software and how that software provides conceptual integrity for the system.

34) Define Stamp coupling.

When a portion of the data structure is passed via the module interface, then it is called as stamp coupling.

35) Define common coupling.

When several modules reference a global data area, then the coupling is called common coupling.

36) Define temporal cohesion.

When a module contains tasks that are related by the fact that all must be executed within the same period, then it is termed as temporal cohesion.

37) Define metrics.

Metrics are defined as the degree to which a system component or process possesses a given attribute.

38) What is COCOMO model?

Constructive Cost Model is a cost model, which gives the estimate of several staff-months it will take to develop the software product.

[Click here for more information](https://www.javatpoint.com/cocomo-model)

39) What is the purpose of the timeline chart?

The objective of the timeline chart is to emphasize the scope of the individual task. Hence set of functions are given as input to the timeline chart.

40) Define Smoke Testing?

Smoke testing is Integration Testing and frequently used when software products are being developed.

41) What are the benefits of Smoke Testing?

Benefits of doing Smoke Testing are:

* Integration Risk is minimized.
* Quality of end-product is improved.
* Error diagnosis and Correction are simplified.
* Progress is easy to assess.

42) What is Equivalence Partition?

Equivalence Partitions Derives an input domain of a program into classes of data from which test cases are derived. It is a Set of Objects have linked by relationships as Symmetric, Transitive, and Reflexive an equivalence class is present.

43) What are the steps followed in testing?

The steps followed in testing are:

* **Unit testing:** The individual elements are tested in this type of testing.
* **Module testing:** Related group of independent items is tested.
* **Sub-system testing:** This is a type of integration testing. Different modules are integrated into a sub-system, and the entire subsystem is tested.
* **System testing:** The entire system is tested in this system.
* **Acceptance testing:** This type of testing contains testing of the system with user data if the system behaves as per client need, then it is accepted.

44) Distinguish between Alpha and Beta testing.

Alpha and Beta testings are the two types of acceptance testing.

* **Alpha test:** The alpha testing is attesting in which the customer tests the version of complete software under the supervision of the developer. This testing is implement at the developer's site.
* **Beta test:** The beta testing is a testing in which the customer tests the version of the software without the developer being present. This testing is performed at the customer's site.

45) What are the types of Static Testing tools?

There are the three types of static testing tools.

* **Code-based testing tools:** These tools take source code as input and generate test cases.
* **Specialized testing tools:** Using this language, the detailed test specification can be written for each test case.
* **Requirement-based testing tools:** These tools help in designing as per user requirements.

46) Define maintenance.

Maintenance is described as the process in which changes are implemented by either modifying the existing system?s architecture or by adding new components to the system.

[Click here for more information](https://www.javatpoint.com/software-engineering-software-maintenance)

47) What are the types of software maintenance?

Types of software maintenance are:

**Corrective Maintenance:** It means the maintenance for correcting the software faults.

**Adaptive maintenance:** It means maintenance for adapting the change in environment.

**Perfective maintenance:** It means modifying or enhancing the system to meet the new requirements.

**Preventive maintenance:** It means changes made to improve future maintainability.

[Click here for more information](https://www.javatpoint.com/software-engineering-causes-of-software-maintenance-problems)

48) What is CASE Tools?

CASE Tools stands for Computer-Aided Software Engineering. It is system software that provides automated support for software process activities. It contains program used to support software process operations such as Requirement Analysis, System Modeling. Debugging and Testing.

[Click here for more information](https://www.javatpoint.com/software-engineering-case-tools-for-software-metrics)

49) What is Risk management?

Risk management is the phase of anticipating hurdles in carrying out the original plan and providing alternate methods so that the impact on the anticipated initially outcome is minimal.

1.

Mention some challenges of software development.

This is an important software engineering interview question that is often asked in interviews. Some of the software development challenges are:

* The rapid advancement of technology: Technology evolves at a phenomenal pace, thus, exerting pressure on software developers to learn and adapt to maintain competitiveness.
* Increasing demands of customers: Different customers have different needs. Software developers must understand the business concept to understand the software requirements of the customers.
* Time constraints: Software developers need to work under strict time deadlines. For remote software developers, the challenge is even bigger as they straddle timezones. Time constraints can reduce the efficiency of software developers and end in mediocre quality software products.
* Limited resources: Many software development companies still struggle with a lack of IT resources such as high-performing software development tools, robust computing platforms, incompetent data storage, and inefficient network capabilities. All these can reduce the overall efficiency and performance of the software development team.
* Disagreements with software testing teams: Software development and software testing teams are often at loggerheads with each other. Major reasons for the same are the difference in team mindsets and job roles, the opposite nature of development and testing, and high work and performance pressure.
* Coping with legacy systems: Many organizations still run with older legacy systems and this can raise the risk of security, inefficiency, and lack of technological compatibility. Thus, the modernization of legacy software is a pressing need.
* Heterogeneity challenge: Since customer demands evolve constantly, an advanced challenge in software development is to create flexible and dependable software that can adapt to changing and future needs.

2.

Mention the limitations of the RAD model.

The limitations of the RAD model are:

* This model has a sufficiently high requirement of human resources to build the required number of teams.
* The system does not work if the developers and users don’t stay committed.
* If the system is not correctly modularized, it may lead to problems in building components.
* If the technical risk is high, this model is not applicable.

When answering software development interview questions about the limitations of certain models, you can also talk about times when you faced such limitations and how you coped with them.

3.

What are the steps involved in software development?

