Things to know when you are being interviewed for a QA position

QA Division

Versión	2009-03-13

Version History

Date	Responsible	Commentaries	Modified Sections
2009-02-20	Alvaro Perez	Initial Document	All
2009-03-04	Juan Carlos Porras	Final Review and formatting	All
2009-03-13	Katherine Prendice	Update Q&A sections	All

TABLE CONTENTS

1 Introduction

1.1. Purpose

The purpose of this document is to define a database of knowledge with the typical questions and situations that the QA Engineers can faced during an interview for a QA position.

1.2. Audience

This document is intended to all the QA Engineers and QA people of Avantica.

1.3. Sope

Present a typical list of questions and situations normally used on QA interviews; it is not pretend to present an exhausted list, but normal questions to validate the knowledge and the skills for a QA position.

1.4. Teminology

Término	Definición
SQAE	Software Quality Assurance Engineer
SQAL	Software Quality Assurance Leader

2 General comments

The interviews typically are stressed and uncomfortable; however they can be the opposite depending on your attitudes and aptitudes.

When you are being interviewed; you must be strong into two branches:

a. Soft skills

b. Technical knowledge.

What we know about soft skills?

Well, the soft skills are not related to technical knowledge; basically they are a set of attitudes that you have to have when you are in front of your client/interviewer. But, what kind of attitudes we are talking about? Please, see the list below:

- Don't be late; always attending the interviews with some minutes before.
- Be calmed.
- Ensure to wear a proper dress code.
- Focus your sight on the interviewer's face. (Don't see objects around, never toward the ceiling)
- Talk with a proper tone and speed; the interviewer must feel that you are ok.
- Be honest; customers want to know what you know, what you can do, and what you are like. Nobody wants to hire a liar.
- Don't give short answer and always try to give direct and clear answers
- Keep confidence in yourself.
- Be positive, demonstrate that you are the right fit for the position that client/interviewer is looking for.
- Try to show interesting about the interviewer's company; for instance talking about the company's business, current projects, company's expectations, etc.

All these attitudes may allow you to have a proper performance in front of the client/interviewer; however, now we have to talk about the minimum QA knowledge that you must have in order to demonstrate that you are the right fit for the client. But if you don't have it; all those attitudes won't help you at all.

Our Technical knowledge branch is divided into two different roles:

1. SQAE (Software Quality Assurance Engineer)
2. SQAL (Software Quality Assurance Leader).

For each of these two roles we have different technical knowledge that must be known.

3 Soft Skills Questions

In order to evaluate your personal skills the interviewer could make you several personal questions. We are listing some possible questions:

1. Why did you ever become involved in QA/Testing?
2. What are you strengths?
3. What are you weaknesses?
4. What motivates you?
5. What goals have you set for yourself? How are you planning to achieve them?
6. What type of work environment appeals to you most?
7. What work experiences have been most valuable to you and why?
8. Tell me about a situation in which you were under tremendous pressure and how you dealt with it.
9. Give me a situation in which you failed, and how you handled it.
10. Have you ever supervised people? What is your management style?
11. Describe the type of manager you prefer.
12. How do you handle a customer or user who becomes irate?
13. How do you handle stress and pressure?

All this questions must be answer sincerely with your personal opinion.

4 Software Quality Assurance Engineer Technical Skills Questions

The Technical knowledge for all SQAEs must be enough to answer all these questions

4.1 About General Concepts

4.1.1 Quality Concepts

1. 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.

2. What is”Quality Control”?

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

3. What's the difference between Quality Assurance (QA) and Quality Control (Testing)?

   Quality Assurance is more a preventive thing, ensuring quality in the company and therefore the product rather than just testing the product for software bugs. Furthermore Quality Assurance measures the quality of processes used to create a quality product.

   Quality Control or Testing measures the quality off a specific product.

   Quality assurance involves the entire software development process and testing involves operation of a system or application to evaluate the results under certain conditions. QA is oriented to prevention and Testing is oriented to detection.

4. What is Software Testing?

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

5. What is testing?

   It is 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).

   It is 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.

4.1.2 Software Tests

1. What are the Software Tests?

   Software Tests have as main goal shown the biggest possible quantity of defects in order to minimize the risk of hide defects appear after the product is released.

2. What is Acceptance Testing?

   Testing 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.

3. What is Accessibility Testing?

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

4. What is Ad Hoc Testing?

   A testing phase where a tester tries to 'break' the system randomly. Can included negative testing as well.

5. What is Agile Methodology?

   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.

6. What is Black Box Testing?

   Testing 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.

7. What is Bottom Up Testing?

   An approach to integration testing where the lowest level components are tested first, 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.

8. What is Boundary Testing?

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

9. What is Compatibility Testing?

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

10. What is Concurrency Testing?

    Multi-user testing 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

11. 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.

12. 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.

13. 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.

14. What is Integration Testing?

    Testing of combined parts of an application to determine if together function correctly. It is usually performed after unit and functional testing.

    This type of testing is especially relevant to client/server and distributed systems.

15. What is Negative Testing?

    Testing aimed at showing software does not work. It is also known as "test to fail".

16. What is Positive Testing?
    Testing aimed at showing software works. It is also known as "test to pass".

17. 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.

18. What is Sanity Testing?

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

19. What is Smoke Testing?

    A quick-and-dirty test used to evaluate the major functions of a piece of software work.

20. What is System Testing?

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

21. What is a Top Down Testing?

    It is an approach of 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.

22. What is Usability Testing?

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

23. What is Unit Testing?

    Testing of individual software's components.

24. What is White Box or Glass Testing?

    Testing based on an analysis of internal workings and structure of a piece of software. It includes techniques such Branch Testing and Path Testing.

    It is also known as Structural Testing and Glass Box Testing. Contrast with Black Box Testing.

25. What kind of testing should be considered?

    Black box testing: Not based on any knowledge of internal design or code. Tests are based on requirements and functionality.

    White box testing: Based on knowledge of the internal logic of an application's code. Tests are based on coverage of code statements, branches, paths, conditions.

    Unit testing: the most 'micro' scale of testing; to test particular functions or code modules. It is typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program design and code. Not always easily done unless the application has a well-designed architecture with tight code; may require developing test driver modules or test harnesses.

Incremental Integration testing: Continuous testing of an application as new functionality is added; requires that various aspects of an application's functionality be independent enough to work separately before all parts of the program are completed, or that test drivers be developed as needed; done by programmers or by testers.

Integration testing: Testing of combined parts of an application to determine if they function together correctly. The 'parts' can be code modules, individual applications, client and server applications on a network, etc. This type of testing is especially relevant to client/server and distributed systems.

Functional testing - black-box type testing: Geared to functional requirements of an application; this type of testing should be done by testers. This doesn't mean that the programmers shouldn't check that their code works before releasing it (which of course applies to any stage of testing.)

System testing - black-box type testing: That is based on overall requirements specifications; covers all combined parts of a system.

End-to-end testing - similar to system testing: The 'macro' end of the test scale; involves testing of 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.

Sanity testing or smoke testing: Typically an initial testing effort to determine if a new software version is performing well enough to accept it for a major testing effort. For example, if the new software is crashing systems every 5 minutes, bogging down systems to a crawl, or corrupting databases, the software may not be in a 'sane' enough condition to warrant further testing in its current state.

