

**DEBASIS PRADHAN**



# ***Software Testing*** **COMPLETE GUIDE**

***Software Testing Tricks***

Get More Tutorials @  
[SoftwareTestingTricks.com](http://SoftwareTestingTricks.com)

# Table of Content

[What is testing?](#)

[Who does testing?](#)

[When to Start Testing?](#)

[When to Stop Testing?](#)

[Verification & Validation](#)

[Testing, Quality Assurance and Quality Control](#)

[Testing and Debugging](#)

[Audit and Inspection](#)

[Testing Types](#)

[Manual testing](#)

[Automation testing](#)

[When to Automate](#)

[How to Automate?](#)

[Some Popular Automation Tools](#)

[Testing Methods](#)

[Black Box Testing](#)

[White Box Testing](#)

[Grey Box Testing](#)

[Black Box vs Grey Box vs White Box](#)

[Functional Testing](#)

[Unit Testing](#)

[Integration Testing](#)

[System Testing](#)

[Regression Testing](#)

[Acceptance Testing](#)

[Non-Functional Testing](#)

[Performance Testing](#)

[Usability Testing](#)

[Security Testing](#)

[Portability Testing](#)

[Software Testing Documentation](#)

***Software Testing Tricks***

Get More Tutorials @  
[SoftwareTestingTricks.com](http://SoftwareTestingTricks.com)

[Test Plan](#)

[Test Scenario](#)

[Test Case](#)

[Traceability Matrix](#)

[Test Estimation Techniques](#)

[Functional Point Analysis](#)

[Test Point Analysis](#)

[Mark-II method](#)

[Miscellaneous](#)

[Conclusion](#)

# What is testing?

Testing is the process of evaluating a system or its component(s) with the intent to find that whether it satisfies the specified requirements or not. This activity results in the actual, expected and difference between their results. In simple words testing is executing a system in order to identify any gaps, errors or missing requirements in contrary to the actual desire or requirements.

According to ANSI/IEEE 1059 standard, Testing can be defined as *A process of analysing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.*

## Who does testing?

It depends on the process and the associated stakeholders of the project(s). In the IT industry, large companies have a team with responsibilities to evaluate the developed software in the context of the given requirements. Moreover, developers also conduct testing which is called Unit Testing. In most cases, following professionals are involved in testing of a system within their respective capacities:

- Software Tester
- Software Developer
- Project Lead/Manager
- End User

Different companies have difference designations for people who test the software on the basis of their experience and knowledge such as Software Tester, Software Quality Assurance Engineer, and QA Analyst etc.

It is not possible to test the software at any time during its cycle. The next two sections state when testing should be started and when to end it during the SDLC.

## When to Start Testing?

An early start to testing reduces the cost, time to rework and error free software that is delivered to the client. However in Software Development Life Cycle (SDLC) testing can be started from the Requirements Gathering phase and lasts till the deployment of the software. However it also depends on the development model that is being used. For example in Water fall model formal testing is conducted in the Testing phase, but in incremental model, testing is performed at the end of every increment/iteration and at the end the whole application is tested.

Testing is done in different forms at every phase of SDLC like during Requirement gathering phase, the analysis and verifications of requirements are also considered testing. Reviewing

the design in the design phase with intent to improve the design is also considered as testing. Testing performed by a developer on completion of the code is also categorized as Unit type of testing.

## When to Stop Testing?

Unlike when to start testing it is difficult to determine when to stop testing, as testing is a never ending process and no one can say that any software is 100% tested. Following are the aspects which should be considered to stop the testing:

- Testing Deadlines.
- Completion of test case execution.
- Completion of Functional and code coverage to a certain point.
- Bug rate falls below a certain level and no high priority bugs are identified.
- Management decision.

## Verification & Validation

S.N.	Verification	Validation
1	Are you building it right?	Are you building the right thing?
2	Ensure that the software system meets all the functionality.	Ensure that functionalities meet the intended behavior.
3	Verification takes place first and includes the checking for documentation, code etc.	Validation occurs after verification and mainly involves the checking of the overall product.
4	Done by developers.	Done by Testers.
5	Have static activities as it includes the reviews, walkthroughs, and inspections to verify that software is correct or not.	Have dynamic activities as it includes executing the software against the requirements.
6	It is an objective process and no subjective decision should be needed to verify the Software.	It is a subjective process and involves subjective decisions on how well the Software works.