The following steps are needed for software development:

* Brainstorming and planning: This stage determines the purpose and functions for which the software program must be developed.
* Project team analysis: This stage determines the team members, tools, and time frame required for the project.
* Design stage: At this stage, the designated software development team designs the program.
* Development stage: This is the stage where the final functionality of the software is developed.
* Quality assurance and testing: This is the penultimate, but perhaps the most important stage before finally launching the program. Here, any bugs, defects, or errors are identified and rectified.
* Launch: This is when the program is finally installed and launched for the intended users.

4.

Mention the benefits of Smoke-testing.

Some of the benefits of Smoke-testing are listed below:

* It minimizes the risk of integration
* It improves the end-product quality
* It simplifies the diagnosis and correction of errors
* It makes assessing progress easy

5.

Talk about equivalence partitioning.

Equivalence partitioning is a method for testing software where the input data of a software unit is divided into partitions containing equivalent data. These partitions help in deriving the test cases, which, in principle, must cover each partition at least once. Through partitioning, the test cases can detect classes of errors, and hence reduce the number of test cases to be developed. Since the number of test cases reduces, the time required for software testing reduces too. Though this method is usually applied to a tested component’s inputs, in some rare cases it could also be applied to the outputs.

6.

Differentiate between alpha and beta testing.

These are two types of acceptance tests.

* Alpha test: Alpha tests are tests where the customer assesses a full-version software under the developer’s supervision. This kind of testing happens at the developer’s site.
* Beta test: Beta tests are tests where the customer assesses a version of the software without the developer’s supervision. This kind of testing happens at the customer’s site.

7.

Mention the types of static testing tools.

The three types of static testing tools are:

* Code-based: For these testing tools, the source code functions as the input and they use that to generate the test cases.
* Specialized: In these testing tools, the language is used to write detailed test specifications for every test case.
* Requirement-based: These testing tools design the test cases as per the user requirements.

For such a question in the software developer interview, remember to add instances from personal experience.

8.

Mention the different types of software maintenance.

The different types of software maintenance are as follows:

* Corrective: Corrective maintenance is used for correcting the faults in the software.
* Adaptive: Adaptive maintenance helps the software to adapt to changes in its environment.
* Perfective: Perfective maintenance helps in changing or improving the system to meet the new requirements.
* Preventive: Preventive maintenance refers to the alterations made to enhance future maintainability.

9.

What are the software engineering categories?

The categories of software engineering include:

* Scientific software: This software provides an engineering solution for specific engineering tasks. Examples include CAM and CAD software, AutoCAD, etc.
* Embedded software: This refers to software used in control and instrumentation appliances such as televisions, satellites, and washing machines.
* AI software: These types of software are designed for pattern recognition, advanced systems, and artificial neural software used in machine learning algorithms and natural language processing.
* Networking and web development software: This provides a medium in communication between computer systems, software, and data storage systems. Examples include web browsers, email clients, and file transfer protocol (FTP) software.
* System software: This category includes software that provides the core functionality for other digital applications. Examples include operating systems like Windows and Linux, device drivers, and utility programs.

10.

How would you define a framework?

A framework is a platform for developing and deploying software. It outlines how software components relate to one another and details the development process. It is a structured and standardized set of tools, libraries, and components that provide a foundation for building software applications.

Frameworks are designed to facilitate the development, maintenance, and scalability of applications by offering a pre-built structure for organizing code, handling common tasks, and interacting with other systems.

11.

Differentiate between computer software and a computer program.

A computer program is a subset of computer software. A computer program contains programming instructions to carry out tasks, while computer software is a collection of computer programs.

Computer software might contain programs, libraries, APIs, components, and frameworks. It is a broad term encompassing any software designed to run on a computer system, including operating systems, application software, utilities, device drivers, and system softwares.

On the other hand, a computer program is a set of instructions written in a specific programming language and executed by a computer. It can be a single application or a group of related applications that perform a specific function or set of functions.

12.

What is the full meaning of SDLC?

SDLC stands for Software Development Life Cycle. It is a structured approach to software development that involves various stages and activities, from planning and analysis to implementation and maintenance. The process enables the creation of high-quality software within the shortest possible time frame.

13.

List different SDLC models.

Software Development Life Cycle (SDLC) is a planned methodology of building software to ensure quality and reliability in the software built. Here is a list of some popular SDLC models:

Waterfall model: In this SDLC model, the entire process is divided into various phases where the outcome of one phase becomes the input of the next phase.

Incremental model: It is a series of separate SDLC cycles where requirements are grouped and addressed in each cycle.

V-model: It plans the phases in parallel with verification phases on the left side and validation phases on the right side of the V-shaped diagram.

Agile Model: [Agile development model](https://www.turing.com/blog/agile-software-development-lifecycle/) promotes continuous collaboration between development and testing teams throughout the development process.

14.

What is debugging?

Debugging is the process of identifying and resolving defects, errors, and faults in software applications. A bug is an error or fault in a software application that causes it to behave unexpectedly or incorrectly. Debugging begins soon after the software code is written and continues through the subsequent stages of software development. The goal is to create a defect-free or error-free software product for market release.

15.

What is the Waterfall method?

The Waterfall approach is a linear and consecutive downward development SDLC approach. In this methodology, the customer and stakeholder requirements are collected at the beginning of the project.

After that, a sequential project plan covers all those requirements. And, hence the Waterfall method is so named as each phase of the project downpours into the next like a waterfall.

16.

What are some use cases of SDLC?