Regression testing: Re-testing after fixes or modifications of the software or its environment. It can be difficult to determine how much re-testing is needed, especially near the end of the development cycle. Automated testing tools can be especially useful for this type of testing.

Acceptance testing: Final testing based on specifications of the end-user or customer, or based on use by end-users/customers over some limited period of time.

Load testing: Testing an application under heavy loads, such as testing of a web site under a range of loads to determine at what point the system's response time degrades or fails.

Stress testing: Term often used interchangeably with 'load' and 'performance' testing. Also used to describe such tests as system functional testing while under unusually heavy loads, heavy repetition of certain actions or inputs, input of large numerical values, large complex queries to a database system, etc.

Performance testing: Term often used interchangeably with 'stress' and 'load' testing. Ideally 'performance' testing (and any other 'type' of testing) is defined in requirements documentation or QA or Test Plans.

Usability testing: Testing for 'user-friendliness'. Clearly this is subjective, and will depend on the targeted end-user or customer. User interviews, surveys, video recording of user sessions, and other techniques can be used. Programmers and testers are usually not appropriate as usability testers.

Install/uninstall testing: Testing of full, partial, or upgrade install/uninstall processes.

Recovery testing: Testing how well a system recovers from crashes, hardware failures, or other catastrophic problems.

Security testing: Testing how well the system protects against unauthorized internal or external access, willful damage, etc; may require sophisticated testing techniques.

Exploratory testing: Often taken to mean a creative, informal software test that is not based on formal test plans or test cases; testers may be learning the software as they test it.

Ad-hoc testing: Similar to exploratory testing, but often taken to mean that the testers have significant understanding of the software before testing it.

User acceptance testing: Determining if software is satisfactory to an end-user or customer.

4.1.3 Software Test Life Cycle

1. What is Functional Specification?

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

2. What are the properties of a good requirement?

   A good requirement must be complete, consistent and traceable. Also, the requirement has to be verifiable, clear & non-ambiguous; using short explanations.

3. How testing is proceeded when SRS or any other document is not given?

   If SRS is not there we can perform exploratory testing. In Exploratory testing the basic module is executed and depending on its results, the next plan is executed.

4. How do test if we have minimal or no documentation about the product?

   When we are on this kind of circumstances; we should use exploratory testing technique or ad-hoc testing. Another option is to follow Prototype model creating the design template of the application

5. What is a Traceability Matrix?

   It is a document showing the relationship between Test Requirements and Test Cases.

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

   - 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.

   - 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).

7. 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 effects areas such as minor improvements in the user interface, for example, it may not be a significant risk.

8. What is a 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.

9. What is a Test Specification?

   It is 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.

10. What is a 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 cases related by what they are intended to test.

11. What is a 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.

12. What is a 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.

13. What is a test scope

    It is a document where the QAE give a formal estimation for Test Cycles, in terms of time (days, hours) and Tests to perform. With this document, the PM could know how much time and resources will be involved into the QA activity.

14. 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.

15. What is a Defect? How do you difference a defect and a non-defect?

    The defect is a nonconformance to requirements or functional program specification. If software misses some feature or function from what is there in requirement it is called as defect; otherwise it is a non-defect.

16. What is a Bug? What differences do you see between a bug and a defect?

    It is a fault in a program, which causes the program to perform in an unintended or unanticipated manner.

17. What is the Life Cycle of a bug?

    The Life Cycle of a bug in general context is:

    Bugs are usually logged by the development team (While Unit Testing) and also by testers (While system or other type of testing).

    A tester finds a new defect/bug, so using a defect tracking tool logs it.

    1. Its status is 'NEW' and assigns to the respective dev team (Team lead or Manager).
    2. The team lead assign's it to the team member, so the status is 'ASSIGNED TO'
    3. The developer works on the bug fixes it and re-assigns to the tester for testing. Now the status is 'RE-ASSIGNED'
    4. The tester, check if the defect is fixed, if its fixed he changes the status to 'VERIFIED'
    5. If the tester has the authority (depends on the company) he can after verifying change the status to 'FIXED'. If not the test lead can verify it and change the status to 'fixed'.
    6. If the defect is not fixed he re-assign's the defect back to the dev team for re-fixing.

18. What is severity level? and how would you categorize the severity of defects?

    The degree of impact the issue or problem has on the project. Severity 1 usually means the highest level requiring immediate attention. Severity 5 usually represents a documentation defect of minimal impact.

Severity	Description
Critical	• Inconsistent data is stored on the database • System crash with data loss
Major	• A process aborts • System crash • Major functionality malfunction • Functional issue with difficult workaround
Minor	• Minor functionality malfunction • Functional issue with simple workaround • Critical usability issue
Trivial	• Typo, grammar mistake, misspelling and wrong terminology usage • General usability issue • Stylistic issue and details

19. Bug Impact Level

    Low impact
    This is for Minor problems, such as failures at extreme boundary conditions that are unlikely to occur in normal use, or minor errors in layout/formatting. These problems do not impact use of the product in any substantive way.

    Medium impact
    This is a problem that a) Effects a more isolated piece of functionality. b) Occurs only at certain boundary conditions. c) Has a workaround (where "don't do that" might be an acceptable answer to the user). d) Occurs only at one or two customers. or e) Is very intermittent

    High impact
    This should be used for only serious problems, effecting many sites, with no workaround. Frequent or reproducible crashes/core dumps/GPFs would fall in this category, as would major functionality not working.

    Urgent impact
    This should be reserved for only the most catastrophic of problems. Data corruption, complete inability to use the product at almost any site, etc. For released products, an urgent bug would imply that shipping of the product should stop immediately, until the problem is resolved.

20. What is Priority? And What are the priority levels?

    Priority is Business.
    Priority is a measure of importance of getting the defect fixed as governed by the impact to the application, number of users affected, and company's reputation, and/or loss of money.

    The priority levels are:
    Now: drop everything and take care of it as soon as you see this (usually for blocking bugs)
    P1: fix before next build to test
    P2: fix before final release
    P3: we probably won't get to these, but we want to track them anyway

21. What difference between Severity and Priority?

    Priority is Relative: the priority might change over time. Perhaps a bug initially deemed P1 becomes rated as P2 or even a P3 as the schedule draws closer to the release and as the test team finds even more heinous errors. Priority is a subjective evaluation of how important an issue is, given other tasks in the queue and the current schedule. It's relative. It shifts over time. And it's a business decision.

    Severity is an absolute: it's an assessment of the impact of the bug without regard to other work in the queue or the current schedule. The only reason severity should change is if we have new information that causes us to re-evaluate our assessment. If it was a high severity issue when I entered it, it's still a high severity issue when it's deferred to the next release. The severity hasn't changed just because we've run out of time. The priority changed.

22. What are the basic elements in a defect report?

    Basic Elements are: Bug ID, Description, Module Name, Screenshot, Open Date, Detected By, Reviewed By, Assigned To, Test Case ID, Severity, Priority, Status, Comments if Any

23. What is Debugging?

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

24. Top ten tips for Bug Tracking

    1. A good tester will always try to reduce the repro steps to the minimal steps to reproduce; this is extremely helpful for the programmer who has to find the bug.

    2. Remember that the only person who can close a bug is the person who opened it in the first place. Anyone can resolve it, but only the person who saw the bug can really be sure that what they saw is fixed.

    3. There are many ways to resolve a bug. Bugzilla allows you to resolve a bug as fixed, won't fix, postponed, not repro, duplicate, or by design.

