# **What is Software Testing?**

Software testing is a process of executing a program or application with the intent of finding the [**software bugs**](http://istqbexamcertification.com/what-is-defect-or-bugs-or-faults-in-software-testing/).

It can also be stated as the process of [validating](http://istqbexamcertification.com/what-is-validation-in-software-testing-or-what-is-software-validation/) and [verifying](http://istqbexamcertification.com/what-is-verification-in-software-testing-or-what-is-software-verification/) that a software program or application or product:

* Meets the business and technical requirements that guided it’s design and development
* Works as expected
* Can be implemented with the same characteristic.

# **Validation**

Determining if the system complies with the requirements and performs functions for which it is intended and meets the organization’s goals and user needs.

Am I building the right product?

# **Verification**

It makes sure that the product is designed to deliver all functionality to the customer.

Am I building the product right?

**Static Testing**

It can test and find defects without executing code. Static Testing is done during verification process. This testing includes reviewing of the documents (including source code) and static analysis. This is useful and cost effective way of testing. For example: reviewing, [**walkthrough**](http://istqbexamcertification.com/what-is-walkthrough-in-software-testing/), [**inspection**](http://istqbexamcertification.com/what-is-inspection-in-software-testing/),

**Dynamic Testing**

In dynamic testing the software code is executed to demonstrate the result of running tests. It’s done during validation process. For example: [**unit testing**](http://istqbexamcertification.com/what-is-unit-testing/), [**integration testing**](http://istqbexamcertification.com/what-is-integration-testing/), [**system testing**](http://istqbexamcertification.com/what-is-system-testing/), etc.

**Test plan**

Test plan is the project plan for the [testing work](http://istqbexamcertification.com/what-is-fundamental-test-process-in-software-testing/) to be done. 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.

**Test design**

Test design is the act of creating and writing test suites for testing a software. Specifying test cases. Test analysis and identifying test conditions gives us a generic idea for testing which covers quite a large range of possibilities. But when we come to make a test case we need to be very specific.

# **Software Testing Objectives and Purpose**

* [**Finding defects**](http://istqbexamcertification.com/what-is-defect-or-bugs-or-faults-in-software-testing/) which may get created by the programmer while developing the software.
* Gaining confidence in and providing information about the level of [**quality**](http://istqbexamcertification.com/what-is-software-quality/).
* To prevent defects.
* To make sure that the end result meets the business and user requirements.
* To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specifications.
* To gain the confidence of the customers by providing them a quality product.

# **What is Defect or bugs or faults in software testing?**

A defect is an error or a bug, in the application which is created.

When actual result deviates from the expected result while testing a software application or product then it results into a defect

These defects or bugs occur because of an error in logic or in coding which results into the [**failure**](http://istqbexamcertification.com/what-is-a-failure-in-software-testing/) or unpredicted or unanticipated results.

# **What is a Failure in software testing?**

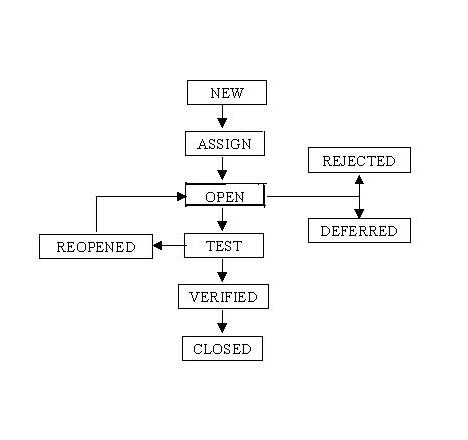
**Failure:** If under certain circumstances these defects get executed by the tester during the testing then it results into the failure which is known as software failure.

**Defect:** The bugs introduced by programmer inside the code are known as a defect. This can happen because of some programatical mistakes.

**Error:** The mistakes made by programmer is known as an ‘Error’

# **Defect Life Cycle**

Defect life cycle is a cycle which a defect goes through during its lifetime. It starts when defect is found and ends when a defect is closed, after ensuring it’s not reproduced.



# **Difference Between Severity and Priority**

1) Severity: It defines the impact that a given defect has on the system

Critical, Major, Moderate, Minor, Cosmetic

2) Priority: Priority defines the order in which we should resolve a defect

Low, Medium, High

For example:If an application or web page crashes when a remote link is clicked, in this case clicking the remote link by an user is rare but the impact of application crashing is severe. So the severity is high but priority is low.

# **Principles of Testing**

1) Testing shows presence of defects: Testing can show the [**defects**](http://istqbexamcertification.com/what-is-defect-or-bugs-or-faults-in-software-testing/) are present, but cannot prove that there are no defects.

2) Exhaustive testing is impossible: Testing everything including all combinations of inputs and preconditions is not possible. So, instead of doing the exhaustive testing we can use [**risks**](http://istqbexamcertification.com/what-is-risk-in-software-testing/) and priorities to focus testing efforts.