SDLC is useful in various scenarios, including:

* Developing new software applications or systems from scratch
* Updating or modifying existing software
* Maintaining software and ensuring its longevity
* Ensuring software quality and reducing the risk of defects
* Managing and controlling project timelines and costs

17.

What is software engineering?

Software engineering is a branch of engineering that focuses on developing and maintaining high-quality software systems. It involves applying engineering principles and methods to design, develop, test, deploy, and maintain software. Software engineering plays a vital role in the current digital age, where software systems are a ubiquitous part of our daily lives, from smartphones to cars, healthcare systems to financial services.

Software engineering involves a range of activities, such as requirements gathering, software design, coding, testing, documentation, and project management. Effective software engineering involves not only technical skills but also collaboration, communication, and problem-solving. Software engineers work with other professionals such as product managers, designers, and quality assurance teams to deliver high-quality software systems that meet user needs.

18.

What are the targeted accomplishments of a software engineering model?

A software engineering model is designed to achieve several objectives, including:

Providing a framework for software development: A model helps in creating a systematic approach to software development, which can be used to plan, design, develop, test and maintain software.

Enhancing software quality: A software engineering model helps in identifying and addressing software quality issues early in the development process, which helps in reducing the cost of development and maintenance, and ensures that the software meets the desired standards of quality.

Improving communication: A model provides a common language and understanding between software developers, managers, and stakeholders, which helps in improving communication and collaboration throughout the development process.

Facilitating project management: A model helps in managing the development process by providing a clear roadmap of activities, roles, and responsibilities, which helps in ensuring that the project is completed on time, within budget, and with the desired quality.

Enabling software reuse: An engineering mode also promotes the reuse of software components, which helps in reducing the time and cost of development and ensures consistency and quality across projects.

19.

What are internal milestones in software engineering?

Internal milestones are measurable and important attributes of processes. They are standard systematic procedures that indicate that the engineers are on the right path. These milestones can be used to assess the development team's progress, identify issues and risks, and make necessary adjustments to the project plan. They can be related to various aspects of the project, such as completing a specific feature, testing and debugging the code, or reaching a certain level of performance or functionality.

20.

Mention some characteristics of software.

Software exhibits various characteristics that make it a unique product in the market. Some of the most notable features include tailor-made design to fit specific user requirements, usability, functionality, maintainability, and reliability. These characteristics are essential to ensure that the software performs as intended, is user-friendly, can be easily updated and repaired, and functions without failure or errors.

21.

What is a software process?

A software process refers to a set of systematic activities and procedures that enable the development and maintenance of software applications. The key activities typically included in a software process are requirements gathering, design, implementation, testing, and maintenance. These activities provide a framework for managing the entire software development life cycle.The software process can be customized to suit the specific requirements of a project or organization, and it can be refined and improved over time to optimize performance and efficiency.

22.

What are the factors to be taken into account during the construction of a system model?

System models are useful tools for understanding complex systems and identifying areas for improvement. When constructing a system model, there are several elements to consider, including:

* Preferences: This refer to the goals or objectives that the system is designed to achieve.
* Assumptions: These are the underlying beliefs or ideas about how the system works.
* Limitations: These factors can constrain the system, such as time or resources.
* Constraints: These are the specific conditions that the system must operate under, such as legal or regulatory requirements.

By considering these elements, developers and stakeholders can ensure that the model accurately reflects the system and can be used to make informed decisions about improvements or changes.

23.

List some project management tools used in software engineering.

Software engineering project management involves a variety of tools, including Jira, Trello, Asana, and more. These tools offer features such as task assignment, progress tracking, Gantt charts, team collaboration, and resource allocation. Other useful project management tools for software engineering include Agile methodologies, Scrum boards, and Kanban boards.

24.

What is the Rapid Application Development (RAD) model?

The RAD model is an iterative and prototyping approach to software development that emphasizes less time spent on traditional planning phases and more on software development. It involves a series of smaller development cycles, including planning, design, construction, and testing, with a focus on quick delivery of functional software to clients and stakeholders. The RAD model incorporates feedback from end-users and stakeholders to refine and improve the software.

25.

What are the different phases of SDLC?

The Software Development Life Cycle (SDLC) consists of several phases that depend on the project's requirements, but the typical phases are:

Requirement Gathering: Gathering information from stakeholders and defining the scope of the project.

System Analysis and Design: Analyzing requirements and designing a system architecture that meets those requirements.

Coding: Implementing the design by writing code in a specific programming language.

Testing: Testing the software to ensure that it works as expected and meets the requirements.

Deployment and Maintenance: Deploying the software to production and maintaining it by fixing bugs, adding new features, and upgrading it over time.

26.

What are some disadvantages of the classic life cycle model, also known as the Waterfall model?

The Waterfall model may not be the best choice for long or continuous projects due to the lack of certainty and high amount of risks involved. It may also present challenges for dynamic or complex projects, as well as those that are object-oriented.

27.

Mention a few disadvantages of the incremental process model.

Some of the disadvantages include:

Complexity: As the number of increments increases, managing and coordinating multiple increments can become challenging.

Lack of overall planning: Since the project is divided into increments, the overall planning may be less detailed than other models. This may lead to difficulty in identifying dependencies and conflicts between increments.

Cost: The incremental model can be expensive as it requires constant testing and integration of different increments.

Time-consuming: The incremental model can be time-consuming as each increment requires design, development, testing, and integration phases. This can lead to longer project timelines.

Difficulty in accommodating changes: Changes to requirements or design in one increment can have a significant impact on subsequent increments, making it difficult to accommodate changes.