## Testing, Quality Assurance and Quality Control

Most people are confused with the concepts and difference between Quality Assurance, Quality Control and Testing. Although they are interrelated and at some level they can be considered as the same activities, but there is indeed a difference between them. Mentioned below are the definitions and differences between them:

S.N.	Quality Assurance	Quality Control	Testing
1	Activities which ensure the implementation of processes, procedures and standards in context to verification of developed software and intended requirements.	Activities which ensure the verification of developed software with respect to documented (or not in some cases) requirements.	Activities which ensure the identification of bugs/error/defects in the Software.
2	Focuses on processes and procedures rather than conducting actual testing on the system.	Focuses on actual testing by executing Software with intent to identify bug/defect through implementation of procedures and process.	Focuses on actual testing.
3	Process oriented activities.	Product oriented activities.	Product oriented activities.
4	Preventive activities.	It is a corrective process.	It is a preventive process.
5	It is a subset of Software Test Life Cycle (STLC).	QC can be considered as the subset of Quality Assurance.	Testing is the subset of Quality Control.

## Testing and Debugging

### Testing:

It involves the identification of bug/error/defect in the software without correcting it. Normally professionals with a Quality Assurance background are involved in the identification of bugs. Testing is performed in the testing phase.

### Debugging:

It involves identifying, isolating and fixing the problems/bug. Developers who code the software conduct debugging upon encountering an error in the code. Debugging is the part of White box or Unit Testing. Debugging can be performed in the development phase while conducting Unit Testing or in phases while fixing the reported bugs.

## Audit and Inspection

### Audit:

A systematic process to determine how the actual testing process is conducted within an organization or a team. Generally, it is an independent examination of processes which are involved during the testing of software. As per IEEE, it is a review of documented processes whether organizations implements and follows the processes or not. Types of Audit include the Legal Compliance Audit, Internal Audit, and System Audit.

### Inspection:

A formal technique which involves the formal or informal technical reviews of any artifact by identifying any error or gap. Inspection includes the formal as well as informal technical reviews. As per IEEE94, Inspection is a formal evaluation technique in which software

requirements, design, or code are examined in detail by a person or group other than the author to detect faults, violations of development standards, and other problems.

Formal Inspection meetings may have following process: Planning, Overview Preparation, Inspection Meeting, Rework, and Follow-up.

## Testing Types

### ***Manual testing***

This type includes the testing of the Software manually i.e. without using any automated tool or any script. In this type the tester takes over the role of an end user and test the Software to identify any un-expected behavior or bug. There are different stages for manual testing like unit testing, Integration testing, System testing and User Acceptance testing.

Testers use test plan, test cases or test scenarios to test the Software to ensure the completeness of testing. Manual testing also includes exploratory testing as testers explore the software to identify errors in it.

### ***Automation testing***

Automation testing which is also known as *Test Automation*, is when the tester writes scripts and uses another software to test the software. This process involves automation of a manual process. Automation Testing is used to re-run the test scenarios that were performed manually, quickly and repeatedly.

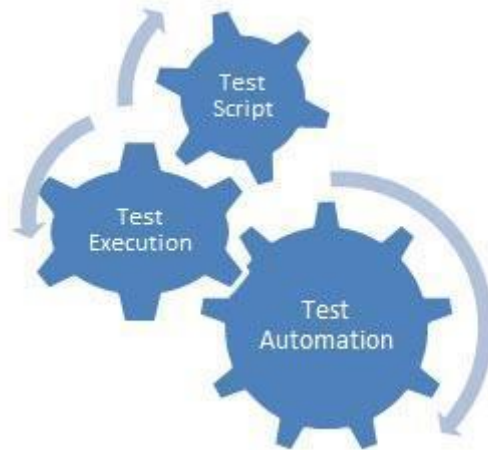
Apart from regression testing, Automation testing is also used to test the application from load, performance and stress point of view. It increases the test coverage; improve accuracy, saves time and money in comparison to manual testing.

