**MANUAL TESTING**

**What is Software Testing? Introduction, Definition, Basics & Types**

**What is Software Testing?**

**SOFTWARE TESTING** is defined as an activity to check **whether the actual results match the expected results** and to ensure that the software system is[Defect](https://www.guru99.com/defect-management-process.html)free.

Software testing also helps to identify errors, gaps or missing requirements in contrary to the actual requirements. It can be either done manually or using automated tools. Some prefer saying Software testing as a [White Box](https://www.guru99.com/white-box-testing.html) and [Black Box Testing](https://www.guru99.com/black-box-testing.html).

In simple terms, Software Testing means Verification of Application Under Test (AUT).

**Why is Software Testing Important?**

Testing is important because software bugs could be expensive or even dangerous. Software bugs can potentially cause monetary and human loss, and history is full of such examples.

* In April 2015, Bloomberg terminal in London crashed due to software glitch affected more than 300,000 traders on financial markets. It forced the government to postpone a 3bn pound debt sale.
* Nissan cars have to recall over 1 million cars from the market due to software failure in the airbag sensory detectors. There has been reported two accident due to this software failure.
* Starbucks was forced to close about 60 percent of stores in the U.S and Canada due to software failure in its POS system. At one point store served coffee for free as they unable to process the transaction.
* Some of the Amazon’s third party retailers saw their product price is reduced to 1p due to a software glitch. They were left with heavy losses.
* Vulnerability in Window 10. This bug enables users to escape from security sandboxes through a flaw in the win32k system.
* In 2015 fighter plane F-35 fell victim to a software bug, making it unable to detect targets correctly.
* China Airlines Airbus A300 crashed due to a software bug on April 26, 1994, killing 264 innocent live
* In 1985, Canada's Therac-25 radiation therapy machine malfunctioned due to software bug and delivered lethal radiation doses to patients, leaving 3 people dead and critically injuring 3 others.
* In April of 1999, a software bug caused the failure of a $1.2 billion military satellite launch, the costliest accident in history
* In may of 1996, a software bug caused the bank accounts of 823 customers of a major U.S. bank to be credited with 920 million US dollars.

**Types of Software Testing**

Typically Testing is classified into three categories.

* Functional Testing
* Non-Functional Testing or [Performance Testing](https://www.guru99.com/performance-testing.html)
* Maintenance (Regression and Maintenance)

|  |  |
| --- | --- |
| **Testing Category** | **Types of Testing** |
| Functional Testing |  [Unit Testing](https://www.guru99.com/unit-testing-guide.html)   [Integration Testing](https://www.guru99.com/integration-testing.html)   Smoke   UAT ( User Acceptance Testing)   Localization   Globalization   Interoperability   So on |
| Non-Functional Testing |  Performance   Endurance   Load   Volume   Scalability   Usability   So on |
| Maintenance |  Regression   Maintenance |

## Here are the 7 Principles:

## 1) Exhaustive testing is not possible

Yes! Exhaustive testing is not possible. Instead, we need the optimal amount of testing based on the risk assessment of the application.

And the million dollar question is, how do you determine this risk?

To answer this let's do an exercise

In your opinion, Which operation is most likely to cause your Operating system to fail?

I am sure most of you would have guessed, Opening 10 different application all at the same time.

So if you were testing this Operating system, you would realize that defects are likely to be found in multi-tasking activity and need to be tested thoroughly which brings us to our next principle[Defect](https://www.guru99.com/defect-management-process.html)Clustering

## 2) Defect Clustering

Defect Clustering which states that a small number of modules contain most of the defects detected. This is the application of the Pareto Principle to software testing: approximately 80% of the problems are found in 20% of the modules.

By experience, you can identify such risky modules. But this approach has its own problems

If the same tests are repeated over and over again, eventually the same test cases will no longer find new bugs.

## 3) Pesticide Paradox

Repetitive use of the same pesticide mix to eradicate insects during farming will over time lead to the insects developing resistance to the pesticide Thereby ineffective of pesticides on insects. The same applies to software testing. If the same set of repetitive tests are conducted, the method will be useless for discovering new defects.

To overcome this, the test cases need to be regularly reviewed & revised, adding new & different test cases to help find more defects.

