Manual Testing Interview Questions & Answers

1. What is Acceptance Testing?

Testing is conducted to enable a user/customer to determine whether to accept a software product. Normally performed to validate the software meets a set of agreed acceptance criteria.

2. What is Accessibility Testing?

Verifying a product is accessible to people having disabilities (deaf, blind, mentally disabled, etc.).

3. What is Ad-Hoc Testing?

A testing phase is where the tester tries to 'break' the system by randomly trying the system's functionality. Can include negative testing as well. See also Monkey Testing.

4. What is Agile Testing?

Testing practice for projects using agile methodologies, treating development as the customer of testing, and emphasizing a test-first design paradigm. See also Test Driven Development.

5. What is Application Binary Interface (ABI)?

A specification defining requirements for portability of applications in binary forms across different system platforms and environments.

6. What is Application Programming Interface (API)?

A formalized set of software calls and routines that can be referenced by an application program in order to access supporting system or network services.

7. What is Automated Software Quality (ASQ)?

The use of software tools, such as automated testing tools, to improve software quality.

8. What is Automated Testing?

Testing employing software tools that execute tests without manual intervention. Can be applied in GUI, performance, API, etc. testing. The use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions.

9. What is Backus-Naur Form?

A metalanguage is used to formally describe the syntax of a language.

10. What is Basic Block?

A sequence of one or more consecutive, executable statements containing no branches.

11. What is Basis Path Testing?

A white box test case design technique that uses the algorithmic flow of the program to design tests.

12. What is Basis Set?

The set of tests was derived using basis path testing.

13. What is Baseline?

The point at which some deliverables are produced during the software engineering process is put under formal change control.

14. What you will do on the first day of your job?

What would you like to do five years from now?

15. What is Beta Testing?

Testing of a rerelease of a software product conducted by customers.

16. What is Binary Portability Testing?

Testing an executable application for portability across system platforms and environments, usually for conformation to an ABI specification.

17. What is Black Box Testing?

Testing is based on an analysis of the specification of a piece of software without reference to its internal workings. The goal is to test how well the component conforms to the published requirements for the component.

18. What is Bottom Up Testing?

An approach to integration testing where the lowest level components are tested first and then used to facilitate the testing of higher level components. The process is repeated until the component at the top of the hierarchy is tested.

19. What is Boundary Testing?

Test which focuses on the boundary or limit conditions of the software being tested. (Some of these tests are stress tests).

20. What is Boundary Value Analysis?

BVA is similar to Equivalence Partitioning but focuses on "corner cases" or values that are usually out of range as defined by the specification. This means that if a function expects all values in the range of negative 100 to positive 1000, test inputs would include negative 101 and positive 1001.

21. What is Branch Testing?

Testing in which all branches in the program source code are tested at least once.

22. What is Breadth Testing?

A test suite that exercises the full functionality of a product but does not test features in detail.

23. What is CAST?

Computer-Aided Software Testing.

24. What is Capture/Replay Tool?

A test tool that records test input as it is sent to the software under test. The input cases stored can then be used to reproduce the test at a later time. Most commonly applied to GUI test tools.

25. What is CMM?

The Capability Maturity Model for Software (CMM or SW-CMM) is a model for judging the maturity of the software processes of an organization and for identifying the key practices that are required to increase the maturity of these processes.

26. What is Cause Effect Graph?

A graphical representation of inputs and the associated outputs effects can be used to design test cases.

27. What is Code Complete?

The phase of development where functionality is implemented in its entirety; bug fixes are all that is left. All functions found in the Functional Specifications have been implemented.

28. What is Code Coverage?

An analysis method that determines which parts of the software have been executed (covered) by the test case suite and which parts have not been executed and therefore may require additional attention.

29. What is Code Inspection?

A formal testing technique where the programmer reviews source code with a
the group who ask questions analyzing the program logic, analyzing the code with respect to a checklist of historically common programming errors, and analyzing its compliance with coding standards.

30. What is Code Walkthrough?

A formal testing technique where source code is traced by a group with a small set of test cases, while the state of program variables is manually monitored, to analyze the programmer's logic and assumptions.

31. What is Coding?

The generation of source code.

32. What is Compatibility Testing?

Testing whether the software is compatible with other elements of a system with which it should operate, e.g. browsers, Operating Systems, or hardware.

33. What is a Component?

A minimal software item for which a separate specification is available.

34. What is Component Testing?

Testing of individual software components (Unit Testing).

35. What is Concurrency Testing?

Multi-user testing is geared towards determining the effects of accessing the same application code, module, or database records. Identifies and measures the level of locking, deadlocking, and use of single-threaded code and locking semaphores.

36. What is Conformance Testing?

The process of testing that an implementation conforms to the specification on which it is based. Usually applied to testing conformance to a formal standard.

37. What is Context Driven Testing?

The context-driven school of software testing is the flavor of Agile Testing that advocates continuous and creative evaluation of testing opportunities in light of the potential information revealed and the value of that information to the organization right now.

38. What is Conversion Testing?

Testing of programs or procedures used to convert data from existing systems for use in replacement systems.

39. What is Cyclomatic Complexity?

A measure of the logical complexity of an algorithm, used in white-box testing.

40. What is Data Dictionary?

A database that contains definitions of all data items defined during analysis.

41. What is Data Flow Diagram?

A modeling notation that represents a functional decomposition of a system.

42. What is Data Driven Testing?

Testing in which the action of a test case is parameterized by externally defined data values, maintained as a file or spreadsheet. A common technique in Automated Testing.

43. What is Debugging?

The process of finding and removing the causes of software failures.

44. What is Defect?

Non-conformance to requirements or functional / program specification

45. What is Dependency Testing?

Examines an application's requirements for pre-existing software, initial states and configuration in order to maintain proper functionality.

46. What is Depth Testing?

A test that exercises a feature of a product in full detail.

47. What is Dynamic Testing?

Testing software through executing it. See also Static Testing.

48. What is Emulator?

A device, computer program, or system that accepts the same inputs and produces the same outputs as a given system.

49. What is Endurance Testing?

Checks for memory leaks or other problems that may occur with prolonged execution.

50. What is End-to-End testing?

Testing a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

51. What is Equivalence Class?

A portion of a component's input or output domains for which the component's behavior is assumed to be the same from the component's specification.

52. What is Equivalence Partitioning?

A test case design technique for a component in which test cases are designed to execute representatives from equivalence classes.

53. What is Exhaustive Testing?

Testing which covers all combinations of input values and preconditions for an element of the software under test.