    4. Not Repro means that nobody could ever reproduce the bug. Programmers often use this when the bug report is missing the repro steps.

5. You'll want to keep careful track of versions. Every build of the software that you give to testers should have a build ID number so that the poor tester doesn't have to retest the bug on a version of the software where it wasn't even supposed to be fixed.

6. If you're a programmer, and you're having trouble getting testers to use the bug database, just don't accept bug reports by any other method. If your testers are used to sending you email with bug reports, just bounce the emails back to them with a brief message: "please put this in the bug database. I can't keep track of emails."

7. If you're a tester, and you're having trouble getting programmers to use the bug database, just don't tell them about bugs - put them in the database and let the database email them.

8. If you're a programmer, and only some of your colleagues use the bug database, just start assigning those bugs in the database. Eventually they'll get the hint.

9. If you're a manager, and nobody seems to be using the bug database that you installed at great expense, start assigning new features to people using bugs. A bug database is also a great "unimplemented feature" database, too.

10. Avoid the temptation to add new fields to the bug database. Every month or so, somebody will come up with a great idea for a new field to put in the database. You get all kinds of clever ideas, for example, keeping track of the file where the bug was found; keeping track of what % of the time the bug is reproducible; keeping track of how many times the bug occurred; keeping track of which exact versions of which DLLs were installed on the machine where the bug happened. It's very important not to give in to these ideas. If you do, your new bug entry screen will end up with a thousand fields that you need to supply, and nobody will want to input bug reports any more. For the bug database to work, everybody needs to use it, and if entering bugs "formally" is too much work, people will go around the bug database.

25. What is a 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).

26. What is Metric?

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

27. What kind of QA Metric can be used?

    QA Metrics depends on each project and organization; however there are some metrics that can be considered:

    - Total Bugs by Severity
    - Total Bugs by Status
    - Test Maintenance Performance
    - Total Test Cases executed by resource and by build
    - Distribution of Test Cases by Resource

- Ratio of Test Cases Effectiveness
- Ratio of Valid Bugs
- Total Bug Reports vs. Total Real Bugs
- Bugs distribution by resource

28. How can it 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

4.1.4 Test Design Techniques

1. What is Equivalence Partitioning?

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

2. 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. his means that if a function expects all values in range of negative 100 to positive 1000, test inputs would include negative 101 and positive 1001.

3. What is Decision Table Analysis?

   A decision table is the method used to build a complete set of test cases without using the internal structure of the program in question. In order to create test cases we use a table to contain the input and output values of a program.

4. What is Cause-Effect Graphic?

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

5. What is State Transition?

   State transition diagrams views the software under test in term of its state, transitions between states and the inputs or events that trigger state changes. Evaluating these states and transitions test cases can be created.

6. What is Entity Life Cycle Test?

   Defining the life cycle stages, indentifying and mapping business scenarios at each life cycle stage and tracking the entity trough the entire life cycle forms the crux of “Entity Life Cycle Testing”. This method is also used to the creation of test cases.

4.1.5 Other General Concepts

1. What is a Configuration Management?

   Configuration management covers the 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.

2. What is the difference between Functional Tests and Non-Functional Tests?

   The Functional Tests are a set of Test intended to validate the right program's functionality according to the specifications; but the Non-Functional Tests are all kind of tests intended to validate the program's behavior testing implicit features not include into the specification document; for instance; performance, robustness, error handling, UI errors, usability, etc…

3. Describe the difference between validation and verification.

   Validation is the process of evaluating software at the end of the software development process to ensure compliance with software requirements. The techniques for validation are testing, inspection and reviewing.

   Verification is the process of determining whether of not the products of a given phase of the software development cycle meet the implementation steps and can be traced to the incoming objectives established during the previous phase. The techniques for verification are testing, inspection and reviewing.

4. 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, Execute the test case, Log the bug, validate the bug & take appropriate action for the bug, Automate the test cases.

4.2 About Automation Testing

1. What is automated Software Quality (ASQ)?

   It is the use of software tools, such as automated testing tools, to improve software quality.

2. What is Automated Testing?

   It is testing employing software tools which execute tests without manual intervention. Can be applied in GUI, performance, API, etc. testing.

   It is 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.

3. How do you plan test automation?
   a. Prepare the automation Test Plan
   b. Identify the scenario
   c. Record the scenario
   d. Enhance the scripts by inserting check points and Conditional Loops
   e. Incorporated Error Handle
   f. Debug the scripts
   g. Fix the issue
   h. Rerun the scripts and report the results

4. What is a Test Script?

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

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

   There are five types of scripting techniques:
   a. Linear
   b. Structured
   c. Shared
   d. Data Driven
   e. Key Driven

6. What are principles of good testing scripts for automation?
    a. Proper code guiding standards
    b. Standard format for defining functions, exception handler, etc
    c. Comments for functions
    d. Proper error handling mechanism
    e. The appropriate synchronization techniques

7. What automating testing tools are you familiar with?

    J-Meter, Selenium, Bad-Boy, Web-Inject

8. Will automated testing tools make testing easier?

    It is possibly. For small projects, the time needed to learn and implement them may not be worth it. For larger projects, or on-going long-term projects they can be valuable.

    A common type of automated tool is the 'record/playback' type. For example, a tester could click through all combinations of menu choices, dialog box choices, buttons, etc. in an application GUI and have them 'recorded' and the results logged by a tool. The 'recording' is typically in the form of text based on a scripting language that is interpretable by the testing tool. If new buttons are added, or some underlying code in the application is changed, etc. the application can then be retested by just 'playing back' the 'recorded' actions, and comparing the logging results to check effects of the changes. The problem with such tools is that if there are continual changes to the system being tested, the 'recordings' may have to be changed so much that it becomes very time-consuming to continuously update the scripts. Additionally, interpretation of results (screens, data, logs, etc.) can be a difficult task. Note that there are record/playback tools for text-based interfaces also, and for all types of platforms.

    Other automated tools can include:

    - Code analyzers; monitor code complexity, adherence to standards, etc.

    - Coverage analyzers; these tools check which parts of the code have been exercised by a test, and may be oriented to code statement coverage, condition coverage, path coverage, etc.

    - Memory analyzers; such as bounds-checkers and leak detectors.

    - Load/performance test tools; for testing client/server and web applications under various load levels.

    - Web test tools; to check that links are valid, HTML code usage is correct, client-side and server-side programs work, a web site's interactions are secure.

    - Other tools; for test case management, documentation management, bug reporting, and configuration management.

9. How did you use automating testing tools in your jobs?
    a. For regression testing
    b. Criteria to decide condition of a particular build
    c. Describe some problems that you had with automating testing tool

10. Can test automation improve test effectiveness?

    Yes, automating a test makes the test process:

    a. Fast
    b. Reliable
    c. Repeatable
    d. Programmable
    e. Reusable
    f. Comprehensive

    Testing the functionality with more test cases becomes laborious as the functionality grows. For multiple sets of data (test cases), you can execute the test once in which you can figure out for which data it has failed and for which data, the test has passed

11. What are the main attributes of test automation?

    Maintainability; the effort needed to update the test automation suites for each new release

    Reliability; the accuracy and repeatability of the test automation

    Flexibility; the ease of working with all the different kinds of automation test ware

    Efficiency; the total cost related to the effort needed for the automation

    Portability; the ability of the automated test to run on different environments

    Robustness; the effectiveness of automation on an unstable or rapidly changing system

    Usability; the extent to which automation can be used by different types of users

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

    Choosing a tool depends on many things:

    a. Application to be tested
    b. Test environment
    c. Scope and limitation of the tool.
    d. Feature of the tool.
    e. Cost of the tool.
    f. Whether the tool is compatible with your application which means tool should be able to interact with your application
    g. Ease of use

    It is recommendable before choose a tool concept test in order to know if the tools is really going to fix to the necessities.