3) Early testing: In the [**software development life cycle**](http://istqbexamcertification.com/what-are-the-software-development-life-cycle-phases/) testing activities should start as early as possible and should be focused on defined objectives.

4) Defect clustering: A small number of modules contains most of the defects discovered during pre-release testing or shows the most operational failures.

5) Pesticide paradox: If the same kinds of tests are repeated again and again, eventually the same set of test cases will no longer be able to find any new bugs.

6) Testing is context depending: Testing is basically context dependent. Different kinds of sites are tested differently.

7) Absence – of – errors fallacy: If the system built is unusable and does not fulfil the user’s needs and expectations then finding and fixing defects does not help.

# **Fundamental Test Process**

**1)** **Planning and Control**

Test Planning

i. To determine the scope and [**risks**](http://istqbexamcertification.com/what-is-risk-in-software-testing/) and identify the objectives of testing.

ii. To determine the test approach.

iii. To implement the test policy and/or the [**test strategy**](http://istqbexamcertification.com/what-are-the-test-approaches-or-strategies-in-software-testing/).

iv. To determine the required test resources like people, test environments, PCs, etc.

v. To schedule test analysis and design tasks, test implementation, execution and evaluation.

vi. To determine the Exit criteria we need to set criteria such as Coverage criteria

Test control

i. To measure and analyze the results of reviews and testing.

ii. To monitor and document progress, [**test coverage**](http://istqbexamcertification.com/what-is-test-coverage-in-software-testing-its-advantages-and-disadvantages/) and exit criteria.

iii. To provide information on testing.

iv. To initiate corrective actions.

v. To make decisions..

**2) Analysis and Design**

i. To review the test basis. (The test basis is the information we need in order to start the test analysis and create our own test cases. Basically it’s a documentation on which test cases are based, such as requirements, design specifications, product risk analysis, architecture and interfaces..)

ii. To identify test conditions.

iii. To design the tests.

iv. To evaluate testability of the requirements and system.

v. To design the test environment set-up and identify and required infrastructure and tools.

**3)** **Implementation and Execution**

During test implementation and execution, we take the test conditions into test cases and procedures and other testware such as scripts for automation, the test environment and any other test infrastructure.

**4) Evaluating exit criteria and Reporting**

Based on the risk assessment of the project we will set the criteria for each test level against which we will measure the “enough testing”. These criteria vary from project to project and are known as exit criteria.

Exit criteria come into picture, when:

— Maximum test cases are executed with certain pass percentage.

— Bug rate falls below certain level.

— When achieved the deadlines.

Evaluating exit criteria has the following major tasks:

i. To check the test logs against the exit criteria specified in test planning.

ii. To assess if more test are needed or if the exit criteria specified should be changed.

iii. To write a test summary report for stakeholders.

**5) Test Closure activities**

Test closure activities are done when software is delivered.

i. To check which planned deliverables are actually delivered and to ensure that all incident reports have been resolved.

ii. To finalize and archive testware such as scripts, test environments, etc. for later reuse.

iii. To handover the testware to the maintenance organization. They will give support to the software.

iv To evaluate how the testing went and learn lessons for future releases and projects.

## 

# **Software Testing Life Cycle (STLC)**

Software Testing Life Cycle is a testing process which is executed in a sequence, in order to meet the quality goals.

# Software testing life cycle (STLC)

# Each of the step mentioned above has some Entry Criteria (it is a minimum set of conditions that should be met before starting the software testing) as well as Exit Criteria (it is a minimum set of conditions that should be completed in order to stop the software testing) on the basis of which it can be decided whether we can move to the next phase of Testing Life cycle or not.

### **Requirement Analysis**

This is the very first phase of Software testing Life cycle (STLC). In this phase testing team goes through the Requirement document with both Functional and non-functional details in order to identify the testable requirements.

In case of any confusion the QA team may setup a meeting with the clients and the stakeholders (Technical Leads, Business Analyst, System Architects and Client etc.) in order to clarify their doubts.

Once the QA team is clear with the requirements they will document the acceptance Criteria and get it approved by the Customers.

Activities to be done in Requirement analysis phase are given below:

* Analyzing the System Requirement specifications from the testing point of view
* Preparation of RTM that is Requirement Traceability Matrix
* Identifying the testing techniques and testing types
* Prioritizing the feature which need focused testing
* Analyzing the Automation feasibility
* Identifying the details about the testing environment where actual testing will be done

Deliverables (Outcome) of Requirement analysis phase are:

* Requirement Traceability Matrix (RTM)
* Automation feasibility report

### **Test Planning**

[**Test Planning**](http://istqbexamcertification.com/what-is-the-purpose-and-importance-of-test-plans/) phase starts soon after the completion of the Requirement Analysis phase. In this phase the QA manager or QA Lead will [**prepare the Test Plan**](http://istqbexamcertification.com/what-things-to-keep-in-mind-while-planning-tests/) and [**Test strategy**](http://istqbexamcertification.com/what-are-the-test-approaches-or-strategies-in-software-testing/) documents. As per these documents they will also come up with the testing effort estimations.

