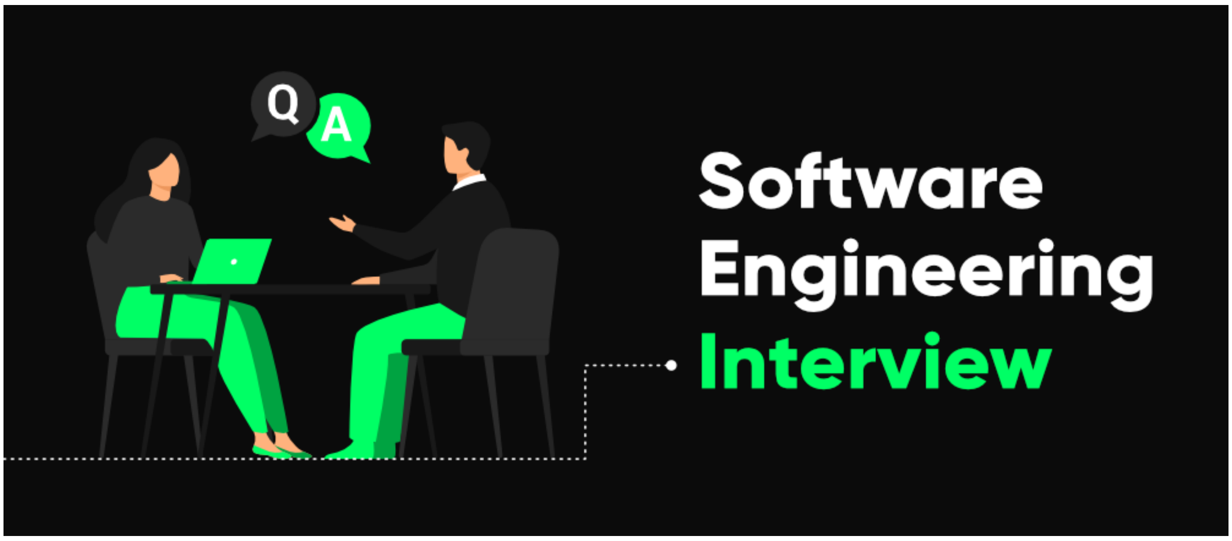
Top 50 Software Engineering Interview Questions and Answers

varshachoudhary

Software Engineering is indeed a must-to-go field for every individual who aspires to **make a successful career as a Software Engineer**, **Software Developer,** etc. in the IT industry. In simple words, it is concerned with the systematic and comprehensive study of designing, development, operations, and maintenance of a software system. In tech interviews of almost every renowned tech company, recruiters asked various questions from Software Engineering concepts such as Software Development Models & Architecture, Software Project Management (SPM), Testing and Debugging, etc. to assess the candidates. Hence, you must be prepared for all such **Software Engineering Interview Questions** to ace the interview.



We know that Software Engineering is a vast field in itself and to find out & prepare for all the important concepts or questions for interviews is not an easy job. So, to make it easier and convenient for you, here, we're providing you with an extensive list of **Commonly Asked Software Engineering Interview Questions** that are often asked by the recruiters. Do check out all these questions from below:

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?.

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.

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.

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.

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).

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.

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.

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.

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	Validation
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.

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.

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.

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	Most of the issues or feedback collected from the beta testing will be implemented in future versions of the

Alpha Testing	Beta Testing
testing.	product

For more details, please refer to the following article Alpha Testing and Beta 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).

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.

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.

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.

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.

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:

$c = e - n + 2p$

```
e = number of edges
n = number of vertices
p = predicates
```

For more details, please refer to the following article 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 article.

50. A regression testing primarily related to which testing?

Regression testing is primarily related to Maintenance testing.