13. What could be wrong with test automation?

    a. The choice of automation tool for certain technologies
    b. Wrong set of test automated

14. What testing activities you may want to automated?

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

    Automated all the test cases belong to an a smoke or sanity test flow.

15. Does automation replace manual testing?

    There can be some functionality which 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 hectic task).When we talk about real environment we do negative testing manually.

16. Describe common problems of test automation

    The common problems are:

    - Maintenance of the old script when there is a feature change or enhancement
    - The change in technology of the application will affect the old scripts

17. What are the limitations of automating software testing?

    Hard-to-create environments like “out of memory”, “invalid input/reply”, and “corrupt registry entries” make applications behave poorly and existing automated tools can’t force these condition - they simply test your application in “normal” environment.

4.3 About Loading, Stress and Performance Testing

1. What is Loading Testing?

   Load Tests are end to end performance tests under anticipated production load. The primary objective of this tests are to determine the response times for various time critical transactions and business processes and that they are within documented expectations. The tests also measure the capability of the application to function correctly under load, by measuring transaction pass/fail/error rates

2. 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.

3. What is Performance Testing?’

   Testing 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 large number of users.

4. What is Concurrency Testing?

   Multi-user testing 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.

5. What is Endurance Testing?

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

6. What is Soak Testing?

   It is 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.

7. What is Ramp-up Rate?

   Although your site may be handling ‘x’ number of users per day, only a small percentage of these users would be hitting your site at the same time.

   Therefore, when preparing your load test scenario, you should take into account the fact that users will hit the website at different times, and that during your peak hour the number of concurrent users will likely gradually build up to reach the peak number of users, before tailing off as the peak hour comes to a close.

   The rate at which the number of users builds up, the "Ramp-up Rate" should be factored into the load test scenarios (i.e. you should not just jump to the maximum value, but increase in a series of steps).

8. What is the difference between Load an Stress Testing?

   One of the most common, but unfortunate misuse of terminology is treating “load testing” and “stress test” as synonymous. The consequence of this ignorant semantic abuse is usually that the system is neither properly “load tested” nor subjected to a meaningful stress test.

   Stress testing is subjecting a system to an unreasonable load while denying it the resources (e.g., RAM, disc, mips, interrupts, etc.) needed to process that load. The idea is to stress a system to the breaking point in order to find bugs that will make that break potentially harmful. The system is not expected to process the overload without adequate resources, but to behave (e.g., fail) in a decent manner (e.g., not corrupting or losing data). Bugs and failure modes discovered under stress testing may or may not be repaired depending on the application, the failure mode, consequences, etc.

   The load (incoming transaction stream) in stress testing is often deliberately distorted so as to force the system into resource depletion.
   Load testing is subjecting a system to a statistically representative (usually) load. The two main reasons for using such loads is in support of software reliability testing and in performance testing. The term 'load testing' by itself is too vague and imprecise to warrant use. For example, do you mean representative load,' 'overload,' 'high load,' etc. In performance testing, load is varied from a minimum (zero) to the maximum level the system can sustain without running out of resources or having, transactions >suffer (application-specific) excessive delay.
   A third use of the term is as a test whose objective is to determine the maximum sustainable load the system can handle. In this usage, 'load testing' is merely testing at the highest transaction arrival rate in performance testing.

9. Why Scalability and Load Testing are important?

   Some very high profile websites have suffered from serious outages and/or performance issues due to the number of people hitting their website. E-commerce sites that spent heavily on advertising but not nearly enough on ensuring the quality or reliability of their service have ended up with poor web-site performance, system downtime and/or serious errors, with the predictable result that customers are being lost.

   When creating an e-Commerce portal, companies will want to know whether their infrastructure can handle the predicted levels of traffic, to measure performance and verify stability.

   These types of services include Scalability / Load / Stress testing, as well as Live Performance Monitoring.

   Load testing tools can be used to test the system behavior and performance under stressful conditions by emulating thousands of virtual users. These virtual users stress the application even harder than real users would, while monitoring the behavior and response times of the different components. This enables companies to minimize test cycles and optimize performance, hence accelerating deployment, while providing a level of confidence in the system.

   Once launched, the site can be regularly checked using Live Performance Monitoring tools to monitor site performance in real time, in order to detect and report any performance problems - before users can experience them.

10. How to prepare a Load Test?

    The first step in designing a Web site load test is to measure as accurately as possible the current load levels.

    The best way to capture the nature of Web site load is to identify and track, (e.g. using a log analyzer) a set of key user session variables that are applicable and relevant to your Web site traffic.

    Some of the variables that could be tracked include:
    - The length of the session (measured in pages)
    - The duration of the session (measured in minutes and seconds)
    - The type of pages that were visited during the session (e.g., home page, product information page, credit card information page etc.)
    - The typical/most popular ‘flow’ or path through the website
    - The % of ‘browse’ vs. ‘purchase’ sessions
    - The % type of users (new user vs. returning registered user)

    Measure how many people visit the site per week/month or day. Then break down these current traffic patterns into one-hour time slices, and identify the peak-hours (i.e. if you get lots of traffic during lunch time etc.), and the numbers of users during those peak hours. This information can then be used to estimate the number of concurrent users on your site.

11. Estimate Target Load Levels

    Once you have identified the current load levels, the next step is to understand as accurately and as objectively as possible the nature of the load that must be generated during the testing.

    Using the current usage figures, estimate how many people will visit the site per week/month or day. Then divide that number to attain realistic peak-hour scenarios.

    It is important to understand the volume patterns, and to determine what load levels your web site might be subjected to (and must therefore be tested for).

    There are four key variables that must be understood in order to estimate target load levels:

    - How the overall amount of traffic to your Web site is expected to grow
    - The peak load level which might occur within the overall traffic
    - How quickly the number of users might ramp up to that peak load level
    - How long that peak load level is expected to last

    Once you have an estimate of overall traffic growth, you will need to estimate the peak level you might expect within that overall volume.

12. Estimate Test Duration

    The duration of the peak is also very important. A Web site that may deal very well with a peak level for five or ten minutes may crumble if that same load level is sustained longer than that. You should use the length of the average user session as a base for determining the load test duration.

13. How to create the scenarios that are to be used to load test the web site?

    The information gathered during the analysis of the current traffic is used to create the scenarios that are to be used to load test the web site.

    The identified scenarios aim to accurately emulate the behavior of real users navigating through the Web site.

    For example, a seven-page session that results in a purchase is going to create more load on the Web site than a seven-page session that involves only browsing. A browsing session might only involve the serving of static pages, while a purchase session will involve a number of elements, including the inventory database, the customer database, a credit card transaction with verification going through a third-party system, and a notification email. A single purchase session might put as much load on some of the system's resources as twenty browsing sessions.