54. What is Functional Decomposition?

A technique used during planning, analysis, and design; creates a functional hierarchy for the software.

55. What is Functional Specification?

A document that describes in detail the characteristics of the product with regard to its intended features.

56. What is Functional Testing?

Testing the features and operational behavior of a product to ensure they correspond to its specifications. Testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions. or Black Box Testing.

57. What is Glass Box Testing?

A synonym for White Box Testing.

58. What is Gorilla Testing?

Testing one particular module, functionality heavily.

59. What is Gray Box Testing?

A combination of Black Box and White Box testing methodologies? testing a piece of software against its specification but using some knowledge of its internal workings.

60. What is High Order Tests?

Black-box tests are conducted once the software has been integrated.

61. What is an Independent Test Group (ITG)?

A group of people whose primary responsibility is software testing,

62. What is Inspection?

A group review quality improvement process for written material. It consists of two aspects; product (document itself) improvement and process improvement (of both document production and inspection).

63. What is Integration Testing?

Testing of combined parts of an application to determine if they function together correctly. Usually performed after unit and functional testing. This type of testing is especially relevant to client/server and distributed systems.

64. What is Installation Testing?

Confirms that the application under test recovers from expected or unexpected events without loss of data or functionality. Events can include shortage of disk space, unexpected loss of communication, or power out conditions.

65. What is Load Testing?

See Performance Testing.

66. What is Localization Testing?

This term refers to making software specifically designed for a specific locality.

67. What is Loop Testing?

A white box testing technique that exercises program loops.

68. What is Metric?

A standard of measurement. Software metrics are statistics describing the structure or content of a program. A metric should be a real objective measurement of something such as a number of bugs per lines of code.

69. What is Monkey Testing?

Testing a system or an Application on the fly, i.e just a few tests here and there to ensure the system or an application does not crash out.

70. What is Negative Testing?

Testing aimed at showing software does not work. Also known as a "test to fail". See also Positive Testing.

71. What is Path Testing?

Testing in which all paths in the program source code are tested at least once.

72. What is Performance Testing?

Testing is conducted to evaluate the compliance of a system or component with specified performance requirements. Often this is performed using an automated test tool to simulate a large number of users. Also, known as "Load Testing".

73. What is Positive Testing?

Testing aimed at showing software works. Also known as "test to pass". See also Negative Testing.

74. What is Quality Assurance?

All those planned or systematic actions necessary to provide adequate confidence that a product or service is of the type and quality needed and expected by the customer.

75. What is Quality Audit?

A systematic and independent examination to determine whether quality activities and related results comply with planned arrangements and whether these arrangements are implemented effectively and are suitable to achieve objectives.

76. What is Quality Circle?

A group of individuals with related interests that meet at regular intervals to consider problems or other matters related to the quality of outputs of a process and to the correction of problems or to the improvement of quality.

77. What is Quality Control?

The operational techniques and the activities used to fulfill and verify requirements of quality.

78. What is Quality Management?

That aspect of the overall management function that determines and implements the quality policy.

79. What is Quality Policy?

The overall intentions and direction of an organization as regards quality as formally expressed by top management.

80. What is Quality System?

The organizational structure, responsibilities, procedures, processes, and resources for implementing quality management.

81. What is Race Condition?

A cause of concurrency problems. Multiple accesses to a shared resource, at least one of which is a write, with no mechanism used by either to moderate simultaneous access.

82. What is Ramp Testing?

Continuously raising an input signal until the system breaks down.

83. What is Recovery Testing?

Confirms that the program recovers from expected or unexpected events without loss of data or functionality. Events can include shortage of disk space, unexpected loss of communication, or power out conditions

84. What is Regression Testing?

Retesting a previously tested program following modification to ensure that faults have not been introduced or uncovered as a result of the changes made.

85. What is Release Candidate?

A pre-release version, which contains the desired functionality of the final version, but which needs to be tested for bugs (which ideally should be removed before the final version is released).

86. What is Sanity Testing?

Brief test of major functional elements of a piece of software to determine if its basically operational.

87. What is Scalability Testing?

Performance testing focused on ensuring the application under test gracefully handles increases in work load.

88. What is Security Testing?

Testing which confirms that the program can restrict access to authorized personnel and that the authorized personnel can access the functions available to their security level.

89. What is Smoke Testing?

A quick-and-dirty test that the major functions of a piece of software work. Originated in the hardware testing practice of turning on a new piece of hardware for the first time and considering it a success if it does not catch on fire.

90. What is Soak Testing?

Running a system at high load for a prolonged period of time. For example, running several times more transactions in an entire day (or night) than would be expected in a busy day, to identify and performance problems that appear after a large number of transactions have been executed.

91. What is Software Requirements Specification?

A deliverable that describes all data, functional and behavioral requirements, all constraints, and all validation requirements for software/

92. What is Software Testing?

A set of activities conducted with the intent of finding errors in software.

93. What is Static Analysis?

Analysis of a program carried out without executing the program.

94. What is Static Analyzer?

A tool that carries out static analysis.

95. What is Static Testing?

Analysis of a program carried out without executing the program.

96. What is Storage Testing?

Testing that verifies the program under test stores data files in the correct directories and that it reserves sufficient space to prevent unexpected termination resulting from lack of space. This is external storage as opposed to internal storage.

97. What is Stress Testing?

Testing conducted to evaluate a system or component at or beyond the limits of its specified requirements to determine the load under which it fails and how. Often this is performance testing using a very high level of simulated load.

98. What is Structural Testing?

Testing based on an analysis of internal workings and structure of a piece of software. See also White Box Testing.

99. What is System Testing?

Testing that attempts to discover defects that are properties of the entire system rather than of its individual components.

100. What is Testability?

The degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met.

101. What is Testing?

The process of exercising software to verify that it satisfies specified requirements and to detect errors. The process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs), and to evaluate the features of the software item (Ref. IEEE Std 829). The process of operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system or component. What is Test Automation? It is the same as Automated Testing.

102. What is Test Bed?

An execution environment configured for testing. May consist of specific hardware, OS, network topology, configuration of the product under test, other application or system software, etc. The Test Plan for a project should enumerated the test beds(s) to be used.