### ***When to Automate?***

Test Automation should be uses by considering the following for the Software:

- Large and critical projects.
- Projects that require testing the same areas frequently.
- Requirements not changing frequently.
- Accessing the application for load and performance with many virtual users.
- Stable Software with respect to manual testing.
- Availability of time.





## ***How to Automate?***

Automation is done by using a supportive computer language like vb scripting and an automated software application. There are a lot of tools available which can be use to write automation scripts. Before mentioning the tools lets identify the process which can be used to automate the testing:

- Identifying areas within a software for automation.
- Selection of appropriate tool for Test automation.
- Writing Test scripts.
- Development of Test suits.
- Execution of scripts.
- Create result reports.
- Identify any potential bug or performance issue.

## ***Some Popular Automation Tools***

Following are the tools which can be use for Automation testing:

- HP Quick Test Professional
- Selenium
- IBM Rational Functional Tester
- SilkTest
- TestComplete
- Testing Anywhere
- WinRunner
- LaodRunner
- Visual Studio Test Professional
- WATIR



# Testing Methods

## ***Black Box Testing***

The technique of testing without having any knowledge of the interior workings of the application is Black Box testing. The tester is oblivious to the system architecture and does not have access to the source code. Typically, when performing a black box test, a tester will interact with the system's user interface by providing inputs and examining outputs without knowing how and where the inputs are worked upon.

Advantages	Disadvantages
<ul style="list-style-type: none"><li>■ Well suited and efficient for large code segments.</li><li>■ Code Access not required.</li><li>■ Clearly separates user's perspective from the developer's perspective through visibly defined roles.</li><li>■ Large numbers of moderately skilled testers can test the application with no knowledge of implementation, programming language or operating systems.</li></ul>	<ul style="list-style-type: none"><li>■ Limited Coverage since only a selected number of test scenarios are actually performed.</li><li>■ Inefficient testing, due to the fact that the tester only has limited knowledge about an application.</li><li>■ Blind Coverage, since the tester cannot target specific code segments or error prone areas.</li><li>■ The test cases are difficult to design.</li></ul>

## ***White Box Testing***

White box testing is the detailed investigation of internal logic and structure of the code. White box testing is also called glass testing or open box testing.

Advantages	Disadvantages
<ul style="list-style-type: none"><li>■ As the tester has knowledge of the source code, it becomes very easy to find out which type of data can help in testing the application effectively.</li><li>■ It helps in optimizing the code.</li><li>■ Extra lines of code can be removed which can bring in hidden defects.</li><li>■ Due to the tester's knowledge about the code, maximum coverage is attained during test scenario writing.</li></ul>	<ul style="list-style-type: none"><li>■ Due to the fact that a skilled tester is needed to perform white box testing, the costs are increased.</li><li>■ Sometimes it is impossible to look into every nook and corner to find out hidden errors that may create problems as many paths will go untested.</li><li>■ It is difficult to maintain white box testing as the use of specialized tools like code analyzers and debugging tools are required.</li></ul>

In order to perform white box testing on an application, the tester needs to possess knowledge of the internal working of the code. The tester needs to have a look inside the source code and find out which unit/chunk of the code is behaving inappropriately.

## Grey Box Testing

Grey Box testing is a technique to test the application with limited knowledge of the internal workings of an application. In software testing, the term *the more you know the better* carries a lot of weight when testing an application.

Mastering the domain of a system always gives the tester an edge over someone with limited domain knowledge. Unlike black box testing, where the tester only tests the application's user interface, in grey box testing, the tester has access to design documents and the database. Having this knowledge, the tester is able to better prepare test data and test scenarios when making the test plan.