    Similar reasoning may apply to purchases from new vs. returning users. A new user purchase might involve a significant amount of account setup and verification — something existing users may not require. The database load created by a single new user purchase may equal that of five purchases by existing users, so you should differentiate the two types of purchases.

14. How to prepare a script to run each scenario with the number of types of users concurrently playing back to five you the load scenario?

    Using the load test tool, write the scripts to run each scenario with the number of types of users concurrently playing back to give you a the load scenario.

    The key elements of a load test design are:

    - Test Objective
    - Pass/ Fail criteria
    - Scritp description
    - Scenario description

    Load Test Objective
    The objective of this load test is to determine if the Web site, as currently configured, will be able to handle the X number of sessions/hr peak load level anticipated. If the system fails to scale as anticipated, the results will be analyzed to identify the bottlenecks.

    Pass/Fail Criteria
    The load test will be considered a success if the Web site will handle the target load of X number of sessions/hr while maintaining the pre-defined average page response times (if applicable). The page response time will be measured and will represent the elapsed time between a page request and the time the last byte is received.

    Since in most cases the user sessions follow just a few navigation patterns, you will not need hundreds of individual scripts to achieve realism—if you choose carefully, a dozen scripts will take care of most Web sites

15. Why System Performance Monitoring is Important?

    It is vital during the execution phase to monitor all aspects of the website, this includes measuring and monitoring the CPU usage and performance aspects of the various components of the website, i.e. not just the web server, but the database and other parts as well (such as firewalls, load balancing tools etc.)

    For example, one e-trailer whose site fell over (apparently due to a high load), when analyzing the performance bottlenecks on their site discovered that the web server had in fact only been operating at 50% of capacity. Further investigation revealed that the credit card authorization engine was the cause of failure - it was not responding quick enough for the website, which then failover when it was waiting for too many responses from the authorization engine. They resolved this issue by changing the authorization engine, and amending the website coding so that if there were any issues with authorization responses in future, the site would not crash.

    Similarly, another e-commerce site found that the performance issues that they were experiencing were due to database performance issues, while the web server CPU usage was only at 25%, the backend db server CPU usage was 86%. Their solution was to upgrade the db server.
    Therefore, it is necessary to use (install if necessary) performance monitoring tools to check each aspect of the website architecture during the execution phase.

16. How to Report Load Testing Results?

    Often the first indication that something is wrong is the end user response times start to climb. Knowing which pages are failing will help you narrow down where the problem is. Whichever load test tool you use, it will need to produce reports that will highlight the following:

    Page response time by load level

    Completed and abandoned session by load level

    Page views and page hits by load level

    HTTP and network errors by load level

    Concurrent user by minute

    Missing links report, if applicable

    Full detailed report which includes response time by page and by transaction, lost sales opportunities, analysis and recommendations

17. What are the important aspects of Websites Load Testing?

    When testing websites, it is critically important to test from outside the firewall. In addition, web-based load testing services, based outside the firewall, can identify bottlenecks that are only found by testing in this manner.

    Web-based stress testing of web sites is therefore more accurate when it comes to measuring a site's capacity constraints.

    Web traffic is rarely uniformly distributed, and most Web sites exhibit very noticeable peaks in their volume patterns. Typically, there are a few points in time (one or two days out of the week, or a couple of hours each day) when the traffic to the Web site is highest.

4.4 About Security Testing

1. What is Security Test

   Security measures protect Web systems from both internal and external threats. Security tests determine whether a company's security policies have been properly implemented; they evaluate the functionality of existing systems, not whether the security policies that have been implemented are appropriate.

   Primary components requiring security testing are:

   - Application software
   - Database
   - Servers
   - Client workstations
   - Networks

2. Security Testing Techniques
   a. ***Network Scanning***; this technique involves using a port scanner to identify all hosts connected to a network, the services operating on those hosts, such as the file transfer protocol (FTP) and hypertext transfer protocol (HTTP) and specific application running the indentified service, internet information server (ISS) and Apache for the HTTP service
   b. ***Vulnerability Scanning;*** like a port scanner a vulnerability scanner indentifies hosts and opens ports, but it also provides information on the associated vulnerabilities. Vulnerabilities scanner can help to indentify out-of-date software versions, applicable patches or system upgrades, and validate compliance with, or deviations from, the organization's security policy
   c. ***Password Cracking;*** verifies that user are employing sufficiently strong password and test how easy or difficult could be "discover" an encrypted password.
   d. ***SQL Injections;*** is a code injection technique that exploits a security vulnerability occurring in the database layer of an application
   e. ***Information Gathering;*** is used to verify if it is possible to recover any internal or/and important information about the application using public searchers like Google with special commands. It is also used to find Old backup and unreferenced files.
   f. ***Denial of Services;*** is an attempt to make a computer resource unavailable to its intended users
   g. ***Cross-site;*** is typically performed when the attacker implants executable code in the client's computer without them knowing.

3. Security Access Control

   The checklist we should use to verify the security access control is:

   a. Is there a defined standard for login names/password?
   b. Are good aging procedures in place for password?
   c. Are users locked out after a given number of password failures?
   d. Is there a link for help (e.g. forgotten passwords?)
   e. Is there a process for password administration?
   f. Have authorization levels been defined?
   g. Is management sign-off in place for authorizations?

4. Security Proxy Servers

   The checklist we should use to verify the security proxy servers is:

   a. Have undesirable/ unauthorized external sites been defined and screened out?
   b. Is traffic logged?
   c. Is user access defined?

5. Security Encryption

   The checklist we should use to verify the security encryption is:

   a. Are encryption systems/level defined?
   b. Is there a standard of what to be encrypted?
   c. Are customers compatible in terms of encryption levels and protocols?
   d. Are encryption techniques for transactions being used for secured transactions?
      - Secure Socket Layer (SSL)
      - Virtual Private Networks (VPN)
   e. Have the encryption process and standards been document?

4.5 About Database Testing

1. What is Database testing and what we test in database testing?

   An1:
   Database testing is all about testing joins, views, imports and exports , testing the procedures, checking locks, indexing etc. Its not about testing the data in the database. Usually database testing is performed by DBA.

   An2:
   Database testing involves some in depth knowledge of the given application and requires more defined plan of approach to test the data.
   Key issues include:
   1) Data Integrity
   2) Data Validity
   3) Data Manipulation and updates
   Tester must be aware of the database design concepts and implementation rules.

   An3:
   Data bas testing basically include the following.
   - Data validity testing.
   - Data Integrity testing
   - Performance related to data base.
   - Testing of Procedure, triggers and functions.

   For doing data validity testing you should be good in SQL queries.
   For data integrity testing you should know about referential integrity and different constraint.
   For performance related things you should have idea about the table structure and design.
   For testing Procedure triggers and functions you should be able to understand the same.

2. What we normally check for in the Database Testing?

   In Database testing we need to check for:

   - The field size validation
   - Check constraint
   - Indexes are done or not (to performance related issues)
   - Stored procedures
   - The field size defined in the application is matching with that in the database.

3. How to Test Database in Manually?

   Observing that operations, which are operated on front-end is effected on back-end or not.

   For example: Enter employee record in database thru' front-end and check if the record is added or not to the back-end (manually).

4. Is a “A fast database retrieval rate” a testable requirement?

   No. I do not think so. Since the requirement seems to be ambiguous. The SRS should clearly mention the performance or transaction requirements i.e. It should say like 'A DB retrieval rate of 5 micro sec'.