Activities to be done in Test Planning phase are given below:

* Estimation of testing effort
* Selection of Testing Approach
* Preparation of Test Plan, Test strategy documents
* Resource planning and assigning roles and responsibility to them
* Selection of Testing tool

Deliverables (Outcome) of Test Planning phase are:

* Test Plan document
* Test Strategy document
* Best suited Testing Approach
* Number of Resources, skill required and their roles and responsibilities
* Testing tool to be used

### **Test Case Development**

In this phase the QA team write test cases. They also write scripts for automation if required. Verification of both the test cases and test scripts are done by peers. Creation of Test Data is done in this phase.

Activities to be done in Test Case Development phase are given below:

* Creation of test cases
* Creation of test scripts if required
* Verification of test cases and automation scripts
* Creation of Test Data in testing environment

Deliverables (Outcome) of Test Case Development phase are:

* Test cases
* Test scripts (for automation if required)
* Test Data

### **Test Environment setup**

This phase includes the setup or installation process of software and hardware which is required for testing the application. In this phase the integration of the third party application is also carried out if required in the project.

After setting up the required software and hardware the installation of build is tested. Once the installation of build is successful and complete then the Test Data is generated.

After the creation of Test data the Smoke testing is executed on the build in order to check whether the basic functionalities are working fine or not. This phase can be done in parallel with the Test Case Development phase.

Activities to be done in Test Environment Setup phase are given below:

* As per the Requirement and Architecture document the list of required software and hardware is prepared
* Setting up of test environment
* Creation of test data
* Installation of build and execution of Smoke testing on it

Deliverables (Outcome) of Test Environment Setup phase are:

* Test Environment setup is ready
* Test Data is created
* Results of Smoke testing

### **Test Execution**

Before starting the Test Execution phase the Test Environment setup should be ready. In Test Execution phase the test cases are executed in the testing environment.

While execution of the test cases the QA team may find bugs which will be reported against that test case. This bug is fixed by the developer and is retested by the QA.

Activities to be done in Test Execution phase are given below:

* Execution of Test Cases
* Reporting test results
* Logging defects for the failed test cases
* Verification and retesting of the defect
* Closure of defects

Deliverables (Outcome) of Test Execution phase are:

* Test execution Report
* Updated test cases with results
* Bug Report

### **Test Cycle Closure**

In order to start the Test Cycle Closure activity the Test Execution phase should be completed. In Test Cycle phase the QA team will meet and discuss about the testing artifacts.

The whole intent of this discussion is to learn lessons from the bad practices. This will help in future projects.

Activities to be done in Test Cycle Closure phase are given below:

* To evaluate the test completion on the basis of Test Coverage and Software Quality
* Documentation of the learning from the project
* Analyzing the test results to find out the distribution of severe defects
* Test Closure Report preparation

Deliverables (Outcome) of Test Cycle Closure phase are:

* Report of Test Closure

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| TLC phases | Entry Criteria | Activity | Exit Criteria | Deliverables (Outcome) |
| Requirement analysis | Availability of Requirement document both Functional as well as non-functional  Architectural document of the application or the product should be available  Acceptance criteria defined and duly signed by the customers | Analysis of System Requirement specifications to understand the different business modules and it’s functionalities  To identify the user profile, user interface and user authentication  Types of tests to be performed on the application or product should be identified  Should collect the details about testing priorities  Preparation of RTM that is Requirement Traceability Matrix  Test Environment details should be identified in order to do testing  Analysis of automation possibility if it is required | RTM should be signed off  The customer should sign off on the test automation feasibility | Requirement Traceability Matrix (RTM)  Report on Automation Feasibility if it is applicable |
| Test Planning | Detailed requirement document  Requirement Traceability Matrix (RTM)  Automation Feasibility Report | Preparation of Test Plan document  Preparation of Test strategy document  To analyze the best suited testing approach for the application or product  To analyze the testing techniques and the types of testing to be carried out in order to maintain the quality  Selection of the testing tool  Estimation on the testing efforts  Resource planning as per the skill required for testing and also assigning roles and responsibility to them | Approved Test Plan document  Approved Test Strategy document  Document of Effort estimation | Test Plan document  Test Strategy document  Effort estimation document |
| Test case development | Detailed Requirement document  Test Plan and Test strategy documents  Automation Feasibility Report | Creation of test cases for all the modules or features in the application or product  Creation of automation scripts if required  Review of test cases and test automation scripts  Test data creation | Reviewed Test cases  Reviewed Test automation scripts  Test data creation ready for testing | Test cases  Test automation scripts  Test data |
| Test Environment setup | System design documents should be available  Architectural document of the application should be available  Environment set-up plan document should be available | Understanding the design and architecture of the application  Setting up the test environment  Installation of required hardware and software in order to start testing the application  Integration of any third party application (if required)  Installation of build  Creation of test data  Execution of smoke testing on the build  Accepting or rejecting the build as per the smoke test result | Environment setup is ready for testing  All the required software and hardware are installed  Build installation is complete and successful  Test data creation is complete  Smoke testing is done | Test environment along with test data  Smoke test result |
| Test Execution | Documents like RTM, Test Plan, Test strategy, Test cases and Test scripts should be ready  Test environment should be ready  Test data should be ready  Integration of third party application (if required) should be successful  Smoke testing of the application should be successful | Execution of test cases  Preparation of test result document  Logging defects for the failed test cases  Mapping of defects with the test cases  To update the test cases and test strategy if required  Fixed defects should be retested  Closure of the defects if they are working as expected  Execution of regression testing of the application or product in order to ensure its stability post defect closure | All test cases are executed  Defects are logged and tracked for closure | Completed the test case execution  Updated the test cases wherever required  Defects reported |
| Test cycle closure | All the test cases are executed and updated  Test results are documented  Defect logs are available | Evaluation of the test completion on the basis of Test Coverage and Software Quality  Preparation of Test Closure report  Analyzing the test results to find out the distribution of severe defects | Signed off Test Closure report by the client | Test closure Report |