Advantages	Disadvantages
<ul style="list-style-type: none"><li>■ Offers combined benefits of black box and white box testing wherever possible.</li><li>■ Grey box testers don't rely on the source code; instead they rely on interface definition and functional specifications.</li><li>■ Based on the limited information available, a grey box tester can design excellent test scenarios especially around communication protocols and data type handling.</li><li>■ The test is done from the point of view of the user and not the designer.</li></ul>	<ul style="list-style-type: none"><li>■ Since the access to source code is not available, the ability to go over the code and test coverage is limited.</li><li>■ The tests can be redundant if the software designer has already run a test case.</li><li>■ Testing every possible input stream is unrealistic because it would take an unreasonable amount of time; therefore, many program paths will go untested.</li></ul>

## Black Box vs Grey Box vs White Box

S.N.	Black Box Testing	Grey Box Testing	White Box Testing
1	The Internal Workings of an application are not required to be known	Somewhat knowledge of the internal workings are known	Tester has full knowledge of the Internal workings of the application
2	Also known as closed box testing, data driven testing and functional testing	Another term for grey box testing is translucent testing as the tester has limited knowledge of the insides of the application	Also known as clear box testing, structural testing or code based testing
3	Performed by end users and also by testers and developers	Performed by end users and also by testers and developers	Normally done by testers and developers
4	Testing is based on external expectations - Internal behavior of the application is unknown	Testing is done on the basis of high level database diagrams and data flow diagrams	Internal workings are fully known and the tester can design test data accordingly
5	This is the least time consuming and exhaustive	Partly time consuming and exhaustive	The most exhaustive and time consuming type of testing
6	Not suited to algorithm testing	Not suited to algorithm testing	Suited for algorithm testing
7	This can only be done by trial and error method	Data domains and Internal boundaries can be tested, if known	Data domains and Internal boundaries can be better tested

## Functional Testing

This is a type of black box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional Testing of the software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

There are five steps that are involved when testing an application for functionality.

Steps	Description
I	The determination of the functionality that the intended application is meant to perform.
II	The creation of test data based on the specifications of the application.
III	The output based on the test data and the specifications of the application.
IV	The writing of Test Scenarios and the execution of test cases.
V	The comparison of actual and expected results based on the executed test cases.

An effective testing practice will see the above steps applied to the testing policies of every organization and hence it will make sure that the organization maintains the strictest of standards when it comes to software quality.

## ***Unit Testing***

This type of testing is performed by the developers before the setup is handed over to the testing team to formally execute the test cases. Unit testing is performed by the respective developers on the individual units of source code assigned areas. The developers use test data that is separate from the test data of the quality assurance team.

The goal of unit testing is to isolate each part of the program and show that individual parts are correct in terms of requirements and functionality.

### **Limitations of Unit Testing**

Testing cannot catch each and every bug in an application. It is impossible to evaluate every execution path in every software application. The same is the case with unit testing.

There is a limit to the number of scenarios and test data that the developer can use to verify the source code. So after he has exhausted all options there is no choice but to stop unit testing and merge the code segment with other units.

## ***Integration Testing***

The testing of combined parts of an application to determine if they function correctly together is Integration testing. There are two methods of doing Integration Testing Bottom-up Integration testing and Top Down Integration testing.

S.N.	Integration Testing Method
1	<b>Bottom-up integration</b> This testing begins with unit testing, followed by tests of progressively higher-level combinations of units called modules or builds.
2	<b>Top-Down integration</b> This testing, the highest-level modules are tested first and progressively lower-level modules are tested after that.

In a comprehensive software development environment, bottom-up testing is usually done first, followed by top-down testing. The process concludes with multiple tests of the complete application, preferably in scenarios designed to mimic those it will encounter in customers' computers, systems and network.

## ***System Testing***

This is the next level in the testing and tests the system as a whole. Once all the components are integrated, the application as a whole is tested rigorously to see that it meets Quality Standards. This type of testing is performed by a specialized testing team.

System testing is so important because of the following reasons:

- System Testing is the first step in the Software Development Life Cycle, where the application is tested as a whole.
- The application is tested thoroughly to verify that it meets the functional and technical specifications.
- The application is tested in an environment which is very close to the production environment where the application will be deployed.
- System Testing enables us to test, verify and validate both the business requirements as well as the Applications Architecture.

## ***Regression Testing***

Whenever a change in a software application is made it is quite possible that other areas within the application have been affected by this change. To verify that a fixed bug hasn't resulted in another functionality or business rule violation is Regression testing. The intent of Regression testing is to ensure that a change, such as a bug fix did not result in another fault being uncovered in the application.