5. How do you test whether a database in updated when information is entered in the front end?

   It depends on your application interface.

   a. If your application provides view functionality for the entered data, then you can verify that from front end only. Most of the time Black box test engineers verify the functionality in this way.

   b. If your application has only data entry from front end and there is no view from front end, then you have to go to Database and run relevant SQL query.

   c. You can also use database checkpoint function.

6. How do you test whether the database is updated as a when an information are added in the front end? Give an example.

   It depends on what level of testing you are doing. When you want to save something from front end obviously, it has to store somewhere in the database so you will need to find out the relevant tables involved in saving the records.

   Data mapping from front end to the tables. Then enter the data from front end and save. Go to database, fire queries to get the same date from the back end.

7. How to test data loading in Database testing?

   You have to do the following things while you are involving in Data Load testing:

   - You have known about source data (table(s), columns, data types and Constraints)
   - You have to know about Target data (table(s), columns, data types and Constraints)
   - You have to check the compatibility of Source and Target.
   - You have to open corresponding DTS package in SQL Enterprise Manager and run the DTS package (If you are using SQL Server).
   - Then you should compare the column's data of Source and Target.
   - You have to check the number to rows of Source and Target.
   - Then you have to update the data in Source and see the change is reflecting in Target or not.
   - You have to check about junk character and Nulls

8. What is way of writing test cases for database testing?

   An1:
   You have to do the following for writing the database test cases.

   - First of all you have to understand the functional requirement of the application thoroughly.

   - Then you have to find out the back end tables used, joined used between the tables, cursors used (if any), triggers used(if any), stored procedures used (if any), input parameter used and output parameters used for developing that requirement.

   - After knowing all these things you have to write the test case with different input values for checking all the paths of SP.

   - Write test cases for backend testing not like functional testing. You have to use white box testing techniques.

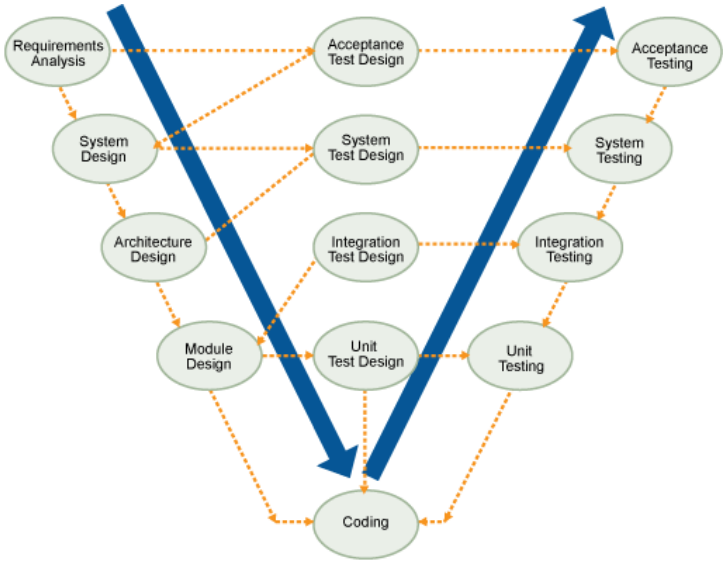
4.6 About Software Development Methodologies

1. V-Model

   Many of the process models currently used can be more generally connected by the V-model where the “V” describes the graphical arrangement of the individual phases. The “V” is also a synonym for verification and validation.

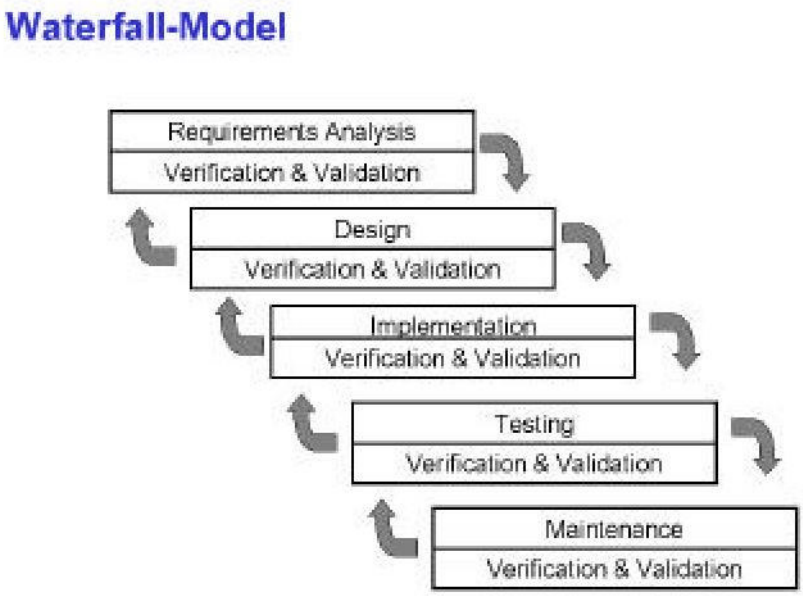
   The model is very simple and easy to understand. By the ordering of activities in time sequence and with abstraction levels the connection between development and test activities becomes clear. Oppositely laying activities complement one another i.e. serve as a base for test activities. So, for example, the system test is carried out on the basis of the results specification phase. The coarse view of the model gives the impression that the test activities first start after the implementation. However, in the description of the individual activities the preparatory work is usually listed. So, for example, the test plan and test strategy should be worked out immediately after the definition of the requirements. Nevertheless it can contribute very well to the structuring of the software development process.

   The disadvantage of the model is the coarse division into constructive work (including the implementation) on the left-hand side of the “V” and the more destructive tasks on the right-hand side. Here also the impression may develop that, after the implementation phase, a ready product can be delivered. A planned-in removal of defects and regression test is not given.



2. Waterfall Model

   One of the first models for software development is the so-called waterfall-model. The individual phases i.e. activities that were defined here are to be found in nearly all models proposed since. In this it was set out that each of the activities in the software development must be completed before the next can begin. A return in the development process was only possible to an immediate previous phase. In the waterfall-model, testing directly follows the implementation. By this model it was suggested that activities for testing could first be started after the implementation. Preparatory tasks for the testing were not clear. A further disadvantage is that testing, as the last activity before release, could be relatively easily shortened or omitted altogether. This, in practice, is unfortunately all too common. In this model, the expense of the removal of faults and defects found is only recognizable through a return to the implementation phase.



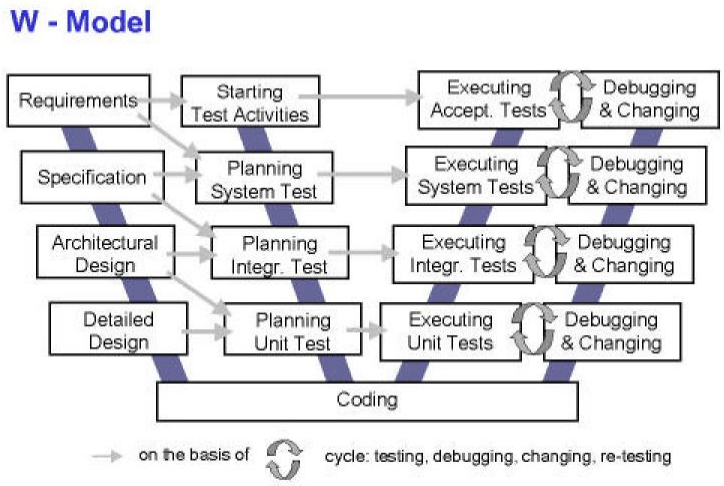
3. W-Model

From the view of testing, all of the models presented previously are deficient in various ways:
• the test activities first start after the implementation
• the connection between the various test stages and the basis for the test is not clear
• the tight link between test, debug and change tasks during the test phase is not clear
In the following, the W-model is presented. This is based on the general V-model and the disadvantages previously mentioned are removed.