Testers cannot simply depend on existing test techniques. He must look out continually to improve the existing methods to make testing more effective. But even after all this sweat & hard work in testing, you can never claim your product is bug-free. To drive home this point, let's see this video of the public launch of Windows 98

You think a company like MICROSOFT would not have tested their OS thoroughly & would risk their reputation just to see their OS crashing during its public launch!

## 4) Testing shows a presence of defects

Hence, testing principle states that - Testing talks about the presence of defects and don’t talk about the absence of defects. i.e. Software Testing reduces the probability of undiscovered defects remaining in the software but even if no defects are found, it is not a proof of correctness.

But what if, you work extra hard, taking all precautions & make your software product 99% bug-free. And the software does not meet the needs & requirements of the clients.

This leads us to our next principle, which states that- Absence of Error

## 5) Absence of Error - fallacy

It is possible that software which is 99% bug-free is still unusable. This can be the case if the system is tested thoroughly for the wrong requirement. Software testing is not mere finding defects, but also to check that software addresses the business needs. The absence of Error is a Fallacy i.e. Finding and fixing defects does not help if the system build is unusable and does not fulfill the user's needs & requirements.

To solve this problem, the next principle of testing states that Early Testing

## 6) Early Testing

Early Testing - Testing should start as early as possible in the Software Development Life Cycle. So that any defects in the requirements or design phase are captured in early stages. It is much cheaper to fix a Defect in the early stages of testing. But how early one should start testing? It is recommended that you start finding the bug the moment the requirements are defined. More on this principle in a later training tutorial.

## 7) Testing is context dependent

Testing is context dependent which basically means that the way you test an e-commerce site will be different from the way you test a commercial off the shelf application. All the developed software’s are not identical. You might use a different approach, methodologies, techniques, and types of testing depending upon the application type. For instance testing, any POS system at a retail store will be different than testing an ATM machine.

## Summary of the Seven Testing Principles

|  |  |
| --- | --- |
| Principle 1 | Testing shows presence of defects |
| Principle 2 | Exhaustive testing is impossible |
| Principle 3 | Early Testing |
| Principle 4 | Defect Clustering |
| Principle 5 | Pesticide Paradox |
| Principle 6 | Testing is context dependent |
| Principle 7 | Absence of errors - fallacy |

## What is V Model?

**V-Model** is an SDLC model that has a testing phase corresponding to every development stage in the waterfall model. It is pronounced as the "vee" model. The V-model is an extension of the waterfall model. V model Testing is done in parallel to development. It is also called a Validation and Verification Model.

## Key Software Engineering Terms:

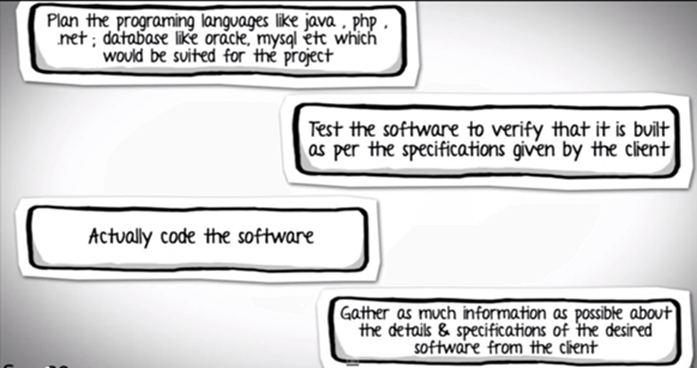
**SDLC:** SDLC is Software Development Life Cycle. It is the sequence of activities carried out by Developers to design and develop high-quality software.

**STLC:** STLC is Software Testing Life Cycle. It consists of a series of activities carried out by Testers methodologically to test your software product.

**Waterfall Model:** Waterfall model is a sequential model divided into different phases of software development activity. Each stage is designed for performing the specific activity. Testing phase in waterfall model starts only after implementation of the system is done.

## EXAMPLE To Understand the V Model

Suppose, you are assigned a task, to develop a custom software for a client. Now, irrespective of your technical background, try and make an educated guess about the sequence of steps you will follow, to achieve the task.