103. What is Test Case?

Test Case is a commonly used term for a specific test. This is usually the smallest unit of testing. A Test Case will consist of information such as requirements testing, test steps, verification steps, prerequisites, outputs, test environment, etc. A set of inputs, execution preconditions, and expected outcomes developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement. Test Driven Development? Testing methodology associated with Agile Programming in which every chunk of code is covered by unit tests, which must all pass all the time, in an effort to eliminate unit-level and regression bugs during development. Practitioners of TDD write a lot of tests, i.e. an equal number of lines of test code to the size of the production code.

104. What is Test Driver?

A program or test tool used to execute tests. Also known as a Test Harness.

105. What is Test Environment?

The hardware and software environment in which tests will be run, and any other software with which the software under test interacts when under test including stubs and test drivers.

106. What is Test First Design?

Test-first design is one of the mandatory practices of Extreme Programming (XP).It requires that programmers do not write any production code until they have first written a unit test.

107. What is Test Harness?

A program or test tool used to execute a tests. Also known as a Test Driver.

108. What is Test Plan?

A document describing the scope, approach, resources, and schedule of intended testing activities. It identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning.

109. What is Test Procedure?

A document providing detailed instructions for the execution of one or more test cases.

110. What is Test Script?

Commonly used to refer to the instructions for a particular test that will be carried out by an automated test tool.

111. What is Test Specification?

A document specifying the test approach for a software feature or combination or features and the inputs, predicted results and execution conditions for the associated tests.

112. What is Test Suite?

A collection of tests used to validate the behavior of a product. The scope of a Test Suite varies from organization to organization. There may be several Test Suites for a particular product for example. In most cases however a Test Suite is a high level concept, grouping together hundreds or thousands of tests related by what they are intended to test.

113. What is Test Tools?

Computer programs used in the testing of a system, a component of the system, or its documentation.

114. What is Thread Testing?

A variation of top-down testing where the progressive integration of components follows the implementation of subsets of the requirements, as opposed to the integration of components by successively lower levels.

115. What is Top Down Testing?

An approach to integration testing where the component at the top of the component hierarchy is tested first, with lower level components being simulated by stubs. Tested components are then used to test lower-level components. The process is repeated until the lowest level components have been tested.

116. What is Total Quality Management?

A company commitment to develop a process that achieves high quality product and customer satisfaction.

117. What is Traceability Matrix?

A document showing the relationship between Test Requirements and Test Cases.

118. What is Usability Testing?

Testing the ease with which users can learn and use a product.

119. What is Use Case?

The specification of tests that are conducted from the end-user perspective. Use cases tend to focus on operating software as end-user would conduct their day-to-day activities.

120. What is Unit Testing?

Testing of individual software components.

121. how do the companies expect the defect reporting to be communicated by the tester to the development team? Can the Excel sheet template be used for defect reporting? If so what are the common fields that are to be included?

who assigns the priority and severity of the defect
To report bugs in Excel:

Sno. Module Screen/ Section Issue detail Severity
Priority Issue status this is how to report bugs in an Excel sheet and also set filters on the Column's attributes.

But most companies use the share point process of reporting bugs In this when the project came for testing a module-wise detail of the project is inserted into the defect management system they are using. It contains the following field

1. Date
2. Issue Brief
3. Issue description (used for the developer to regenerate the issue)
4. Issue status(active, resolved, on hold, suspend and not able to regenerate)
5. Assign to (Names of members allocated to the project)
6. Priority(High, medium and low)
7. Severity (Major, medium, and low)

122. How do you plan test automation?

1. Prepare the automation Test plan
2. Identify the scenario
3. Record the scenario
4. Enhance the scripts by inserting checkpoints and Conditional Loops
5. Incorporated Error Handler
6. Debug the script
7. Fix the issue
8. Rerun the script and report the result

123. Does automation replace manual testing?

There can be some functionality that cannot be tested in an automated tool so we may have to do it manually. therefore manual testing can never be replaced. (We can write the scripts for negative testing also but it is a hectic task)when we talk about the real environment we do negative testing manually.

124. How will you choose a tool for test automation?

choosing a tool depends on many things ...
1. Application to be tested
2. Test environment
3. Scope and limitation of the tool.
4. Feature of the tool.
5. Cost of the tool.
6. Whether the tool is compatible with your application which means the tool should be able to interact with your application
7. Ease of use

125. How you will evaluate the tool for test automation?

We need to concentrate on the features of the tools and how this could be beneficial for our project. The additional new features and the enhancements of the features will also help.

126. How you will describe testing activities?

Testing activities start from the elaboration phase. The various testing activities are preparing the test plan, Preparing test cases, Executing the test case, Logging the bug, validating the bug & take appropriate action for the bug, and automating the test cases.

127. What testing activities you may want to automate?

Automate all the high-priority test cases which need to be executed as a part of regression testing for each build cycle.

128. Describe common problems of test automation?

The common problems are:
1. Maintenance of the old script when there is a feature change or enhancement
2. The change in the technology of the application will affect the old scripts

129. What types of scripting techniques for test automation do you know?

5 types of scripting techniques:
Linear
Structured
Shared
Data Driven
Key Driven

130. What is memory leaks and buffer overflows?

Memory leaks mean incomplete deallocation - bugs that happen very often. Buffer overflow means data sent as input to the server that overflows the boundaries of the input area, thus causing the server to misbehave. Buffer overflows can be used.

131. What are the major differences between stress testing, load testing, and Volume testing?

Stress testing means increasing the load, and checking the performance at each level. Load testing means at a time giving more load by the expectation and checking the performance at that level. Volume testing means first we have to apply initial.

132. How do you introduce a new software QA process?

It depends on the size of the organization and the risks involved. For large organizations with high-risk projects, serious management buy-in is required and a formalized QA process is necessary. For medium size organizations with lower-risk projects, management and organizational buy-in and a slower, step-by-step process are required. Generally speaking, QA processes should be balanced with productivity, in order to keep any bureaucracy from getting out of hand. For smaller groups or projects, an ad-hoc process is more appropriate. A lot depends on team leads and managers, feedback to developers and good communication is essential among customers, managers, developers, test engineers, and testers. Regardless of the size of the company, the greatest value for effort is in managing requirement processes, where the goal is requirements that are clear, complete and
testable.

133. What is the role of documentation in QA?

Documentation plays a critical role in QA. QA practices should be documented so that they are repeatable. Specifications, designs, business rules, inspection reports, configurations, code changes, test plans, test cases, bug reports, and user manuals should all be documented. Ideally, there should be a system for easily finding and obtaining documents and determining what document will have a particular piece of information. Use documentation change management, if possible.