# 

# 

# 

# 

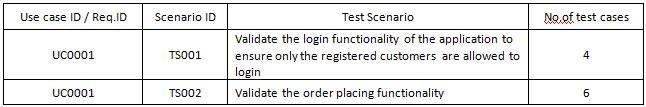
# 

# 

# Test case, Test suite, Test Scenario, Test Script

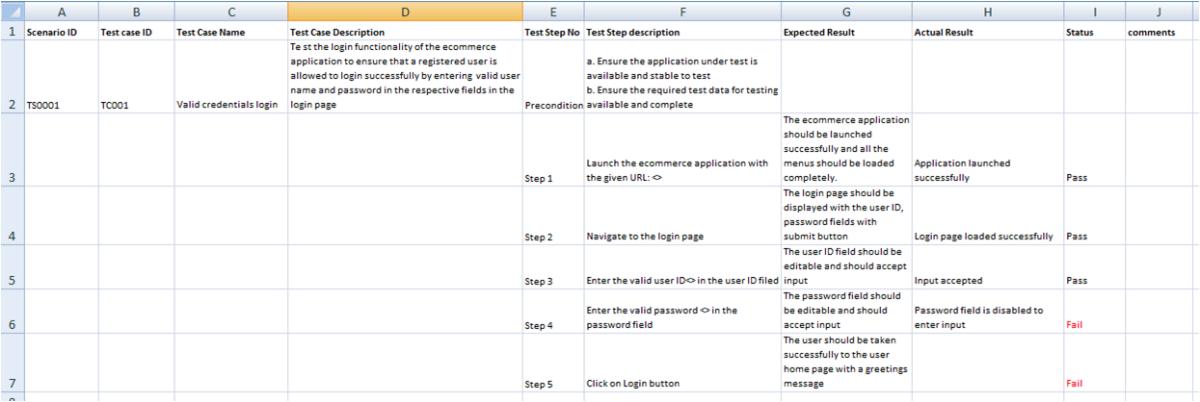
[**http://www.ibm.com/support/knowledgecenter/SS2L6K\_6.0.1/com.ibm.rational.test.qm.doc/topics/c\_testcase\_overview.html**](http://www.ibm.com/support/knowledgecenter/SS2L6K_6.0.1/com.ibm.rational.test.qm.doc/topics/c_testcase_overview.html)

# **Test Scenario**

Test scenarios are the high level classification of test requirement grouped depending on the functionality of a module.

# **Test Case**

* A test case answers the question: "What am I going to test?" You develop test cases to define the things that you must validate to ensure that the system is working correctly
* Test cases is a set of conditions under which a tester will determine whether an application is working correctly or not.
* Test cases are the set of valid and invalid executable procedure of a test scenario. A test case with valid functionality is called positive test case and a test case with invalid functionality is called negative test case.



## **Test Suites**

* A test suite is a collection of test cases that are grouped for test execution purposes.
* Test suite is a collection of test cases that are used to test a software program to show that it has some specified set of behaviours. A test suite often contains detailed instructions and information for each collection of test cases on the system configuration to be used during testing. Test suites are used to group similar test cases together

## Test suites can identify gaps in a testing effort where the successful completion of one test case must occur before you begin the next test case. For instance, you cannot add new products to a shopping cart before you successfully log in to the application.