28.

List some advantages of the incremental process model.

The advantages of the incremental process model include the following:

Incremental resource deployment: Instead of deploying all resources at once, they can be gradually introduced as the project progresses, reducing the risk of resource waste.

Low initial delivery cost: Since each iteration focuses on delivering a specific functionality, the initial delivery cost is relatively low

Ease of error recognition: With incremental development, errors are easier to recognize since each iteration is focused on a specific functionality. This means that developers can identify and address errors early on in the development process, reducing the risk of critical issues arising later on.

Offers more flexibility: By delivering working software in increments, developers can make changes as necessary, enabling them to respond quickly to changing requirements and stakeholder feedback.

Easily tested and debugged: Each iteration is focused on a specific functionality, making it easier to test and debug.

29.

What are some disadvantages of the spiral model?

The spiral model is a software development process that involves repeated cycles of risk analysis, design, and development. Some disadvantages of this model are:

Costly: The spiral model can be expensive to implement due to the need for a high level of expertise in risk analysis and the potential for additional iterations.

Expertise required: The risk analysis involved in this model requires a high level of expertise, which can be challenging to find.

Not suited for small projects: The spiral model is best suited for large and complex projects, making it less suitable for smaller projects.

Critical risk analysis: The project's success depends heavily on the accuracy and effectiveness of the risk analysis, making it a critical factor for the success of the project.

30.

What is a software scope?

In software development, the term software scope refers to the extent and boundaries of the software's features and functionalities that are defined during the planning stage of the project. It involves identifying what the software should do and what it should not do, which is critical to ensuring that the software meets the business requirements and user expectations.

The software scope includes various elements such as the project's budget, resource allocation, and time allocation.

31.

What type of data is passed through the HTTP headers?

HTTP headers allow the client and server to pass additional information through the request or response. This information can include authentication credentials, data type specifications, cache instructions, connection details, and information about the client and server. Non-standard headers can also be used to pass custom data as needed

32.

What are some advantages of evolutionary prototyping?

The advantages of this type of prototyping include the following:

Early and continuous feedback: With each iteration, stakeholders can provide feedback on the prototype, allowing the system to evolve and meet the project needs better.

Better risk management: The system is developed incrementally, so potential risks can be identified early and addressed before they become major issues.

Flexibility: The system can adapt to changing requirements, as each iteration allows for modifications and enhancements to be made.

Time and cost-effectiveness: The system can be developed and tested in smaller increments, reducing the overall development time and cost.

33.

What are some disadvantages of evolutionary prototyping?

The disadvantages of evolutionary prototyping include the following:

Time-consuming: Evolutionary prototyping can be a time-consuming process because it involves multiple iterations, testing, and refinement.

Lack of structure: This approach does not have a well-defined structure, which can lead to the development of a system that lacks coherence and consistency.

Cost: Developing multiple prototypes can be expensive, particularly if the development team does not have a clear understanding of the project requirements.

Scope creep: The iterative nature of evolutionary prototyping can lead to scope creep, where additional features are added to the software without considering their impact on the overall project goals and timeline.

34.

What is software prototyping?

Prototyping is an iterative process in which developers create a preliminary software application version to test its functionality, user interface, and other features. The prototype can gather feedback from stakeholders, identify potential design flaws, and refine the software development process.

35.

How can project execution be measured?

Project execution can be measured by milestone checklists, activity monitoring, and status reports. These methods help project managers track the project's progress and identify any potential issues that may arise during the project execution. Additionally, they provide a clear picture of the project's status and allow stakeholders to make informed decisions based on the information provided by these methods.

36.

What are some evolutionary process models?

Some of evolutionary models include:

* Prototype model
* Spiral model
* Incremental model
* Concurrent model

37.

What are the different techniques in rapid prototyping?

The different types of rapid prototyping techniques include:

* Mockups & wireframes
* Automatic code generation
* 3D printing
* Rapid Application Development (RAD)

38.

What is coupling?

Coupling in software engineering defines interdependence between two or more software modules. It shows how closely related or connected two modules are. High coupling can make a software system more difficult to maintain and modify, while low coupling can make it more modular, flexible, and easier to work with.

39.

What is cohesion?

Cohesion indicates the bond strength between elements in a module. It measures the relationship strength between the data of a class/methods and the unifying factor or purpose the class serves. High cohesion means that the elements within a module are closely related and work together towards a common goal, while low cohesion implies that the elements are loosely related and do not have a clear or unified purpose.

40.

What is Agile SDLC?

Agile software development life cycle is an iterative process of software development based on an adaptive and decision-making approach, using continuous design improvements to produce better software.

41.

What are the different types of coupling in software engineering?

Listed below are the different types of coupling in software engineering:

* Stamp
* Data
* Content
* Common
* External
* Control

42.

What are the different types of cohesion in software engineering?

The following are the different types of cohesion in software engineering:

* Sequential
* Functional
* Communicational
* Temporal
* Procedural
* Logical
* Coincidental

43.

What are some advantages of cohesion?

Some of the advantages of cohesion include:

* Reduces complexity of modules
* Increases module reusability
* Improves system maintainability
* Improves testabilty
* Creates balance between coupling and unit complexity

44.

What is a COCOMO model?

This is an acronym for Constructive Cost Model. COCOMO model is a procedural regression model that calculates the estimated effort, time, and cost required to develop the software.The model considers various factors such as the size of the project, complexity, required software reliability, team experience, and development environment to provide a reliable estimate of the software development effort.

45.

How would you describe a state transition diagram?