The test process usually receives too little attention and usually appears as an unattractive task to be carried out after coding. In order to place testing on an equal footing, a second “V” dedicated to testing is integrated into the model. Both V together give the “W” of the W-Model.

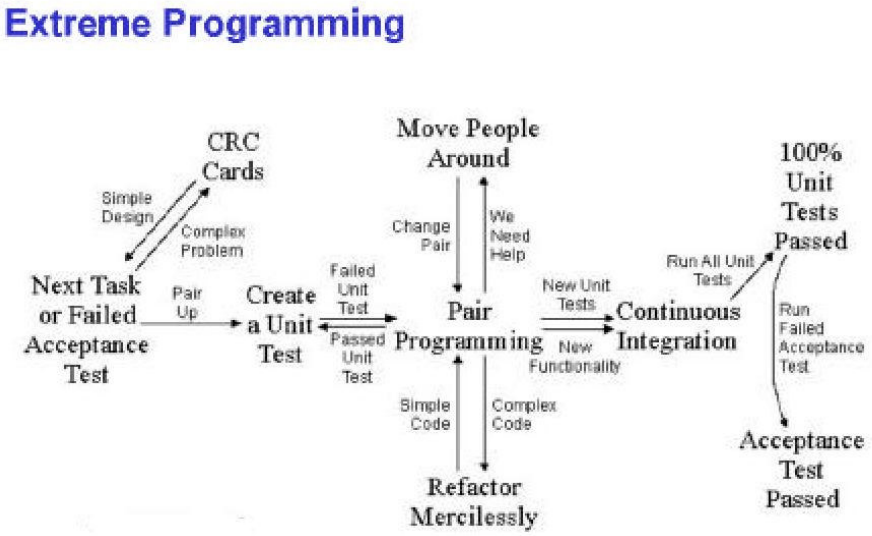
- Advantages of the W-Model
In the W-model the importance of the tests and the ordering of the individual activities for testing are clear. Parallel to the development process, in a tighter sense, a further process - the test process - is carried out. This is not first started after the development is complete.
The strict division between constructive tasks on the left-hand side and the more destructive tasks on the right-hand side that exists in the V-model is done away with. In the W-model it is clear that such a division between tasks is not sensible and that a closer co-operation between development and testing activities must exist. From the project outset onwards the testers and the developers are entrusted with tasks and are seen as an equal-rights partnership. During the test phase, the developer is responsible for the removal of defects and the correction of the implementation. The early collaboration and the tight co-operation between the two groups can often in practice avoid conflict meetings.
The W-model becomes closer to practice when the test expenditure is given 40% and more. The model clearly emphasizes the fact that testing is more than just construction, execution and evaluation of test cases.

- Disadvantages of the W-Model
Models simplify the real facts. In practice there are more relations between the different parts of a development process. However, there is a need for a simple model if all people involved in a project are to accept it. This is also a reason why the simple V-model so frequently used in practice.
The models of software development presented do not clarify the expenditure needed for resources that need to be assigned to the individual activities. Also in the W-model it appears that the different activities have an equal requirement for resources (time, personnel, etc.) In practice this is certainly not the case. In each project the most important aspects may vary and so therefore the resource allocation is unlikely to be equal across activities. For highly critical applications the test activities certainly have higher weighting or at least equal weighting with other activities.



4. Extreme Programming

   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'. 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 developed 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.



4.7 About QA Roles skills

1. What makes a good test engineer?

   A good test engineer has a 'test to break' attitude, an ability to take the point of view of the customer, a strong desire for quality, and an attention to detail. Tact and diplomacy are useful in maintaining a cooperative relationship with developers, and an ability to communicate with both technical (developers) and non-technical (customers, management) people is useful. Previous software development experience can be helpful as it provides a deeper understanding of the software development process, gives the tester an appreciation for the developers' point of view, and reduce the learning curve in automated test tool programming. Judgment skills are needed to assess high-risk areas of an application on which to focus testing efforts when time is limited.

2. What makes a good Software QA engineer?

   The same qualities a good tester has are useful for a QA engineer. Additionally, they must be able to understand the entire software development process and how it can fit into the business approach and goals of the organization. Communication skills and the ability to understand various sides of issues are important. In organizations in the early stages of implementing QA processes, patience and diplomacy are especially needed. An ability to find problems as well as to see 'what's missing' is important for inspections and reviews.

   Also it is important:

   Credibility with Programmers; independent testers often have to deal with programmers far more senior than themselves. Unless they've been through a coop program as an undergraduate, all their programming experience is with academic toys: the novice often has no real idea of what programming in a professional, cooperative, programming environment is all about. As such, they have no credibility with their programming counterpart who cans snuff off their concerns with "Look, kid. You just don't understand how programming is done here, or anywhere else, for that matter." It is setting up the novice tester for failure.

3. What makes a good QA or Test Manager?

   - Be familiar with the software development process
   - Be able to maintain enthusiasm of their team and promote a positive atmosphere, despite what is a somewhat 'negative' process (e.g., looking for or preventing problems)
   - Be able to promote teamwork to increase productivity
   - Be able to promote cooperation between software, test, and QA engineers
   - Have the diplomatic skills needed to promote improvements in QA processes
   - Have the ability to withstand pressures and say 'no' to other managers when quality is insufficient or QA processes are not being adhered to
   - Have people judgment skills for hiring and keeping skilled personnel
   - Be able to communicate with technical and non-technical people, engineers, managers, and customers.
   - Be able to run meetings and keep them focused

4. A good software tester should be Hyper-Sensitivity to Little Things

   Good testers notice little things that others (including programmers) miss or ignore. Testers see symptoms, not bugs. We know that a given bug can have many different symptoms, ranging from innocuous to catastrophic. We know that the symptoms of a bug are arbitrarily related in severity to the cause. Consequently, there is no such thing as a minor symptom-because a symptom isn't a bug. It is only after the symptom is fully explained (i.e., fully debugged) that you have the right to say if the bug that caused that symptom is minor or major. Therefore, anything at all out of the ordinary is worth pursuing. The screen flickered this time, but not last time-a bug. The keyboard is a little sticky-another bug. The account balance is off by 0.01 cents-great bug. Good testers notice such little things and use them as an entree to finding a closely-related set of inputs that will cause a catastrophic failure and therefore get the programmers' attention.

5. A good software tester should be Skeptical

   That doesn't mean hostile, though. I mean skepticism in the sense that nothing is taken for granted and that all is fit to be questioned. Only tangible evidence in documents, specifications, code, and test results matter. While they may patiently listen to the reassuring, comfortable words from the programmers ("Trust me. I know where the bugs are.")And do it with a smile-they ignore all such in-substantive assurances.