# 

# Test suites are also useful for the following types of tests:

* Build verification tests: A collection of test cases that perform a basic validation of most the functional areas in the product. The tests are executed after each product build and before the build is promoted for use by a larger audience.
* Smoke tests: A collection of test cases that ensure basic product functionality. Typically, smoke tests are the first level of testing that is performed after changes are made to the system under test.
* End-to-End integration tests: A collection of test cases that cross product boundaries and ensure that the integration points between products are exercised and validated.
* Functional verification tests: A collection of test cases that focus on a specific product function. Executing this type of test with a test suite ensures that several aspects of a specific feature are tested.
* Regression tests: A collection of test cases that are used to make a regression pass over functional product areas.

## **Test scripts**

Each test case is typically associated with a test script, although you can run a test with no associated test script.

A test script is a manual or automated script that contains the instructions for implementing a test case. You can write manual test scripts to be run by a human tester, or you can automate some or all of the instructions in the test script.

* Manual test scripts – are the manual test cases fabricated with the multiple set of test data to enable even a layman to do the testing as per the documentation
* Automation test Script – are the programmed test cases with the combination of test data which can be executed by a tool

# 

# **Test Coverage**

Test coverage measures the amount of testing performed by a set of test.

100% coverage does *not* mean 100% tested.

# **Code Coverage**

* It creates additional test cases to increase coverage
* It helps in finding areas of a program not exercised by a set of test cases
* It helps in determining a quantitative measure of code coverage, which indirectly measure the quality of the application or product.
* One drawback of code coverage measurement is that it measures coverage of what has been written, i.e. the code itself; it cannot say anything about the software that has *not* been written.
* If a specified function has not been implemented or a function was omitted from the specification, then structure-based techniques cannot say anything about them it only looks at a structure which is already there.

# **Test analysis**

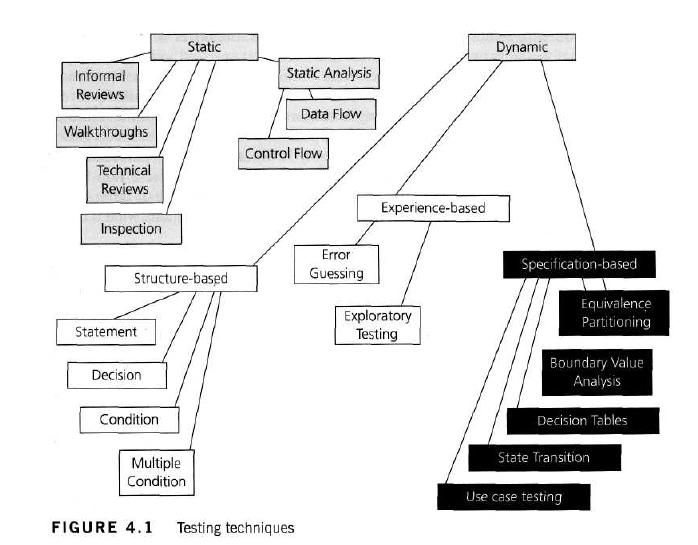
How to identify the test conditions

* Test analysis is the process of looking at something that can be used to derive test information. This basis for the tests is called the test basis.
* The test basis is the information we need in order to start the test analysis and create our own test cases. Basically it’s a documentation on which test cases are based, such as requirements, design specifications, product risk analysis, architecture and interfaces.
* From testing perspective we look at the test basis in order to see what could be tested. These are the test conditions. A test condition is simply something that we could test.
* While identifying the test conditions we want to identify as many conditions as we can and then we select about which one to take forward and combine into test cases. We could call them test possibilities.

# 

# **Test Design**

Test Design is creating a set of inputs for given software that will provide a set of expected outputs. The idea is to ensure that the system is working good enough and it can be released with as few problems as possible for the average user.



**Static Testing**

* Static testing is the testing of the software work products manually, or with a set of tools, but they are not executed.
* Most static testing techniques can be used to ‘test’ any form of document including source code, design documents and models, functional specifications and requirement specifications.

**Dynamic Testing**

* It is done during Validation process.
* The software is tested by executing it on computer.
* Example of this Dynamic Testing Technique: [**Unit testing**](http://istqbexamcertification.com/what-is-unit-testing/), [**integration testing**](http://istqbexamcertification.com/what-is-integration-testing/), [**system testing**](http://istqbexamcertification.com/what-is-system-testing/).