[](https://www.guru99.com/images/6-2015/052715_0904_GuidetoSDLC1.png)

The correct sequence would be.

|  |  |
| --- | --- |
| **Different phases of the Software Development Cycle** | **Activities performed in each stage** |
| **Requirement Gathering stage** | * Gather as much information as possible about the details & specifications of the desired software from the client. This is nothing but the Requirements gathering stage. |
| **Design Stage** | * Plan the programming language like [Java](https://www.guru99.com/java-tutorial.html), [PHP](https://www.guru99.com/php-tutorials.html), .net; database like Oracle, MySQL, etc. Which would be suited for the project, also some high-level functions & architecture. |
| **Build Stage** | * After the design stage, it is build stage, that is nothing but actually code the software |
| **Test Stage** | * Next, you test the software to verify that it is built as per the specifications are given by the client. |
| **Deployment stage** | * Deploy the application in the respective environment |
| **Maintenance stage** | * Once your system is ready to use, you may require to change the code later on as per customer request |

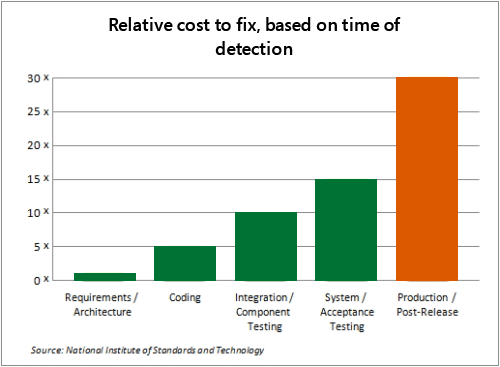
All these levels constitute the **waterfall method** of the software development lifecycle.

## Problem with the Waterfall Model

As you may observe, that **testing in the model starts only after implementation is done.**

But if you are working in the large project, where the systems are complex, it's easy to miss out the key details in the requirements phase itself. In such cases, an entirely wrong product will be delivered to the client and you might have to start afresh with the project OR if you manage to note the requirements correctly but make serious mistakes in design and architecture of your software you will have to redesign the entire software to correct the error.

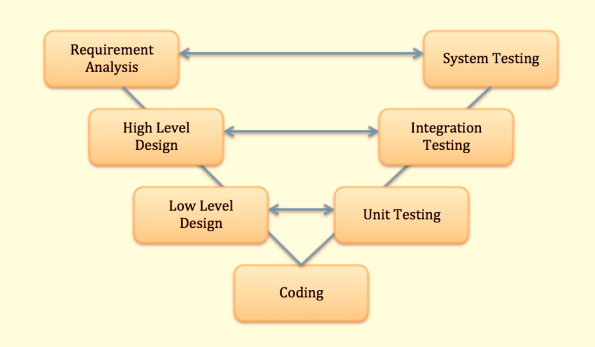
Assessments of thousands of projects have shown that **defects introduced during requirements & design make up close to half of the total number of defects.**

[](https://www.guru99.com/images/qwer.png)

Also, the **costs of fixing a defect increase across the development lifecycle. The earlier in life cycle a defect is detected, the cheaper it is to fix it.** As they say, "A stitch in time saves nine."

## Solution: The V Model

To address this concern, **the V model of testing** was developed where **for every phase, in the Development life cycle there is a corresponding Testing phase**

[](https://www.guru99.com/images/6-2015/052715_0904_GuidetoSDLC3.png)

* The left side of the model is Software Development Life Cycle - **SDLC**
* The right side of the model is Software Test Life Cycle - **STLC**
* The entire figure looks like a V, hence the name **V - model**

Apart from the V model, there are iterative development models, where development is carried in phases, with each phase adding a functionality to the software. Each phase comprises its independent set of development and testing activities.

Good examples of Development lifecycles following iterative method are Rapid Application Development, Agile Development

### Conclusion

There are numerous development life cycle models. **Development model selected for a project depends on the aims and goals of that project.**

* Testing is not a stand-alone activity, and it has to adapt the development model chosen for the project.
* In any model, testing should be performed at all levels i.e. right from requirements until maintenance.