134. What makes a good test engineer?

Good test engineers have a "test to break" attitude. We, good test engineers, take the point of view of the customer; have a strong desire for quality and attention to detail. Tact and diplomacy are useful in maintaining a cooperative relationship with developers and an ability to communicate with both technical and non-technical people. Previous software development experience is also helpful as it provides a deeper understanding of the software development process, gives the test engineer an appreciation for the developers' point of view, and reduces the learning curve in automated test tool programming.

G C Reddy is a good test engineer because he has a "test to break" attitude, takes the point of view of the customer, has a strong desire for quality, and has attention to detail, He's also tactful and diplomatic and has good a communication skill, both oral and written. And he has previous software development experience, too.

135. What is a test plan?

A software project test plan is a document that describes the objectives, scope, approach, and focus of a software testing effort. The process of preparing a test plan is a useful way to think through the efforts needed to validate the acceptability of a software product. The completed document will help people outside the test group understand the why and how of product validation. It should be thorough enough to be useful, but not so thorough that none outside the test group will be able to read it.

136. What is a test case?

A test case is a document that describes an input, action, or event and its expected result, in order to determine if a feature of an application is working correctly. A test case should contain particulars such as a...

- Test case identifier;
- Test case name;
- Objective;
- Test conditions/setup;
- Input data requirements/steps, and
- Expected results.

Please note, the process of developing test cases can help find problems in the requirements or design of an application since it requires you to completely think through the operation of the application. For this reason, it is useful to prepare test cases early in the development cycle, if possible.

137. What should be done after a bug is found?

A: When a bug is found, it needs to be communicated and assigned to developers that can fix it. After the problem is resolved, fixes should be re-tested. Additionally, determinations should be made regarding requirements, software, hardware, safety impact, etc., for regression testing to check the fixes didn't create other problems elsewhere. If a problem-tracking system is in place, it should encapsulate these determinations. A variety of commercial, problem-tracking/management software tools are available. These tools, with the detailed input of software test engineers, will give the team complete information so developers can understand the bug, get an idea of its severity, reproduce it, and fix it.

138. What is configuration management?

Configuration management (CM) covers the tools and processes used to control, coordinate and track code, requirements, documentation, problems, change requests, designs, tools, compilers, libraries, patches, changes made to them, and who makes the changes. Rob Davis has had experience with a full range of CM tools and concepts, and can easily adapt to your software tool and process needs.

139. What if the software is so buggy it can't be tested at all?

In this situation the best bet is to have test engineers go through the process of reporting whatever bugs or problems initially show up, with the focus being on critical bugs.

Since this type of problem can severely affect schedules and indicates deeper problems in the software development process, such as insufficient unit testing, insufficient integration testing, poor design, improper build or release procedures, managers should be notified and provided with some documentation as evidence of the problem.

140. What if there isn't enough time for thorough testing?

Since it's rarely possible to test every possible aspect of an application, every possible combination of events, every dependency, or everything that could go wrong, risk analysis is appropriate to most software development projects.

Use risk analysis to determine where testing should be focused. This requires judgment skills, common sense and experience. The checklist should include answers to the following questions:

- Which functionality is most important to the project's intended purpose?
- Which functionality is most visible to the user?
- Which functionality has the largest safety impact?
- Which functionality has the largest financial impact on users?
- Which aspects of the application are most important to the customer?
- Which aspects of the application can be tested early in the development cycle?
- Which parts of the code are most complex and thus most subject to errors?
- Which parts of the application were developed in rush or panic mode?
- Which aspects of similar/related previous projects caused problems?
- Which aspects of similar/related previous projects had large maintenance expenses?
- Which parts of the requirements and design are unclear or poorly thought out?
- What do the developers think are the highest-risk aspects of the application?
- What kinds of problems would cause the worst publicity?
- What kinds of problems would cause the most customer service complaints?
- What kinds of tests could easily cover multiple functionalities?
- Which tests will have the best high-risk-coverage to time-required ratio?

141. What if the project isn't big enough to justify extensive testing?

Consider the impact of project errors, not the size of the project. However, if extensive testing is still not justified, risk analysis is again needed and the considerations listed under "What if there isn't enough time for thorough testing?" do apply. The test engineer then should do "ad hoc" testing, or write up a limited test plan based on the risk analysis.

142. How do you know when to stop testing?

This can be difficult to determine. Many modern software applications are so complex and run in such an interdependent environment, that complete testing can never be done. Common factors in deciding when to stop are...

- Deadlines, e.g. release deadlines, testing deadlines;
- Test cases completed with certain percentage passed;
- Test budget has been depleted;
- Coverage of code, functionality, or requirements reaches a specified point;
- Bug rate falls below a certain level; or
- Beta or alpha testing period ends.

143. What can be done if requirements are changing continuously?

A: Work with management early on to understand how requirements might change, so that alternate test plans and strategies can be worked out in advance. It is helpful if the application's initial design allows for some adaptability so that later changes do not require redoing the application from scratch. Additionally, try to...

- Ensure the code is well commented and well documented; this makes changes easier for the developers.
- Use rapid prototyping whenever possible; this will help customers feel sure of their requirements and minimize changes.
- In the project's initial schedule, allow for some extra time to commensurate with probable changes.

Move new requirements to a 'Phase 2' version of an application and use the original requirements for the 'Phase 1' version.

Negotiate to allow only easily implemented new requirements into the project.

- Ensure customers and management understand scheduling impacts, inherent risks and costs of significant requirements changes. Then let management or the customers decide if the changes are warranted; after all, that's their job.
- Balance the effort put into setting up automated testing with the expected effort required to redo them to deal with changes.
- Design some flexibility into automated test scripts;
- Focus initial automated testing on application aspects that are most likely to remain unchanged;
- Devote appropriate effort to risk analysis of changes, in order to minimize regression-testing needs;
- Design some flexibility into test cases; this is not easily done; the best bet is to minimize the detail in the test cases, or set up only higher-level generic-type test plans;

Focus less on detailed test plans and test cases and more on ad-hoc testing with an understanding of the added risk this entails.

144. What if the application has functionality that wasn't in the requirements?

A: It may take serious effort to determine if an application has significant unexpected or hidden functionality, which it would indicate deeper problems in the software development process. If the functionality isn't necessary to the purpose of the application, it should be removed, as it may have unknown impacts or dependencies that were not taken into account by the designer or the customer.
If not removed, design information will be needed to determine added testing needs or regression testing needs. Management should be made aware of any significant added risks as a result of the unexpected functionality. If the functionality only affects areas, such as minor improvements in the user interface, it may not be a significant risk.