A state transition diagram describes the various states an object can possess and the events under which such an object can transition. It consists of nodes, which represent the states, and directed edges, which represent the transitions. It is commonly used in software engineering to model the behavior of a software system or a specific module within a system.

46.

Explain refactoring.

Refactoring is the process of optimizing the software system without changing the software’s functionality or code behavior. It is typically done to improve the code's readability, reduce complexity, remove duplication, and increase the code's overall quality. Refactoring is a common practice in software development and is often done to improve a codebase's long-term maintainability and scalability.

47.

What is smoke testing?

Smoke testing is a preliminary software testing in which software is tested to reveal stability faults or failures. This is normally done to confirm the quality assurance team to proceed with the software testing. Smoke testing typically involves a set of simple and crucial tests that verify the basic features of the software, and any critical issues found during smoke testing are addressed before proceeding to more comprehensive testing.

48.

What are the differences between tags and branches?

A tag is a pointer to a specific commit in Git history. Tags are generally used to mark a specific release or version of the codebase. Tags are immutable, meaning that they cannot be changed once created.

On the other hand, a branch is a parallel line of development in Git that allows developers to work on new features or make changes to the codebase without affecting the main codebase. Branches are mutable, which means they can be updated and modified as new commits are added.

49.

In your opinion, which of testing frameworks is best for testing code quality?

There are several testing frameworks available that can help to test code quality, each with its own strengths and weaknesses. Here are some popular options:

JUnit: JUnit is a widely used testing framework for Java applications. It is designed to help developers write and run repeatable tests to ensure their code works as expected. JUnit can be used to test individual methods or entire classes, and it provides a variety of built-in assertion methods to help developers check the correctness of their code.

PyTest: PyTest is a popular testing framework for Python applications. It is designed to be simple and easy to use while also providing powerful features for testing complex systems. PyTest supports a wide range of testing scenarios, including unit testing, functional testing, and integration testing.

NUnit: NUnit is a testing framework for .NET applications. It provides various tools for testing different types of applications, including desktop applications, web applications, and mobile applications. NUnit supports a wide range of testing scenarios, including unit testing, integration testing, and acceptance testing.

Mocha: Mocha is a testing framework for JavaScript applications. It provides a simple, flexible syntax for writing tests and supports synchronous and asynchronous testing scenarios. Mocha can be used to test a variety of JavaScript frameworks and libraries, including Node.js, React, and Angular.

50.

What is software architecture?

[Software architecture](https://www.turing.com/blog/software-architecture-patterns-types/) is the design and structure of a software system, including its components, relationships, and principles. Examples of software architecture include layered, client-server, or microservices. It serves as a blueprint for the development, deployment, and maintenance of the system and is crucial for ensuring scalability, maintainability, and reliability.

51.

List the methods used for determining the size of a software product.

Some of the methods that can be used to determine the size of software are:

Function points: It is a method that measures the functionality provided by the software product based on user requirements.

Delivered code: It measures the size of the code delivered as part of the software product.

Lines of Code (LOC): This method involves counting the total number of lines of code written to develop the software product.

Story Points (SP): This method involves estimating the size of the software product based on the complexity of its features and user stories.

52.

What does the acronym CASE stand for?

CASE stands for Computer-Aided Software Engineering. It uses computer-based tools and techniques to aid in software development. These tools can assist developers in analyzing, designing, coding, testing, and maintaining software systems.

53.

What are function points?

Function points are important features that can be used to quantify the functionality provided by the software product. Function points are calculated based on various factors, such as inputs, outputs, inquiries, files, and interfaces, which are assessed to determine the size and complexity of the software.

54.

How would you differentiate between Queue and Stack?

The main difference between a queue and a stack is their principle of operation. A queue is a data structure that works on the principle of First-In-First-Out (FIFO), which means that the element that is inserted first is the first one to be removed. On the other hand, a stack is a data structure that works on the principle of Last-In-First-Out (LIFO), which means that the element that is inserted last is the first one to be removed.

In a queue, elements are inserted at the rear end and removed from the front end. In contrast, in a stack, elements are inserted at the top and removed from the top. Another key difference is that queues are used for breadth-first search and sequential processing, whereas stacks are used for depth-first search, recursive programming, and backtracking.

Tired of interviewing candidates to find the best developers?

Hire top vetted developers within 4 days.

[Hire Now](https://customers.turing.com/)

INTERMEDIATE SOFTWARE ENGINEERING INTERVIEW QUESTIONS AND ANSWERS

1.

Which of the SDLC models would you recommend as the best?

The best SDLC model is the Agile model because it combines functionalities of iterative and incremental models and is the most widely used one. The Agile SDLC model offers several advantages, including flexibility, faster time to market, optimized quality, and increased collaboration between teams.

However, here it is worth noting that it’s important to evaluate the specific needs of each project and organization and choose the SDLC model that best fits those needs.

2.

What are the different phases of the Waterfall model?

The Waterfall model is a linear, sequential approach to software development that is divided into distinct phases. The phases of the Waterfall model are as follows:

* Requirements gathering and analysis
* Design
* Implementation
* Testing and integration
* Deployment
* Maintenance

3.

What is Quality Function Deployment (QFD)?

Quality Function Deployment (QFD) is a structured approach that helps organizations interpret customer requirements and translate them into engineering specifications. QFD ensures that the product design and development align with the customer's needs, resulting in a product that is more likely to satisfy the customer and succeed in the market.

4.

What would you consider to be some benefits of software prototyping?