## What is V Model?

**V-Model** is an SDLC model that has a testing phase corresponding to every development stage in the waterfall model. It is pronounced as the "vee" model. The V-model is an extension of the waterfall model. V model Testing is done in parallel to development. It is also called a Validation and Verification Model.

## Key Software Engineering Terms:

**SDLC:** SDLC is Software Development Life Cycle. It is the sequence of activities carried out by Developers to design and develop high-quality software.

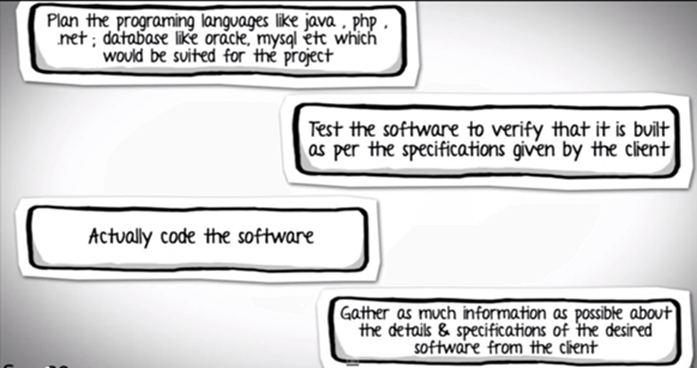
**STLC:** STLC is Software Testing Life Cycle. It consists of a series of activities carried out by Testers methodologically to test your software product.

**Waterfall Model:** Waterfall model is a sequential model divided into different phases of software development activity. Each stage is designed for performing the specific activity. Testing phase in waterfall model starts only after implementation of the system is done.

Click [here](https://www.guru99.com/faq.html#1) if the video is not accessible

## EXAMPLE To Understand the V Model

Suppose, you are assigned a task, to develop a custom software for a client. Now, irrespective of your technical background, try and make an educated guess about the sequence of steps you will follow, to achieve the task.

[](https://www.guru99.com/images/6-2015/052715_0904_GuidetoSDLC1.png)

The correct sequence would be.

|  |  |
| --- | --- |
| **Different phases of the Software Development Cycle** | **Activities performed in each stage** |
| **Requirement Gathering stage** | * Gather as much information as possible about the details & specifications of the desired software from the client. This is nothing but the Requirements gathering stage. |
| **Design Stage** | * Plan the programming language like [Java](https://www.guru99.com/java-tutorial.html), [PHP](https://www.guru99.com/php-tutorials.html), .net; database like Oracle, MySQL, etc. Which would be suited for the project, also some high-level functions & architecture. |
| **Build Stage** | * After the design stage, it is build stage, that is nothing but actually code the software |
| **Test Stage** | * Next, you test the software to verify that it is built as per the specifications are given by the client. |
| **Deployment stage** | * Deploy the application in the respective environment |
| **Maintenance stage** | * Once your system is ready to use, you may require to change the code later on as per customer request |

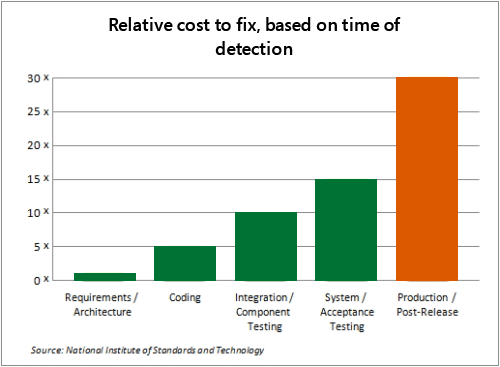
All these levels constitute the **waterfall method** of the software development lifecycle.

## Problem with the Waterfall Model

As you may observe, that **testing in the model starts only after implementation is done.**

But if you are working in the large project, where the systems are complex, it's easy to miss out the key details in the requirements phase itself. In such cases, an entirely wrong product will be delivered to the client and you might have to start afresh with the project OR if you manage to note the requirements correctly but make serious mistakes in design and architecture of your software you will have to redesign the entire software to correct the error.