Regression testing is so important because of the following reasons:

- Minimize the gaps in testing when an application with changes made has to be tested.
- Testing the new changes to verify that the change made did not affect any other area of the application.
- Mitigates Risks when regression testing is performed on the application.
- Test coverage is increased without compromising timelines.
- Increase speed to market the product.

## ***Acceptance Testing***

This is arguably the most importance type of testing as it is conducted by the Quality Assurance Team who will gauge whether the application meets the intended specifications and satisfies the client's requirements. The QA team will have a set of pre written scenarios and Test Cases that will be used to test the application.

More ideas will be shared about the application and more tests can be performed on it to gauge its accuracy and the reasons why the project was initiated. Acceptance tests are not only intended to point out simple spelling mistakes, cosmetic errors or Interface gaps, but

also to point out any bugs in the application that will result in system crashers or major errors in the application.

By performing acceptance tests on an application the testing team will deduce how the application will perform in production. There are also legal and contractual requirements for acceptance of the system.

### **Alpha Testing**

This test is the first stage of testing and will be performed amongst the teams (developer and QA teams). Unit testing, integration testing and system testing when combined are known as alpha testing. During this phase, the following will be tested in the application:

- Spelling Mistakes
- Broken Links
- Cloudy Directions
- The Application will be tested on machines with the lowest specification to test loading times and any latency problems.

### **Beta Testing**

This test is performed after Alpha testing has been successfully performed. In beta testing a sample of the intended audience tests the application. Beta testing is also known as pre-release testing. Beta test versions of software are ideally distributed to a wide audience on the Web, partly to give the program a "real-world" test and partly to provide a preview of the next release. In this phase the audience will be testing the following:

- Users will install, run the application and send their feedback to the project team.
- Typographical errors, confusing application flow, and even crashes.
- Getting the feedback, the project team can fix the problems before releasing the software to the actual users.
- The more issues you fix that solve real user problems, the higher the quality of your application will be.
- Having a higher-quality application when you release to the general public will increase customer satisfaction.

## **Non-Functional Testing**

This section is based upon the testing of the application from its non-functional attributes. Non-functional testing of Software involves testing the Software from the requirements which are non functional in nature related but important a well such as performance, security, user interface etc.

Some of the important and commonly used non-functional testing types are mentioned as follows:



## ***Performance Testing***

It is mostly used to identify any bottlenecks or performance issues rather than finding the bugs in software. There are different causes which contribute in lowering the performance of software:

- Network delay.
- Client side processing.
- Database transaction processing.
- Load balancing between servers.
- Data rendering.

Performance testing is considered as one of the important and mandatory testing type in terms of following aspects:

- Speed (i.e. Response Time, data rendering and accessing)
- Capacity
- Stability
- Scalability

It can be either qualitative or quantitative testing activity and can be divided into different sub types such as **Load testing** and **Stress testing**.

### **Load Testing**

A process of testing the behavior of the Software by applying maximum load in terms of Software accessing and manipulating large input data. It can be done at both normal and peak load conditions. This type of testing identifies the maximum capacity of Software and its behavior at peak time.

Most of the time, Load testing is performed with the help of automated tools such as Load Runner, AppLoader, IBM Rational Performance Tester, Apache JMeter, Silk Performer, Visual Studio Load Test etc.

Virtual users (VUsers) are defined in the automated testing tool and the script is executed to verify the Load testing for the Software. The quantity of users can be increased or decreased concurrently or incrementally based upon the requirements.

### **Stress Testing**

This testing type includes the testing of Software behavior under abnormal conditions. Taking away the resources, applying load beyond the actual load limit is Stress testing.

The main intent is to test the Software by applying the load to the system and taking over the resources used by the Software to identify the breaking point. This testing can be performed by testing different scenarios such as:

- Shutdown or restart of Network ports randomly.
- Turning the database on or off.



- Running different processes that consume resources such as CPU, Memory, server etc.

## ***Usability Testing***

This section includes different concepts and definitions of Usability testing from Software point of view. It is a black box technique and is used to identify any error(s) and improvements in the Software by observing the users through their usage and operation.

According to Nielsen, Usability can be defined in terms of five factors i.e. Efficiency of use, Learn-ability, Memor-ability, Errors/safety, satisfaction. According to him the usability of the product will be good and the system is usable if it possesses the above factors.