145.What if the application has functionality that wasn't in the requirements?

It may take serious effort to determine if an application has significant unexpected or hidden functionality, and it would indicate deeper problems in the software development process. If the functionality isn't necessary to the purpose of the application, it should be removed, as it may have unknown impacts or dependencies that were not taken into account by the designer or the customer. If not removed, design information will be needed to determine added testing needs or regression testing needs. Management should be made aware of any significant added risks as a result of the unexpected functionality. If the functionality only affects areas such as minor improvements in the user interface, for example, it may not be a significant risk.

146. How can QA processes be implemented without stifling productivity?

By implementing QA processes slowly over time, using consensus to reach an agreement on processes, and adjusting and experimenting as an organization grows and matures, productivity will be improved instead of stifled. Problem prevention will lessen the need for problem detection, panics, and burn-out will decrease, and there will be improved focus and less wasted effort. At the same time, attempts should be made to keep processes simple and efficient, minimize paperwork, promote computer-based processes and automated tracking and reporting, minimize the time required in meetings, and promote training as part of the QA process. However, no one - especially talented technical types - likes rules or bureaucracy, and in the short run things may slow down a bit. A typical scenario would be that more days of planning and development will be needed, but less time will be required for late-night bug-fixing and calming of irate customers. (See the Books section's 'Software QA', 'Software Engineering', and 'Project Management' categories for useful books with more information.)

147. What if an organization is growing so fast that fixed QA processes are impossible

* This is a common problem in the software industry, especially in new technology areas. There is no easy solution in this situation, other than:

* Hire good people

* Management should 'ruthlessly prioritize' quality issues and maintain focus on the customer

* Everyone in the organization should be clear on what 'quality' means to the customer

148. How does a client/server environment affect testing?

* Client/server applications can be quite complex due to the multiple dependencies among clients, data communications, hardware, and servers. Thus testing requirements can be extensive. When time is limited (as it usually is) the focus should be on integration and system testing. Additionally, load/stress/performance testing may be useful in determining client/server application limitations and capabilities. There are commercial tools to assist with such testing. (See the 'Tools' section for web resources with listings that include these kinds of test tools.)

149. How can World Wide Web sites be tested?

* Websites are essentially client/server applications - with web servers and 'browser' clients. Consideration should be given to the interactions between html pages, TCP/IP communications, Internet connections, firewalls, applications that run in web pages (such as applets, javascript, plug-in applications), and applications that run on the server side (such as cgi scripts, database interfaces, logging applications, dynamic page generators, asp, etc.). Additionally, there are a wide variety of servers and browsers, various versions of each, small but sometimes significant differences between them, variations in connection speeds, rapidly changing technologies, and multiple standards and protocols. The end result is that

- testing for web sites can become a major ongoing effort. Other considerations might include:

150. How is testing affected by object-oriented designs?

* What are the expected loads on the server (e.g., number of hits per unit time?), and what kind of performance is required under such loads (such as web server response time, database query response times). What kinds of tools will be needed for performance testing (such as web load testing tools, other tools already in house that can be adapted, web robot downloading tools, etc.)?

* Who is the target audience? What kind of browsers will they be using? What kind of connection speeds will they by using? Are they intra- organization (thus with likely high connection speeds and similar browsers) or Internet-wide (thus with a wide variety of connection speeds and browser types)?

* What kind of performance is expected on the client side (e.g., how fast should pages appear, how fast should animations, applets, etc. load and run)?

* Will down time for server and content maintenance/upgrades be allowed? how much?
* Will down time for server and content maintenance/upgrades be allowed? how much?
* How reliable are the site's Internet connections required to be? And how does that affect backup system or redundant connection requirements and testing?

* What processes will be required to manage updates to the web site's content, and what are the requirements for maintaining, tracking, and controlling page content, graphics, links, etc.?
* Which HTML specification will be adhered to? How strictly? What variations will be allowed for targeted browsers?
* Will there be any standards or requirements for page appearance and/or graphics throughout a site or parts of a site?
* How will internal and external links be validated and updated? how often?

* Can testing be done on the production system, or will a separate test system be required? How are browser caching, variations in browser option settings, dial-up connection variabilities, and real-world internet 'traffic congestion' problems to be accounted for in testing?
* How extensive or customized are the server logging and reporting requirements; are they considered an integral part of the system and do they require testing?

* How are cgi programs, applets, javascripts, ActiveX components, etc. to be maintained, tracked, controlled, and tested?
* Pages should be 3-5 screens max unless content is tightly focused on a single topic. If larger, provide internal links within the page.

* The page layouts and design elements should be consistent throughout a site, so that it's clear to the user that they're still within a site.
* Pages should be as browser-independent as possible, or pages should be provided or generated based on the browser-type.
* All pages should have links external to the page; there should be no dead-end pages.
* The page owner, revision date, and a link to a contact person or organization should be included on each page.
What is Extreme Programming and what's it got to do with testing?

* Extreme Programming (XP) is a software development approach for small teams on risk-prone projects with unstable requirements. It was created by Kent Beck who described the approach in his book 'Extreme Programming Explained' (See the Softwareqatest.com Books page.). Testing ('extreme testing') is a core aspect of Extreme Programming. Programmers are expected to write unit and functional test code first - before the application is developed. Test code is under source control along with the rest of the code. Customers are expected to be an integral part of the project team and to help develope scenarios for acceptance/black box testing. Acceptance tests are preferably automated, and are modified and rerun for each of the frequent development iterations. QA and test personnel are also required to be an integral part of the project team. Detailed requirements documentation is not used, and frequent re-scheduling, re-estimating, and re-prioritizing is expected.

151. What is the general testing process?

The general testing process is the creation of a test strategy (which sometimes includes the creation of test cases), the creation of a test plan/design (which usually includes test cases and test procedures), and the execution of tests.

152. How do you create a test plan/design?

Test scenarios and/or cases are prepared by reviewing the functional requirements of the release and preparing logical groups of functions that can be further broken into test procedures. Test procedures define test conditions, data to be used for testing, and expected results, including database updates, file outputs, and report results. Generally speaking...