Some of the benefits are listed below:

* It allows easy management of the system, design, and final product
* It offers a platform for creating system specifications
* It increases the system usability
* It reviews and improves the product quality

5.

How would you define change control?

Change control is a systematic approach to managing software changes and configurations. It is a critical function that ensures all changes to software follow organizational guidelines and rules. The process typically involves identifying proposed changes, evaluating their impact, obtaining approvals, implementing them, and monitoring their effects.

The ultimate goal of change control is to guarantee consistency, control, and transparency in software changes, while minimizing disruption to systems and maintaining their integrity and reliability.

6.

What equation would you use to calculate the cyclomatic complexity of a program?

The cyclomatic complexity can be calculated using the graph theory’s formula: V(G) = e – n + 2 of the Cyclomatic Complexity Theorem. Here "e" is the number of edges in the program's control flow graph, and "n" is the number of nodes in the graph.

7.

What are some prototyping methods?

There are four main prototyping methods, including:

Extreme prototyping: This type of prototyping is mostly used for web applications and contains three distinct stages, namely, low, mid, and high-fidelity models.

Throwaway prototyping: This method intends to build the prototype into something other than a working system. Thus, it is discarded after use.

Incremental prototyping: This method creates several prototypes, which are then assembled into a final working system once proven to satisfy consumers’ needs.

Evolutionary prototyping: As the name indicates, this prototype is built to scale, evolve, and be refined as the development progresses.

8.

What are blocking calls?

Blocking calls, also known as synchronous calls, refer to situations in which the execution of a program is paused until a certain operation is completed. In the context of JavaScript, this means that the code execution is blocked until a non-JavaScript operation, such as a network request or a file I/O operation, is finished.

Blocking calls can have significant performance implications, as they can cause the entire application to freeze while waiting for the operation to complete. To avoid this, developers can use asynchronous programming techniques, such as callbacks or promises, which allow the code to continue executing while waiting for the operation to complete.

9.

What do you understand about functional programming?

[Functional programming](https://www.turing.com/kb/introduction-to-functional-programming) is a programming paradigm that uses functions to solve problems. In functional programming, functions can be passed as arguments to other functions, returned as values from functions, and stored in data structures.

One of the key characteristics of functional programming is the emphasis on immutability, meaning that once a value is defined, it cannot be changed. Instead of changing existing values, functional programming typically creates new values based on existing ones.

10.

What are non-blocking calls?

Non-blocking calls are asynchronous operations that allow programs to continue executing without waiting for a response from a particular function or task. By freeing up resources and allowing for parallel execution, non-blocking calls improve the responsiveness and performance of applications. This approach is widely used in modern programming languages to optimize system utilization and deliver seamless user experiences.

11.

How would you define a system context diagram?

A system context diagram is a high-level view of a system that defines its boundaries and indicates the entities that interact with it. It is a visual representation in software engineering that provides an overview of the entire system and is similar to a block diagram.

12.

What are CASE tools, and list some examples?

CASE (Computer-Aided Software Engineering) tools are software applications used to automate and improve various activities involved in the Software Development Life Cycle (SDLC). They assist in various phases of software development such as analysis, design, coding, testing, and maintenance.

Some examples of CASE tools are:

* Software design tools (Rational Rose, Visual Paradigm, Enterprise Architect.)
* Requirement analysis tools (CaliberRM, DOORS, RequisitePro)
* Code generation tools (CodeSmith, CodeCharge Studio, JCodeModel)
* Integrated development environment (IDE) (Eclipse, Visual Studio, NetBeans)

13.

In software, what are functional and non-functional requirements?

Functional requirements refer to the features and standards required of the software by the users. These features are expected to be a part of the final product and they must be incorporated into the software during the development process.

The non-functional requirements, on the other hand, are concerned with the quality constraints and checks that a software system must satisfy. The non-functional requirements are associated with user interface, security, performance, etc.

14.

What is ERD, and how would you describe it?

ERD stands for Entity-Relationship Diagram. An ERD is a visual representation of the relationships between different entities or objects within a system or domain. It is primarily used in database design to model the relationships between tables and their attributes.

15.

What is the primary use of User Interface prototyping?

User Interface (UI) prototyping is primarily used to create a visual representation of a software product's user interface design. It allows designers and developers to create a mockup or working model of the user interface that can be tested and refined before the final product is developed.

By creating a user interface prototype, designers and developers can gain valuable insights into how users interact with the system and identify potential usability issues early in the development process. This, in turn, helps to ensure that the final product is more user-friendly and meets the needs of its intended audience.

16.

What is requirement engineering?

Requirement engineering or RE can be defined as defining, documenting, and maintaining the requirements in the software engineering process. It involves understanding the needs and expectations of stakeholders, translating them into functional and non-functional requirements, and ensuring that the software system meets those requirements throughout its lifecycle.

17.

Define the purpose of a timeline chart.

A timeline chart serves the purpose of visually explaining the sequence or process of events of a project to get insights into the essence of that project. It allows for a quick understanding of the order of events and their relationship to one another.

A timeline chart is often used as a [project management tool](https://www.turing.com/blog/project-management-tools-every-organization-needs-in-2022/) to track progress and deadlines, and it can also be used for educational or informational purposes.

18.

What is an SRS?

SRS stands for Software Requirement Specification and it helps users determine if the software meets their needs. It is a representation of the software assessed by the users.

The SRS is typically created at the beginning of the software development process and serves as a guide for the rest of the development process. The SRS describes the software system's features, functionalities, and characteristics in detail, including user interfaces, inputs, outputs, and performance requirements.