Nigel Bevan and Macleod considered that Usability is the quality requirement which can be measured as the outcome of interactions with a computer system. This requirement can be fulfilled and the end user will be satisfied if the intended goals are achieved effectively with the use of proper resources.

Molich in 2000 stated that user friendly system should fulfil the following five goals i.e. Easy to Learn, Easy to Remember, Efficient to Use, Satisfactory to Use and Easy to Understand.

In addition to different definitions of usability, there are some standards and quality models and methods which define the usability in the form of attributes and sub attributes such as ISO-9126, ISO-9241-11, ISO-13407 and IEEE std.610.12 etc.

### **UI vs Usability Testing**

UI testing involves the testing of Graphical User Interface of the Software. This testing ensures that the GUI should be according to requirements in terms of color, alignment, size and other properties.

On the other hand Usability testing ensures that a good and user friendly GUI is designed and is easy to use for the end user. UI testing can be considered as a sub part of Usability testing.

## ***Security Testing***

Security testing involves the testing of Software in order to identify any flaws and gaps from security and vulnerability point of view. Following are the main aspects which Security testing should ensure:

- Confidentiality.
- Integrity.
- Authentication.
- Availability.
- Authorization.
- Non-repudiation.
- Software is secure against known and unknown vulnerabilities.

- Software data is secure.
- Software is according to all security regulations.
- Input checking and validation.
- SQL insertion attacks.
- Injection flaws.
- Session management issues.
- Cross-site scripting attacks.
- Buffer overflows vulnerabilities.
- Directory traversal attacks.

## ***Portability Testing***

Portability testing includes the testing of Software with intend that it should be re-useable and can be moved from another Software as well. Following are the strategies that can be used for Portability testing.

- Transferred installed Software from one computer to another.
- Building executable (.exe) to run the Software on different platforms.

Portability testing can be considered as one of the sub parts of System testing, as this testing type includes the overall testing of Software with respect to its usage over different environments. Computer Hardware, Operating Systems and Browsers are the major focus of Portability testing. Following are some pre-conditions for Portability testing:

- Software should be designed and coded, keeping in mind Portability Requirements.
- Unit testing has been performed on the associated components.
- Integration testing has been performed.
- Test environment has been established.

## **Software Testing Documentation**

Testing documentation involves the documentation of artifacts which should be developed before or during the testing of Software.

Documentation for Software testing helps in estimating the testing effort required, test coverage, requirement tracking/tracing etc. This section includes the description of some commonly used documented artifacts related to Software testing such as:

- Test Plan
- Test Scenario
- Test Case
- Traceability Matrix

## Test Plan

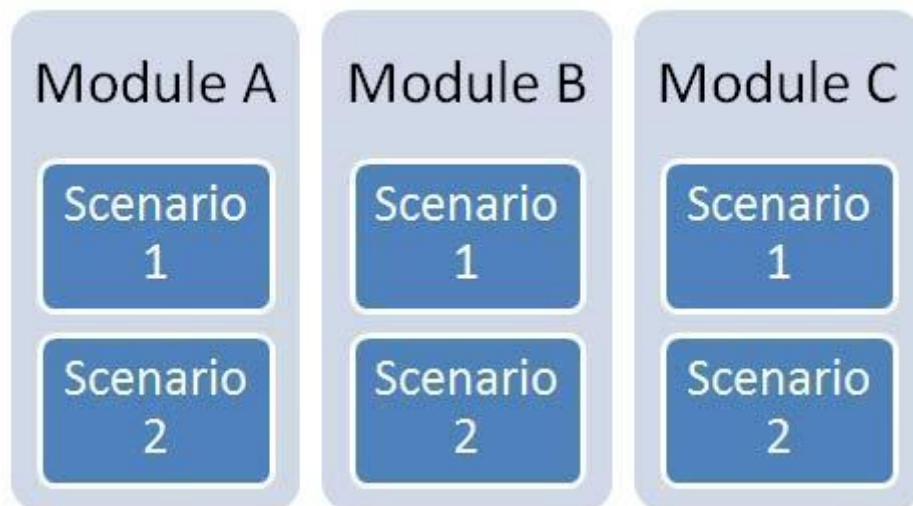
A test plan outlines the strategy that will be used to test an application, the resources that will be used, the test environment in which testing will be performed, the limitations of the testing and the schedule of testing activities. Typically the Quality Assurance Team Lead will be responsible for writing a Test Plan.