- Test cases and scenarios are designed to represent both typical and unusual situations that may occur in the application.
- Test engineers define unit test requirements and unit test cases. Test engineers also execute unit test cases.
- It is the test team that, with the assistance of developers and clients, develops test cases and scenarios for integration and system testing.
- Test scenarios are executed through the use of test procedures or scripts.
- Test procedures or scripts define a series of steps necessary to perform one or more test scenarios.
- Test procedures or scripts include the specific data that will be used for testing the process or transaction.
- Test procedures or scripts may cover multiple test scenarios.
- Test scripts are mapped back to the requirements and traceability matrices are used to ensure each test is within scope.
- Test data is captured and base lined, prior to testing. This data serves as the foundation for unit and system testing and is used to exercise system functionality in a controlled environment.
- Some output data is also base-lined for future comparison. Base-lined data is used to support future application maintenance via regression testing.
- A pretest meeting is held to assess the readiness of the application and the environment and data to be tested. A test readiness document is created to indicate the status of the entrance criteria of the release.

Inputs for this process:

- Approved Test Strategy Document.
- Test tools, or automated test tools, if applicable.
- Previously developed scripts, if applicable.
- Test documentation problems uncovered as a result of testing.
- A good understanding of software complexity and module path coverage, derived from general and detailed design documents, e.g. software design document, source code, and software complexity data.

Outputs for this process:

- Approved documents of test scenarios, test cases, test conditions, and test data.
- Reports of software design issues, given to software developers for correction.

153. What is a security clearance?

A: Security clearance is a process of determining your trustworthiness and reliability before granting you access to national security information.

154. What are the levels of classified access?

A: The levels of classified access are confidential, secret, top secret, and sensitive compartmented information, of which top secret is the highest.

155. How do you execute tests?

A: Execution of tests is completed by following the test documents in a methodical manner. As each test procedure is performed, an entry is recorded in a test execution log to note the execution of the procedure and whether or not the test procedure uncovered any defects. Checkpoint meetings are held throughout the execution phase. Checkpoint meetings are held daily, if required, to address and discuss testing issues, status and activities.

- The output from the execution of test procedures is known as test results. Test results are evaluated by test engineers to determine whether the expected results have been obtained. All discrepancies/anomalies are logged and discussed with the software team lead, hardware test lead, programmers, software engineers and documented for further investigation and resolution. Every company has a different process for logging and reporting bugs/defects uncovered during testing.
- A pass/fail criteria is used to determine the severity of a problem, and results are recorded in a test summary report. The severity of a problem, found during system testing, is defined in accordance to the customer's risk assessment and recorded in their selected tracking tool.
- Proposed fixes are delivered to the testing environment, based on the severity of the problem. Fixes are regression tested and flawless fixes are migrated to a new baseline. Following completion of the test, members of the test team prepare a summary report. The summary report is reviewed by the Project Manager, Software QA Manager and/or Test Team Lead.
- After a particular level of testing has been certified, it is the responsibility of the Configuration Manager to coordinate the migration of the release software components to the next test level, as documented in the Configuration Management Plan. The software is only migrated to the production environment after the Project Manager's formal acceptance.
- The test team reviews test document problems identified during testing, and update documents where appropriate.

Inputs for this process:

- Approved test documents, e.g. Test Plan, Test Cases, Test Procedures.
- Test tools, including automated test tools, if applicable.
- Developed scripts.
- Changes to the design, i.e. Change Request Documents.
- Test data.
- Availability of the test team and project team.
- General and Detailed Design Documents, i.e. Requirements Document, Software Design Document.
- A software that has been migrated to the test environment, i.e. unit tested code, via the Configuration/Build Manager.
- Test Readiness Document.
- Document Updates.

Outputs for this process:

- Log and summary of the test results. Usually this is part of the Test Report. This needs to be approved and signed-off with revised testing deliverables.
- Changes to the code, also known as test fixes.
- Test document problems uncovered as a result of testing. Examples are Requirements document and Design Document problems.
- Reports on software design issues, given to software developers for correction. Examples are bug reports on code issues.
- Formal record of test incidents, usually part of problem tracking.
- Base-lined package, also known as tested source and object code, ready for migration to the next level.

156. What's a 'test plan'?

A software project test plan is a document that describes the objectives, scope, approach, and focus of a software testing effort. The process of preparing a test plan is a useful way to think through the efforts needed to validate the acceptability of a software product. The completed document will help people outside the test group understand the 'why' and 'how' of product validation. It should be thorough enough to be useful but not so thorough that no one outside the test group will read it. The following are some of the items that might be included in a test plan, depending on the particular project:

* Title

* Identification of software including version/release numbers.

* Revision history of document including authors, dates, approvals.

* Table of Contents.

* Purpose of document, intended audience

* Objective of testing effort

* Software product overview

* Relevant related document list, such as requirements, design documents, other test plans, etc.

* Relevant standards or legal requirements

* Traceability requirements

* Relevant naming conventions and identifier conventions

* Overall software project organization and personnel/contact-info/responsibilties

* Test organization and personnel/contact-info/responsibilities

* Assumptions and dependencies

* Project risk analysis

* Testing priorities and focus

* Scope and limitations of testing

* Test outline - a decomposition of the test approach by test type, feature, functionality, process, system, module, etc. as applicable

* Outline of data input equivalence classes, boundary value analysis, error classes

* Test environment - hardware, operating systems, other required software, data configurations, interfaces to other systems

* Test environment validity analysis - differences between the test and production systems and their impact on test validity.

* Test environment setup and configuration issues

* Software migration processes

* Software CM processes

• * Test data setup requirements

* Database setup requirements

* Outline of system-logging/error-logging/other capabilities, and tools such as screen capture software, that will be used to help describe and report bugs

* Discussion of any specialized software or hardware tools that will be used by testers to help track the cause or source of bugs

* Test automation - justification and overview

* Test tools to be used, including versions, patches, etc.

* Test script/test code maintenance processes and version control

* Problem tracking and resolution - tools and processes

* Project test metrics to be used

* Reporting requirements and testing deliverables

* Software entrance and exit criteria

* Initial sanity testing period and criteria

* Test suspension and restart criteria

* Personnel allocation

* Personnel pre-training needs

* Test site/location

* Outside test organizations to be utilized and their purpose, responsibilties, deliverables, contact persons, and coordination issues.

* Relevant proprietary, classified, security, and licensing issues.

* Open issues

* Appendix - glossary, acronyms, etc.

157. What's a 'test case'?

* A test case is a document that describes an input, action, or event and an expected response, to determine if a feature of an application is working correctly. A test case should contain particulars such as test case identifier, test case name, objective, test conditions/setup, input data requirements, steps, and expected results.

* Note that the process of developing test cases can help find problems in the requirements or design of an application, since it requires completely thinking through the operation of the application. For this reason, it's useful to prepare test cases early in the development cycle if possible.