19.

Kindly explain the bottom-up and top-down design models.

Bottom-up and top-down are two common design models used in software engineering. The bottom-up approach involves starting with specific and smaller components and then gradually combining them to create larger systems. For example, a software engineer might start by building and testing individual functions or modules before integrating them into a complete system.

On the other hand, the top-down approach starts with a generalized view and then breaks it down into smaller, more specific components. In this model, the design begins with a high-level overview of the system's goals and functions, and then gradually drills down into the details. For instance, a software engineer might start by identifying the overall functions that a system needs to perform, and then break them down into smaller tasks and subtasks.

20.

What is the major difference between object-oriented design (OOD) and component-oriented design (COD)?

The major difference between OOD and COD is that OOD splits complex systems into smaller, manageable units and can be re-operated with lesser complexity. However, COD deals with the decomposition process to create sections or software components, creating independent modules which establish various functionalities across the software system.

21.

Describe the testing process.

There are five basic steps involved in testing, and they are:

* Unit testing: Testing of individual elements
* Module testing: Testing of a similar cluster of independent elements
* Sub-system testing: Different modules are integrated and tested as a sub-system in this step.
* System testing: Testing of the entire system
* Acceptance testing: This involves the testing of the system to see if it conforms to the user’s requirements

22.

In software development, what is a baseline?

In software development, a baseline is a milestone that indicates the completion of singular or multiple software deliverables. This helps to regulate vulnerability that can spiral the project out of control or increase damage. Baselines can include things like code, documentation, and other aspects and are often used to assess progress, track changes, and manage version control.

23.

How can you differentiate between validation and verification?

Validation is the process of determining if a software project meets the specified quality standards.

On the other hand, verification is the process that ensures that a software product meets its stated objectives in terms of functionalities.

24.

What is the difference between fixed website design and fluid website design?

Fixed website designs use fixed pixel widths for easy launching and running but are less user-friendly. Fixed website designs have a set width that remains the same regardless of the size of the screen or browser window. This means that the design may appear differently on different screen sizes or resolutions, and users may have to scroll horizontally to view the content on smaller screens.

However, fluid websites utilize percentages as relative indicators for widths. This allows the content to expand or contract to fit the screen, creating a more flexible and user-friendly experience. However, designing a fluid layout can be more challenging and requires careful consideration of the content and how it will adjust to different screen sizes.

25.

Differentiate between functional and object-oriented programming.

In functional programming, the primary focus is on functions, which are mathematical mappings between inputs and outputs. In this paradigm, functions can be passed around as arguments to other functions, returned as values, and stored in variables. This allows for a highly modular and composable style of programming.

In contrast, [OOP](https://www.turing.com/kb/object-oriented-programming-help-the-developers-to-code-better) focuses on objects and instances of classes that encapsulate data and behavior. This allows for the natural modeling of real-world entities and their interactions. Objects communicate with each other by sending messages, which invoke methods defined in their respective classes.

Functional programming supports Python, Javascript, Scala, and Haskell. The languages supported by object-oriented programming include Python, Java, C++, Lisp, and Perl.

Tired of interviewing candidates to find the best developers?

Hire top vetted developers within 4 days.

[Hire Now](https://customers.turing.com/)

ADVANCED SOFTWARE ENGINEERING INTERVIEW QUESTIONS AND ANSWERS

1.

What are software metrics?

Software metrics are quantifiable measures used to evaluate specific aspects of software development. They are numerical measurements used to assess the quality, efficiency, and effectiveness of software development processes and products. Some common examples of software metrics include code complexity, code coverage, and defect density. By measuring these metrics, software development teams can identify areas for improvement, monitor progress, and make data-driven decisions to optimize their development processes and products.

2.

What are some benefits of software metrics?

Software metrics offer several benefits and a few are listed below:

* Workload reduction
* Cost reduction
* Increase in ROI
* Identification of improvement areas

3.

What are the DORA research program's key metrics to measure software performance?

The DORA research program, which stands for DevOps Research and Assessment, has defined several metrics to measure software delivery performance. These metrics include Mean Time To Restore (MTTR), Deployment Frequency, Change Lead Time, Change Failure Rate, Test Automation Percentage, and Change Fail Percentage among others.

4.

How would you differentiate between black box testing and white box testing?

Black box testing is a high-level testing method that involves testing the software without knowledge of the internal workings of the program. The goal of black box testing is to ensure that the software meets its functional and non-functional requirements.

On the other hand, white box testing is a testing method that involves testing the internal workings of the software or its architecture. White box testing is typically carried out by developers who have knowledge of the code and its implementation. The goal of white box testing is to identify defects in the code and to ensure that the software is working as expected at the code level.

Overall, black box testing focuses on the behavior of the software while white box testing focuses on the implementation of the software. Both methods are important for ensuring that software is of high quality and meets the desired requirements.

Learn more about [black box testing vs. white box testing](https://www.turing.com/blog/black-box-testing-vs-white-box-testing-understanding-key-differences/).

5.

What is the need for concurrency in software, and how is it done?

Concurrency is a concept that allows the software to carry out multiple tasks simultaneously and seamlessly. Programming languages such as Java and C++ achieve concurrency through multithreading. However, other programming languages such as Python and JavaScript also support concurrency and have their own techniques for achieving it, such as asyncio and web workers.

6.

How would you define Software Configuration Management (SCM)?