6. A good software tester should be Organized person

   Good testers use files, data bases, and all the other accouterments of an organized mind. They make up checklists to keep themselves on track. They recognize that they too can make mistakes, so they double-check their findings. They have the facts and figures to support their position. When they claim that there's a bug-believe it, because if the developers don't, the tester will flood them with well-organized, overwhelming, evidence.

   A consequence of a well-organized mind is a facility for good written and oral communications

4.8 General Questions

1. What are the five common problems in the software development process?
    a. Poor requirement; if requirements are unclear, incomplete, too general, or not testable there will be problems
    b. Unrealistic schedule; if too much work is crammed in too little time, problems are inevitable
    c. Inadequate Testing; no one will know whether or not the program is any good until the customer complains or system crash
    d. Featuritis; request to pile on new features after development is underway, extremely common
    e. Miscommunication; if developers don't know what's needed or customer's have erroneous expectations, problems are guaranteed.

2. What is the role of documentation in QA?

    Critical. (Note that documentation can be electronic, not necessarily paper.) QA practices should be documented such that they are repeatable. Specifications, designs, business rules, inspection reports, configurations, code changes, test plans, test cases, bug reports, user manuals, etc. should all be documented. There should ideally be a system for easily finding and obtaining documents and determining what documentation will have a particular piece of information. Change management for documentation should be used if possible.

3. How can Software QA process be implemented without stifling productivity?

    By implementing QA processes slowly over time, using consensus to reach 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 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.

4. What if an organization is growing so fast that fixed QA process 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 maintains focus on the customer
    - Everyone in the organization should be clear on what 'quality' means to the customer

5. How does 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.

6. How can World Wide Web sites be tested?

   Web sites 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:

   - 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 wills them 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?

   - What kinds of security (firewalls, encryptions, passwords, etc.) will be required and what is it expected to do? How can it be tested?

   - 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 variability, 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.

7. How can new Software QA processes be introduced in an existing organization?

   A lot depends on the size of the organization and the risks involved. For large organizations with high-risk (in terms of lives or property) projects, serious management buy-in is required and a formalized QA process is necessary.
   Where the risk is lower, management and organizational buy-in and QA implementation may be a slower, step-at-a-time process. QA processes should be balanced with productivity so as to keep bureaucracy from getting out of hand.
   For small groups or projects, a more ad-hoc process may be appropriate, depending on the type of customers and projects. A lot will depend on team leads or managers, feedback to developers, and ensuring adequate communications among customers, managers, developers, and testers.
   The most value for effort will often be in
   - Requirements management processes, with a goal of clear, complete, testable requirement specifications embodied in requirements or design documentation, or in 'agile'-type environments extensive continuous coordination with end-users,
   - Design inspections and code inspections, and
   - Post-mortems/retrospectives.

8. 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.

9. 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 judgment 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 time-required ratio?

10. Why Defect-Free Software is Important?

    Delivering defect-free software reduces support costs.
    Delivering defect-free software reduces programming costs.
    Delivering defect-free software reduces development time.
    Delivering defect-free software can provide a competitive advantage

5 Software Quality Assurance Lead Technical Skills Questions

The Technical knowledge for all SQALs must be enough to answer all these questions:

The QAL must have full knowledge for all previous questions driven to SQAEs; but we have to add a special domain in some other areas or situations that the SQALs may have in their daily work. The following scenarios are not related with concepts; they are related with personal attitudes, skills and experience.

5.1 General Knowledge

1. In your organization, testers are delivering code for system testing without performing unit testing. Give an example of test policy:
    a. Policy Statement
    b. Methodology
    c. Measurement
2. What three tools would you purchase for your company for use in testing? Justify the needed
3. What happens to the test plan if the application has a functionality not mentioned in the requirements?
4. Your customer does not have experience in writing Acceptance Test Plan. How will you do that in coordination with customer? What will be the contents of Acceptance Test Plan?
5. The QAI is starting a project to put the CSTE certification online. They will use an automated process for recording candidate information, scheduling candidates for exams, keeping track of results and sending out certificates. Write a brief test plan for this new project.
6. Testers in your organization are performing tests on the deliverables even after significant defects have been found. This has resulted in unnecessary testing of little value, because re-testing needs to be done after defects have been rectified. You are going to update the test plan with recommendations on when to halt testing. What recommendations are you going to make?
7. You are a tester for testing a large system. The system data model is very large with many attributes and there are a lot of inter-dependencies within the fields. What steps would you use to test the system and also what are the effects of the steps you have taken on the test plan
8. What are the various status reports you will generate to Developers and Senior Management?

5.2 Management Skills

1. What is your management style?
2. What do you think is the most difficult thing about being a manager or executive?
3. Can you explain the practice of risk management? How should risk be managed?
4. How do you prioritize testing tasks within a project?
5. What is your approach solving problem?
6. How many of the three variables scope, time and cost can be fixed by the customer?
7. How do you agree on scope and time with the customer, when the customer wants to much?
8. How exactly testing the application compatibility on different browsers and on different operating systems is done?
9. Tell me about a time when you had a lot of work to accomplish in a short time. How did you manage that situation? What was the result?
10. If your company is going to conduct a review meeting, who should be on the review committee and why?
11. You are the test lead starting on system testing. The development team says that due to a change in the requirements, they will be able to deliver the system for SQA 5 days past the deadline. You cannot change the resources (work hours, days, or test tools). What steps will you take to be able to finish the testing on time?
12. Your company is about to roll out an e-commerce application. It's not possible to test the application on all types of browsers on all platforms and operating systems. What steps would you take in the testing environment to reduce the business risks and commercial risks?
13. The project had a very high cost of testing. After going in detail, someone found out that the testers are spending their time on software that doesn't have too many defects. How will you make sure that this is correct?
14. The top management was feeling that when there are any changes in the technology being used, development schedules etc, it was a waste of time to update the Test Plan. Instead, they were emphasizing that you should put your time into testing than working on the test plan. Your Project Manager asked for your opinion. You have argued that Test Plan is very important and you need to update your test plan from time to time. It's not a waste of time and testing activities would be more effective when you have your plan clear. Use some metrics. How you would support your argument to have the test plan consistently updated all the time.
15. Testers in your organization are performing tests on the deliverables even after significant defects have been found. This has resulted in unnecessary testing of little value, because re-testing needs to be done after defects have been rectified. You are going to update the test plan with recommendations on when to halt testing. What recommendations are you going to make?

5.3 Estimations and Measurements

1. How do you measure?
    a. Test Effectiveness
    b. Test Efficiency
2. How do you estimate and measure the number of hours required for a job?
3. How will responsibilities and performance be measured?
4. How do you measure success on the job?
5. What type of metrics would you use?
6. How would you ensure 100% coverage during software testing?

5.4 Resources Management

1. What is your ideal work environment?
2. How would you build a test team?
3. How well do you work with a team?
4. You found out the senior testers are making more mistakes then junior testers; you need to communicate this aspect to the senior tester. Also, you don't want to lose this tester. How should one go about constructive criticism?
5. Tell me about a time when you had to evaluate someone who wasn't meeting the requirements of the job. How did you handle that situation? What was the outcome?
6. Think about a time when you found it necessary to verbally communicate critical information to an employee or employees. What was the issue? How did you determine the success of that communication?
7. Have you ever had to fire people? What were the reasons, and how did you handled the situation?
8. Describe some staff conflicts you have handled.
9. How do you deal with environments that are hostile to quality change efforts?