# **Test Strategy**

* Analytical:
  + The risk-based strategy involves performing a risk analysis using project documents and stakeholder input, then planning, estimating, designing, and prioritizing the tests based on risk.
  + The requirements-based strategy, where an analysis of the requirements specification forms the basis for planning, estimating and designing tests. Analytical test strategies have in common the use of some formal or informal analytical technique, usually during the requirements and design stages of the project.
* Model-based: For example, you can build mathematical models for loading and response for e commerce servers, and test based on that model. If the behavior of the system under test conforms to that predicted by the model, the system is deemed to be working. Model-based test strategies have in common the creation or selection of some formal or informal model for critical system behaviors, usually during the requirements and design stages of the project.
* Methodical: For example, you might have a checklist that you have put together over the years that suggests the major areas of testing to run or you might follow an industry-standard for software quality, such as ISO 9126, for your outline of major test areas. You then methodically design, implement and execute tests following this outline.
* Process – or standard-compliant: Let us take an example to understand this. You might adopt the IEEE 829 standard for your testing, using books such as [Craig, 2002] or [Drabick, 2004] to fill in the methodological gaps. Alternatively, you might adopt one of the agile methodologies such as Extreme Programming. Process- or standard-compliant strategies have in common reliance upon an externally developed approach to testing, often with little – if any – customization and may have an early or late point of involvement for testing.
* Dynamic: Let us take an example to understand this. You might create a lightweight set of testing guide lines that focus on rapid adaptation or known weaknesses in software. Dynamic strategies, such as exploratory testing, have in common concentrating on finding as many defects as possible during test execution and adapting to the realities of the system under test as it is when delivered, and they typically emphasize the later stages of testing.
* Consultative or directed: Let us take an example to understand this. You might ask the users or developers of the system to tell you what to test or even rely on them to do the testing. Consultative or directed strategies have in common the reliance on a group of non-testers to guide or perform the testing effort and typically emphasize the later stages of testing simply due to the lack of recognition of the value of early testing.
* Regression-averse: Let us take an example to understand this. You might try to automate all the tests of system functionality so that, whenever anything changes, you can re-run every test to ensure nothing has broken. Regression-averse strategies have in common a set of procedures – usually automated – that allow them to detect regression defects. A regression-averse strategy may involve automating functional tests prior to release of the function, in which case it requires early testing, but sometimes the testing is almost entirely focused on testing functions that already have been released, which is in some sense a form of post release test involvement.

# **Retesting**

Retesting is testing of a particular [bug](http://istqbexamcertification.com/what-is-defect-or-bugs-or-faults-in-software-testing/) after it has been fixed. Usually tester raises the bug when they find it while testing the product or its component. This bug is assigned to a developer and he fixes it. Post fixing the bug is assigned to the tester for its [verification](http://istqbexamcertification.com/what-is-verification-in-software-testing-or-what-is-software-verification/). This testing is known as retesting.

# **Functional Testing**

In functional testing basically the testing of the functions of component or system is done. It refers to activities that verify a specific action or function of the code. Functional test tends to answer the questions like “can the user do this” or “does this particular feature work”. This is typically described in a requirements specification or in a functional specification.

* Requirement-based testing: In this type of testing the requirements are prioritized depending on the risk criteria and accordingly the tests are prioritized. This will ensure that the most important and most critical tests are included in the testing effort.
* Business-process-based testing: In this type of testing the scenarios involved in the day-to-day business use of the system are described.

# **Non-functional testing**

In non-functional testing the quality characteristics of the component or system is tested. Non-functional refers to aspects of the software that may not be related to a specific function or user action such as scalability or security.

* + [Reliability testing](http://istqbexamcertification.com/what-is-reliability-testing-in-software/): Reliability Testing is about exercising an application so that failures are discovered and removed before the system is deployed. The purpose of reliability testing is to determine product reliability, and to determine whether the software meets the customer’s reliability requirements.
  + [Usability testing](http://istqbexamcertification.com/what-is-usability-testing-in-software-and-its-benifits-to-end-user/): In usability testing basically the testers tests the ease with which the user interfaces can be used. It tests that whether the application or the product built is user-friendly or not.

Usability testing includes the following five components:

1. Learnability: How easy is it for users to accomplish basic tasks the first time they encounter the design?
2. Efficiency: How fast can experienced users accomplish tasks?
3. Memorability: When users return to the design after a period of not using it, does the user remember enough to use it effectively the next time, or does the user have to start over again learning everything?
4. Errors: How many errors do users make, how severe are these errors and how easily can they recover from the errors?
5. Satisfaction: How much does the user like using the system?
   * [Efficiency testing](http://istqbexamcertification.com/what-is-efficiency-testing-in-software/): Efficiency testing test the amount of code and testing resources required by a program to perform a particular function. Software Test Efficiency is number of test cases executed divided by unit of time (generally per hour).
   * [Maintainability testing](http://istqbexamcertification.com/what-is-maintainability-testing-in-software/): It basically defines that how easy it is to maintain the system. This means that how easy it is to analyze, change and test the application or product.
   * [Portability testing](http://istqbexamcertification.com/what-is-portability-testing-in-software/): It refers to the process of testing the ease with which a computer software component or application can be moved from one environment to another, e.g. moving of any application from Windows 2000 to Windows XP. This is usually measured in terms of the maximum amount of effort permitted. Results are measured in terms of the time required to move the software and complete the and documentation updates.
   * [Baseline testing](http://istqbexamcertification.com/what-is-baseline-testing-in-software/): It refers to the validation of documents and specifications on which test cases would be designed. The requirement specification validation is baseline testing.
   * [Compliance testing](http://istqbexamcertification.com/what-is-compliance-testing-in-software/): It is related with the IT standards followed by the company and it is the testing done to find the deviations from the company prescribed standards.
   * [Documentation testing](http://istqbexamcertification.com/what-is-documentation-testing/): As per the IEEE Documentation describing plans for, or results of, the testing of a system or component, Types include test case specification, test incident report, test log, test plan, test procedure, test report. Hence the testing of all the above mentioned documents is known as documentation testing.
   * [Endurance testing](http://istqbexamcertification.com/what-is-endurance-testing-in-software/): Endurance testing involves testing a system with a significant load extended over a significant period of time, to discover how the system behaves under sustained use. For example, in software testing, a system may behave exactly as expected when tested for 1 hour but when the same system is tested for 3 hours, problems such as memory leaks cause the system to fail or behave randomly.
   * [Load testing](http://istqbexamcertification.com/what-is-load-testing-in-software/): A load test is usually conducted to understand the behavior of the application under a specific expected load. Load testing is performed to determine a system’s behavior under both normal and at peak conditions. It helps to identify the maximum operating capacity of an application as well as any bottlenecks and determine which element is causing degradation. E.g. If the number of users are increased then how much CPU, memory will be consumed, what is the network and bandwidth response time
   * [Performance testing](http://istqbexamcertification.com/what-is-performance-testing-in-software/): Performance testing is testing that is performed, to determine how fast some aspect of a system performs under a particular workload. It can serve different purposes like it can demonstrate that the system meets performance criteria. It can compare two systems to find which performs better. Or it can measure what part of the system or workload causes the system to perform badly.
   * [Compatibility testing](http://istqbexamcertification.com/what-is-compatibility-testing-in-software/): Compatibility testing is basically the testing of the application or the product built with the computing environment. It tests whether the application or the software product built is compatible with the hardware, operating system, database or other system software or not.
   * [Security testing](http://istqbexamcertification.com/what-is-security-testing-in-software/): Security testing is basically to check that whether the application or the product is secured or not. Can anyone came tomorrow and hack the system or login the application without any authorization. It is a process to determine that an information system protects data and maintains functionality as intended.
   * [Scalability testing](http://istqbexamcertification.com/what-is-scalability-testing-in-software/): It is the testing of a software application for measuring its capability to scale up in terms of any of its non-functional capability like load supported, the number of transactions, the data volume etc.
   * [Volume testing](http://istqbexamcertification.com/what-is-volume-testing-in-software/): Volume testing refers to testing a software application or the product with a certain amount of data. E.g., if we want to volume test our application with a specific database size, we need to expand our database to that size and then test the application’s performance on it.
   * [Stress testing](http://istqbexamcertification.com/what-is-stress-testing-in-software/): It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results. It is a form of testing that is used to determine the stability of a given system. It put greater emphasis on robustness, availability, and error handling under a heavy load, rather than on what would be considered correct behavior under normal circumstances. The goals of such tests may be to ensure the software does not crash in conditions of insufficient computational resources (such as memory or disk space).
   * [Recovery testing](http://istqbexamcertification.com/what-is-recovery-testing-in-software/): Recovery testing is done in order to check how fast and better the application can recover after it has gone through any type of crash or hardware failure etc. Recovery testing is the forced failure of the software in a variety of ways to verify that recovery is properly performed. For example, when an application is receiving data from a network, unplug the connecting cable. After some time, plug the cable back in and analyze the application’s ability to continue receiving data from the point at which the network connection got disappeared. Restart the system while a browser has a definite number of sessions and check whether the browser is able to recover all of them or not.
   * [Internationalization testing and Localization testing](http://istqbexamcertification.com/what-is-internationalization-testing-and-localization-testing-in-software/):Internationalization is a process of designing a software application so that it can be adapted to various languages and regions without any changes. Whereas Localization is a process of adapting internationalized software for a specific region or language by adding local specific components and translating text.

# **Structural testing**

* The structural testing is the testing of the structure of the system or component.
* Structural testing is often referred to as ‘white box’ or ‘glass box’ or ‘clear-box testing’ because in structural testing we are interested in what is happening ‘inside the system/application’.
* In structural testing the testers are required to have the knowledge of the internal implementations of the code. Here the testers require knowledge of how the software is implemented, how it works.
* During structural testing the tester is concentrating on how the software does it. For example, a structural technique wants to know how loops in the software are working. Different test cases may be derived to exercise the loop once, twice, and many times. This may be done regardless of the functionality of the software.
* Structural testing can be used at all levels of testing. Developers use structural testing in component testing and component integration testing, especially where there is good tool support for code coverage. Structural testing is also used in system and acceptance testing, but the structures are different. For example, the coverage of menu options or major business transactions could be the structural element in system or acceptance testing.