Software Configuration Management (SCM) is the process of identifying, organizing, and controlling changes to software and related documents throughout the software development lifecycle. SCM ensures that changes to code and related documents are tracked, reviewed, and approved by the appropriate stakeholders to prevent errors and maintain consistency across versions. SCM plays a critical role in software development by providing a systematic approach to managing changes to software and related documentation.

7.

Differentiate between Quality Assurance and Quality Control in software engineering.

Commonly referred to as QA and QC, [QA](https://www.turing.com/blog/software-quality-assurance-and-its-importance/) is a preventive process of ensuring that the end software product meets the required standards, and it applies to the full software development cycle. It involves activities such as requirement analysis, design review, code review, and process improvement.

On the other hand, QC refers to the corrective process of ensuring that a software production process meets the required standards to produce the required products. This is employed in the testing phase. The primary goal of QC is to verify that the final product meets the quality standards and requirements set by the stakeholders.

8.

Why is modularization important in software engineering?

Modularization is a method used to divide a software system into different discrete modules for their independent operations. Modularization is a crucial aspect of software engineering as it helps achieve better code organization, maintainability, and reusability.

When software is broken down into smaller modules, each module can be designed, tested, and maintained independently. This helps in identifying and fixing bugs easier, improving code quality, and reducing development time.

9.

What does 0-level data flow diagram refer to?

The 0-level data flow diagram (DFD), also known as context diagram, is the highest level of abstraction. All the inputs and outputs of the system are shown in this diagram, along with the various entities that interact with the system.

10.

To prevent software from getting into trouble, what process model will you use?

I will use the clean room engineering model because it detects and eliminates defects before the software worsens. The clean room engineering model is one approach to software development that emphasizes rigorous testing and verification to ensure the correctness and reliability of the software.

Other process models that can also help prevent software issues include agile, waterfall, iterative, and incremental development.

11.

Software analysis and design tools are important in software development. What software analysis and design tools would you recommend?

The following software analysis and design tools are highly recommended:

* Data dictionary
* Structured charts
* Data flow diagrams
* Hierarchical Input Process Output diagrams
* Entity Relationship Diagrams and Decision Tables
* Use case diagrams
* Unified Modeling Language (UML)

12.

How would you differentiate between an EXE and DLL?

EXE is a program that can be executed. This means an EXE file contains encoded steps of instructions that can be executed when a user clicks on the file icon. A DLL, on the other hand, is a code library that different programs can dynamically use simultaneously.

13.

What are strong and weak typing, and which would you recommend?

Weak typing checks the types of variables in a system at run time, while strong typing checks the types of variables at compilation time. Strong typing is recommended because it reduces bugs. However, it can also make the code more rigid and harder to modify. Eventually, the choice depends on your project requirements.

14.

Why do you think using catch (exception) is a bad idea?

I think it is a bad idea because of these reasons:

* It cannot be used when exception types are not known
* Because if there are no variables defined, the exception is hard to read
* It can lead to unexpected behavior if the code is modified later and new exceptions are introduced, as they may also be caught by the catch (exception) block and handled inappropriately.

15.

Why do we use polymorphism in software engineering?

Polymorphism allows objects of different classes to be treated as if they are objects of the same class. This means that the same method can be called on different objects, and each object will respond in a way that is appropriate to its class. This enables code to be more flexible, reusable, and easier to maintain.

16.

How would you describe a responsive web design?

Responsive web designs are just websites with responsive designs that use media queries to focus and aim at breakpoints that scale images, adjust the layout, and wrap texts such that websites can easily adjust to fit any screen size.

17.

Would you choose a microservice approach or a monolithic one for app development? State the reason(s) for the answer.

I would choose a microservice approach. This is because building an app with a microservice approach gives it a combination of various independent services that can act robustly without one another. This gives the app faster performance, higher flexibility, and more efficiency.

18.

What are web workers in HTML5, and why do they matter in software engineering?

Web workers are JavaScripts that run independently in the background of an HTML5 page without affecting the page's performance.

They matter in software engineering because it helps in multithreading simulation in JavaScript. This means that they enable the simultaneous run of different scripts. This is particularly useful for handling computationally-intensive tasks, such as processing large amounts of data, without causing the page to freeze or become unresponsive.

19.

Have you worked with microservices architecture? If so, can you describe the challenges you faced and how you overcame them?

Yes, I got a chance to work with microservices architecture in my previous role. One of the major challenges we faced was managing the communication between microservices. As the system grew in complexity, we had to ensure that the microservices could communicate with each other effectively and efficiently.

To overcome this challenge, we implemented an API gateway, which acted as a single entry point for all requests to the microservices. This helped us manage the communication between microservices by routing requests and responses between them. We also implemented a service registry, which allowed us to keep track of all the microservices in the system and their endpoints.

Another challenge we faced was ensuring the scalability of the system. As the number of microservices grew, we had to ensure that the system could handle the increased load. To address this, we implemented containerization using Docker, which allowed us to isolate each microservice and scale them independently.

20.

Explain the big-O notation in the simplest words.

The big-O in software engineering is used to explain the complexity of an algorithm. The big-O notation analyzes the efficiency of an algorithm as the input tends to the largest possible input size by explaining the worst-case scenario. It can be used to describe the time for execution or memory space occupied by the algorithm.

21.

If a non-technical person asked you the meaning of an API, how would you explain it?

Firstly, API is an acronym for [Application Programming Interface](https://www.turing.com/kb/application-programming-interface). It is simply an interface that allows two programs or systems to communicate. It does the job of taking the request from one system to another and then delivering the response. It enables developers to access the functionality of another system or application without having to understand how it works internally.