158. How do you create a test strategy?

The test strategy is a formal description of how a software product will be tested. A test strategy is developed for all levels of testing, as required. The test team analyzes the requirements, writes the test strategy, and reviews the plan with the project team. The test plan may include test cases, conditions, the test environment, a list of related tasks, pass/fail criteria, and risk assessment.
Inputs for this process:
• A description of the required hardware and software components, including test tools. This information comes from the test environment, including test tool data.
• A description of roles and responsibilities of the resources required for the test and schedule constraints. This information comes from man-hours and schedules.
• Testing methodology. This is based on known standards.
• Functional and technical requirements of the application. This information comes from requirements, change requests, and technical and functional design documents.
• Requirements that the system can not provide, e.g. system limitations.
Outputs for this process:
• An approved and signed-off test strategy document, and test plan, including test cases.
• Testing issues requiring resolution. Usually, this requires additional negotiation at the project management level.

159. What if the software is so buggy it can't really be tested at all?

* The best bet in this situation is for the testers to go through the process of reporting whatever bugs or blocking-type problems initially show up, with the focus being on critical bugs. Since this type of problem can severely affect schedules, and indicates deeper problems in the software development process (such as insufficient unit testing or insufficient integration testing, poor design, improper build or release procedures, etc.) managers should be notified, and provided with some documentation as evidence of the problem.

159. How can it be known when to stop testing?

This can be difficult to determine. Many modern software applications are so complex, and run in such an interdependent environment, that complete testing can never be done. Common factors in deciding when to stop are:

* Deadlines (release deadlines, testing deadlines, etc.)

* Test cases completed with certain percentage passed

* Test budget depleted

* Coverage of code/functionality/requirements reaches a specified point

* Bug rate falls below a certain level

* Beta or alpha testing period ends

160. What should be done after a bug is found?

* The bug needs to be communicated and assigned to developers that can fix it. After the problem is resolved, fixes should be re-tested, and determinations made regarding requirements for regression testing to check that fixes didn't create problems elsewhere. If a problem-tracking system is in place, it should encapsulate these processes. A variety of commercial problem-tracking/management software tools are available (see the 'Tools' section for web resources with listings of such tools). The following are items to consider in the tracking process:

* Complete information such that developers can understand the bug, get an idea of it's severity, and reproduce it if necessary.
* Bug identifier (number, ID, etc.)

* Current bug status (e.g., 'Released for Retest', 'New', etc.)

* The application name or identifier and version

* The function, module, feature, object, screen, etc. where the bug occurred

* Environment specifics, system, platform, relevant hardware specifics

* Test case name/number/identifier

* One-line bug description

* Full bug description

* Description of steps needed to reproduce the bug if not covered by a test case or if the developer doesn't have easy access to the test case/test script/test tool

* Names and/or descriptions of file/data/messages/etc. used in test

* File excerpts/error messages/log file excerpts/screen shots/test tool logs that would be helpful in finding the cause of the problem

* Severity estimate (a 5-level range such as 1-5 or 'critical'-to-'low' is common)

* Was the bug reproducible?

* Tester name

* Test date

* Bug reporting date

* Name of developer/group/organization the problem is assigned to

* Description of problem cause

* Description of fix

* Code section/file/module/class/method that was fixed

* Date of fix

* Application version that contains the fix

* Tester responsible for retest

* Retest date

* Retest results

* Regression testing requirements

* Tester responsible for regression tests

* Regression testing results

* A reporting or tracking process should enable notification of appropriate personnel at various stages. For instance, testers need to know when retesting is needed, developers need to know when bugs are found and how to get the needed information, and reporting/summary capabilities are needed for managers.

161. What if there isn't enough time for thorough testing?

* Use risk analysis to determine where testing should be focused. Since it's rarely possible to test every possible aspect of an application, every possible combination of events, every dependency, or everything that could go wrong, risk analysis is appropriate to most software development projects. This requires judgement skills, common sense, and experience. (If warranted, formal methods are also available.) Considerations can include:

* Which functionality is most important to the project's intended purpose?

* Which functionality is most visible to the user?

* Which functionality has the largest safety impact?

* Which functionality has the largest financial impact on users?

* Which aspects of the application are most important to the customer?

* Which aspects of the application can be tested early in the development cycle?

* Which parts of the code are most complex, and thus most subject to errors?

* Which parts of the application were developed in rush or panic mode?

* Which aspects of similar/related previous projects caused problems?

* Which aspects of similar/related previous projects had large maintenance expenses?

* Which parts of the requirements and design are unclear or poorly thought out?

* What do the developers think are the highest-risk aspects of the application?

* What kinds of problems would cause the worst publicity?

* What kinds of problems would cause the most customer service complaints?

* What kinds of tests could easily cover multiple functionalities?

* Which tests will have the best high-risk coverage to the time-required ratio?

162. How can it be known when to stop testing?

This can be difficult to determine. Many modern software applications are so complex, and run in such an interdependent environment, that complete testing can never be done. Common factors in deciding when to stop are:

* Deadlines (release deadlines, testing deadlines, etc.)

* Test cases completed with certain percentage passed

* Test budget depleted

* Coverage of code/functionality/requirements reaches a specified point

* Bug rate falls below a certain level

* Beta or alpha testing period ends

163. What if the project isn't big enough to justify extensive testing?

* Consider the impact of project errors, not the size of the project. However, if extensive testing is still not justified, risk analysis is again needed and the same considerations as described previously in 'What if there isn't enough time for thorough testing?' apply. The tester might then do ad hoc testing, or write up a limited test plan based on the risk analysis.

164. What can be done if requirements are changing continuously?

A common problem and a major headache

* Work with the project's stakeholders early on to understand how requirements might change so that alternate test plans and strategies can be worked out in advance, if possible.

* It's helpful if the application's initial design allows for some adaptability so that later changes do not require redoing the application from scratch.

* If the code is well-commented and well-documented this makes changes easier for the developers.

* Use rapid prototyping whenever possible to help customers feel sure of their requirements and minimize changes.

* The project's initial schedule should allow for some extra time commensurate with the possibility of changes.

* Try to move new requirements to a 'Phase 2' version of an application, while using the original requirements for the 'Phase 1' version.

* Negotiate to allow only easily-implemented new requirements into the project, while moving more difficult new requirements into future versions of the application.

* Be sure that customers and management understand the scheduling impacts, inherent risks, and costs of significant requirements changes. Then let management or the customers (not the developers or testers) decide if the changes are warranted - after all, that's their job.