# **Software Testing Levels**

Testing levels are basically to identify missing areas and prevent overlap and repetition between the development life cycle phases. In software development life cycle models there are defined phases like requirement gathering and analysis, design, coding or implementation, testing and deployment. Each phase goes through the testing. Hence there are various levels of testing. The various levels of testing are:

1. [**Unit testing:**](http://istqbexamcertification.com/what-is-unit-testing/) It is basically done by the developers to make sure that their code is working fine and meet the user specifications. They test their piece of code which they have written like classes, functions, interfaces and procedures.
2. [**Component testing:**](http://istqbexamcertification.com/what-is-component-testing/) It is also called as module testing. The basic difference between the unit testing and component testing is in unit testing the developers test their piece of code but in component testing the whole component is tested. For example, in a student record application there are two modules one which will save the records of the students and other module is to upload the results of the students. Both the modules are developed separately and when they are tested one by one then we call this as a component or module testing.
3. [**Integration testing:**](http://istqbexamcertification.com/what-is-integration-testing/) testing done before, during and after integration of a new module into the main software package. This involves testing of each individual code module. One piece of software can contain several modules which are often created by several different programmers. It is crucial to test each module's effect on the entire program model.:
   * Big bang integration testing
   * Top down
   * Bottom up
   * Functional incremental
4. [**Component integration testing:**](http://istqbexamcertification.com/what-is-component-integration-testing/) In the example above when both the modules or components are integrated then the testing done is called as Component integration testing. This testing is basically done to ensure that the code should not break after integrating the two modules.
5. [**System integration testing:**](http://istqbexamcertification.com/what-is-system-integration-testing/) System integration is defined as the process of bringing together the component subsystems into one system and ensuring that the subsystems function together as a system.
6. [**System testing:**](http://istqbexamcertification.com/what-is-system-testing/) System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

System testing falls within the scope of black-box testing. As a rule, system testing takes, as its input, all of the "integrated" software components that have passed integration testing and also the software system itself integrated with any applicable hardware system(s). The purpose of integration testing is to detect any inconsistencies between the software units that are integrated together (called assemblages) or between any of the assemblages and the hardware. System testing is a more limited type of testing; it seeks to detect defects both within the "inter-assemblages" and also within the system as a whole.

Below we have listed types of system testing a large software development company would typically use:

1. **Usability Testing -** Usability testing mainly focuses on the user's ease to use the application, flexibility in handling controls and ability of the system to meet its objectives
2. **Smoke Testing:** Smoke Testing is a type of software testing that comprises of a non-exhaustive set of tests that aim at ensuring that the most important functions work. The results of this testing is used to decide if a build is stable enough to proceed with further testing.
3. **Load Testing -** Load testing is necessary to know that a software solution will perform under real life loads.
4. **Regression Testing** - Regression testing involves testing done to make sure none of the changes made over the course of the development process have caused new bugs. It also makes sure no old bugs appear from the addition of new software modules over time.
5. **Recovery Testing -** Recovery testing is done to demonstrate a software solution is reliable, trustworthy and can successfully recoup from possible crashes.
6. **Migration Testing -** Migration testing is done to ensure that the software can be moved from older system infrastructures to current system infrastructures without any issues.
7. **Functional Testing -** Also known as functional completeness testing, functional testing involves trying to think of any possible missing functions. Testers might make a list of additional functionalities that a product could have to improve it during functional testing.
8. **Hardware/Software Testing -** IBM refers to Hardware/Software testing as "HW/SW Testing". This is when the tester focuses his/her attention on the interactions between the hardware and software during system testing.
9. [**Acceptance testing:**](http://istqbexamcertification.com/what-is-acceptance-testing/) Acceptance testing are basically done to ensure that the requirements of the specification are met.
10. [**Alpha testing:**](http://istqbexamcertification.com/what-is-alpha-testing/) Alpha testing is done at the developers’ site. It is done at the end of the development process
11. [**Beta testing:**](http://istqbexamcertification.com/what-is-beta-testing/) Beta testing is done at the customers’ site. It is done just before the launch of the product.

# **Traceability in Software Testing**

Test conditions should be able to be linked back to their sources in the test basis, this is known as traceability.

* The requirements for a given function or feature have changed. Some of the fields now have different ranges that can be entered. Which tests were looking at those boundaries? They now need to be changed. How many tests will actually be affected by this change in the requirements? These questions can be answered easily if the requirements can easily be traced to the tests.
* A set of tests that has run OK in the past has now started creating serious problems. What functionality do these tests actually exercise? Traceability between the tests and the requirement being tested enables the functions or features affected to be identified more easily.

# **V-model**