A test plan will include the following.

- Introduction to the Test Plan document
- Assumptions when testing the application
- List of test cases included in Testing the application
- List of features to be tested
- What sort of Approach to use when testing the software
- List of Deliverables that need to be tested
- The resources allocated for testing the application
- Any Risks involved during the testing process
- A Schedule of tasks and milestones as testing is started

## Test Scenario

A one line statement that tells what area in the application will be tested. Test Scenarios are used to ensure that all process flows are tested from end to end. A particular area of an application can have as little as one test scenario to a few hundred scenarios depending on the magnitude and complexity of the application.



The term test scenario and test cases are used interchangeably however the main difference being that test scenarios has several steps however test cases have a single step. When viewed from this perspective test scenarios are test cases, but they include several test

cases and the sequence that they should be executed. Apart from this, each test is dependent on the output from the previous test.

## ***Test Case***

Test cases involve the set of steps, conditions and inputs which can be used while performing the testing tasks. The main intent of this activity is to ensure whether the Software Passes or Fails in terms of its functionality and other aspects. There are many types of test cases like: functional, negative, error, logical test cases, physical test cases, UI test cases etc.

Furthermore test cases are written to keep track of testing coverage of Software. Generally, there is no formal template which is used during the test case writing. However, following are the main components which are always available and included in every test case:

- Test case ID.
- Product Module.
- Product version.
- Revision history.
- Purpose
- Assumptions
- Pre-Conditions.
- Steps.
- Expected Outcome.
- Actual Outcome.
- Post Conditions.

Many Test cases can be derived from a single test scenario. In addition to this, some time it happened that multiple test cases are written for single Software which is collectively known as test suites.

## ***Traceability Matrix***

Traceability Matrix (also known as Requirement Traceability Matrix - RTM) is a table which is used to trace the requirements during the Software development life Cycle. It can be used for forward tracing (i.e. from Requirements to Design or Coding) or backward (i.e. from Coding to Requirements). There are many user defined templates for RTM.

Each requirement in the RTM document is linked with its associated test case, so that testing can be done as per the mentioned requirements. Furthermore, Bug ID is also include and linked with its associated requirements and test case. The main goals for this matrix are:

- Make sure Software is developed as per the mentioned requirements.
- Helps in finding the root cause of any bug.

- Helps in tracing the developed documents during different phases of SDLC.

## Test Estimation Techniques

Estimating effort for test is one of the major and important tasks in SDLC. Correct estimation helps in testing the Software with maximum coverage. This section describes some of the techniques which can be useful during the estimating of effort for testing.

### ***Functional Point Analysis***

This method is based on the analysis of functional user requirements of the Software with following categories:

- Outputs
- Inquiries
- Inputs
- Internal files
- External files

### ***Test Point Analysis***

This estimation process is used for function point analysis for Black box or Acceptance testing. It is use the main elements of this method are: Size, Productivity, Strategy, Interfacing, Complexity and Uniformity etc.

### ***Mark-II method***

It is estimation method used for analysis and measuring the estimation based on end user functional view. The procedure for Mark-II method is:

- Determine the View Point
- Purpose and Type of Count
- Define the Boundary of Count
- Identify the Logical transactions
- Identify and Categorize Data Entity Types
- Count the Input Data Element Types
- Count the Functional Size

### ***Miscellaneous***

You can use other popular estimation techniques like:

- Delphi Technique
- Analogy Based Estimation
- Test Case Enumeration Based Estimation
- Task (Activity) based Estimation and IFPUG method

# Conclusion

You just got a quick overview of Software Testing using this short guide. This guide has been designed from the tutorial given at the site [SoftwareTestingTricks.com](https://SoftwareTestingTricks.com) and there is lot of content which has been cut short to make it a quick guide. So if you are interested in full content then you can go through the complete tutorial available at the site.