* Balance the effort put into setting up automated testing with the expected effort required to re-do them to deal with changes.

* Try to design some flexibility into automated test scripts.

* Focus initial automated testing on application aspects that are most likely to remain unchanged.

* Devote appropriate effort to risk analysis of changes to minimize regression testing needs.

* Design some flexibility into test cases (this is not easily done; the best bet might be to minimize the detail in the test cases, or set up only higher-level generic-type test plans)

* Focus less on detailed test plans and test cases and more on ad hoc testing (with an understanding of the added risk that this entails).

165. What is performance testing?

Although performance testing is described as a part of system testing, it can be regarded as a distinct level of testing. Performance testing verifies loads, volumes and response times, as defined by requirements.

166. What is load testing?

A: Load testing is testing an application under heavy loads, such as the testing of a website under a range of loads to determine at what point the system response time will degrade or fail.

167. What is installation testing?

Installation testing is testing full, partial, upgrade, or install/uninstall processes. The installation test for a release is conducted with the objective of demonstrating production readiness.
This test includes the inventory of configuration items, performed by the application's System Administration, the evaluation of data readiness, and dynamic tests focused on basic system functionality. When necessary, a sanity test is performed, following installation testing.

168. What is security/penetration testing?

A: Security/penetration testing is testing how well the system is protected against unauthorized internal or external access, or willful damage.

This type of testing usually requires sophisticated testing techniques.

169. What is recovery/error testing?

Recovery/error testing is testing how well a system recovers from crashes, hardware failures, or other catastrophic problems.

170. What is compatibility testing?

Compatibility testing is testing how well software performs in a particular hardware, software, operating system, or network
This test includes the inventory of configuration items, performed by the application's System Administration, the evaluation of data readiness, and dynamic tests focused on basic system functionality. When necessary, a sanity test is performed, following installation testing.

171. What is security/penetration testing?

Security/penetration testing is testing how well the system is protected against unauthorized internal or external access or willful damage.

This type of testing usually requires sophisticated testing techniques.

172. What is recovery/error testing?

Recovery/error testing is testing how well a system recovers from crashes, hardware failures, or other catastrophic problems.

173. What is compatibility testing?

Compatibility testing is testing how well software performs in a particular hardware, software, operating system, or network

174. What is comparison testing?

A: Comparison testing is testing that compares software weaknesses and strengths to those of competitors' products.

175. What is acceptance testing?

Acceptance testing is black box testing that gives the client/customer/project manager the opportunity to verify the system functionality and usability prior to the system being released to production.

The acceptance test is the responsibility of the client/customer or project manager, however, it is conducted with the full support of the project team. The test team also works with the client/customer/project manager to develop the acceptance criteria.

176. What is alpha testing?

Alpha testing is testing of an application when development is nearing completion. Minor design changes can still be made as a result of alpha testing. Alpha testing is typically performed by a group that is independent of the design team, but still within the company, e.g. in-house software test engineers, or software QA engineers.

177. What is beta testing?

Beta testing is testing an application when development and testing are essentially completed and final bugs and problems need to be found before the final release. Beta testing is typically performed by end-users or others, not programmers, software engineers, or test engineers.

178. What is a Test/QA Team Lead?

A: The Test/QA Team Lead coordinates the testing activity, communicates testing status to management and manages the test team.

179. What testing roles are standard on most testing projects?

Depending on the organization, the following roles are more or less standard on most testing projects: Testers, Test Engineers, Test/QA Team Lead, Test/QA Manager, System Administrator, Database Administrator, Technical Analyst, Test Build Manager and Test Configuration Manager.

Depending on the project, one person may wear more than one hat. For instance, Test Engineers may also wear the hat of Technical Analyst, Test Build Manager and Test Configuration Manager.

180. What is a Test Build Manager?

Test Build Managers deliver current software versions to the test environment, install the application's software and apply software patches, to both the application and the operating system, set-up, maintain and back up test environment hardware.

Depending on the project, one person may wear more than one hat. For instance, a Test Engineer may also wear the hat of a Test Build Manager.

181. What is a Test Engineer?

We, test engineers, are engineers who specialize in testing. We, test engineers, create test cases, procedures, and scripts, and generate data. We execute test procedures and scripts, analyze standards of measurement, and evaluate the results of system/integration/regression testing. We also...

- Speed up the work of the development staff;
- Reduce your organization's risk of legal liability;
- Give you the evidence that your software is correct and operates properly;
- Improve problem tracking and reporting;
- Maximize the value of your software;
- Maximize the value of the devices that use it;
- Assure the successful launch of your product by discovering bugs and design flaws, before users get discouraged before shareholders lose their cool, and before employees get bogged down;
- Help the work of your development staff, so the development team can devote its time to building up your product;
- Promote continual improvement;
- Provide documentation required by FDA, FAA, other regulatory agencies, and your customers;
- Save money by discovering defects 'early' in the design process, before failures occur in production, or in the field;
- Save the reputation of your company by discovering bugs and design flaws; before bugs and design flaws damage the reputation of your company.

182. What is a System Administrator?

Test Build Managers, System Administrators, Database Administrators deliver current software versions to the test environment, install the application's software and apply software patches, to both the application and the operating system, set-up, maintain and back up test environment hardware.

Depending on the project, one person may wear more than one hat. For instance, a Test Engineer may also wear the hat of a System Administrator.

183. What is a Database Administrator?

Test Build Managers, System Administrators and Database Administrators deliver current software versions to the test environment, install the application's software and apply software patches, to both the application and the operating system, set-up, maintain and back up test environment hardware. Depending on the project, one person may wear more than one hat. For instance, a Test Engineer may also wear the hat of a Database Administrator.

184. What is a Technical Analyst?

A: Technical Analysts perform test assessments and validate system/functional test requirements. Depending on the project, one person may wear more than one hat. For instance, Test Engineers may also wear the hat of a Technical Analyst.

185. What is a Test Configuration Manager?

A: Test Configuration Managers maintain test environments, scripts, software and test data. Depending on the project, one person may wear more than one hat. For instance, Test Engineers may also wear the hat of a Test Configuration Manager.

186. What is a test schedule?

A: The test schedule is a schedule that identifies all tasks required for a successful testing effort, a schedule of all test activities and resource requirements.

187. What is software testing methodology?

A: One software testing methodology is the use a three-step process of...
1. Creating a test strategy;
2. Creating a test plan/design; and
3. Executing tests.
This methodology can be used and molded to your organization's needs.