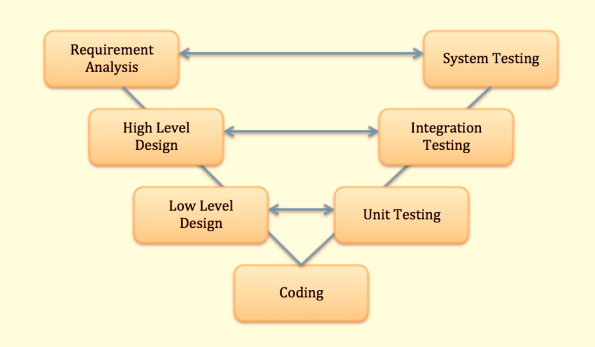
Assessments of thousands of projects have shown that **defects introduced during requirements & design make up close to half of the total number of defects.**

[](https://www.guru99.com/images/qwer.png)

Also, the **costs of fixing a defect increase across the development lifecycle. The earlier in life cycle a defect is detected, the cheaper it is to fix it.** As they say, "A stitch in time saves nine."

## Solution: The V Model

To address this concern, **the V model of testing** was developed where **for every phase, in the Development life cycle there is a corresponding Testing phase**

[](https://www.guru99.com/images/6-2015/052715_0904_GuidetoSDLC3.png)

* The left side of the model is Software Development Life Cycle - **SDLC**
* The right side of the model is Software Test Life Cycle - **STLC**
* The entire figure looks like a V, hence the name **V - model**

Apart from the V model, there are iterative development models, where development is carried in phases, with each phase adding a functionality to the software. Each phase comprises its independent set of development and testing activities.

Good examples of Development lifecycles following iterative method are Rapid Application Development, Agile Development

### Conclusion

There are numerous development life cycle models. **Development model selected for a project depends on the aims and goals of that project.**

* Testing is not a stand-alone activity, and it has to adapt the development model chosen for the project.
* In any model, testing should be performed at all levels i.e. right from requirements until maintenance.

## What is Software Testing Life Cycle (STLC)?

**SOFTWARE TESTING LIFE CYCLE(STLC)** is a sequence of specific activities conducted during the testing process to ensure software quality goals are met. STLC involves both verification and validation activities. Contrary to popular belief, Software Testing is not just a single/isolate activity, i.e. testing. It consists of a series of activities carried out methodologically to help certify your software product. STLC stands for Software Testing Life Cycle.

## Different Phases of the STLC Model

[](https://www.guru99.com/images/stories/software-test-life-cycle.jpg)STLC Diagram

Below are the phases of STLC:

* Requirement Analysis
* Test Planning
* Test case development
* Test Environment setup
* Test Execution
* Test Cycle closure

Each of these stages has a definite Entry and Exit criteria, Activities & Deliverables associated with it.

### What is Entry and Exit Criteria?

* **Entry Criteria:** Entry Criteria gives the prerequisite items that must be completed before testing can begin.
* **Exit Criteria:** Exit Criteria defines the items that must be completed before testing can be concluded

You have Entry and Exit Criteria for all levels in the Software Testing Life Cycle (STLC)

In an Ideal world, you will not enter the next stage until the exit criteria for the previous stage is met. But practically this is not always possible. So for this tutorial, we will focus on activities and deliverables for the different stages in STLC life cycle. Let's look into them in detail.

## Requirement Analysis

During this phase, test team studies the requirements from a testing point of view to identify the testable requirements.

The QA team may interact with various stakeholders (Client, Business Analyst, Technical Leads, System Architects etc) to understand the requirements in detail.

Requirements could be either Functional (defining what the software must do) or Non Functional (defining system performance /security availability )

Automation feasibility for the given testing project is also done in this stage.

**Activities**

* Identify types of tests to be performed.
* Gather details about testing priorities and focus.
* Prepare [Requirement Traceability Matrix (RTM)](https://www.guru99.com/traceability-matrix.html).
* Identify test environment details where testing is supposed to be carried out.
* Automation feasibility analysis (if required).

**Deliverables**

* RTM
* Automation feasibility report. (if applicable)

## Test Planning

Typically, in this stage, a Senior QA manager will determine effort and cost estimates for the project and would prepare and finalize the Test Plan. In this phase, Test Strategy is also determined.

**Activities**

* Preparation of test plan/strategy document for various types of testing
* Test tool selection
* Test effort estimation
* Resource planning and determining roles and responsibilities.
* Training requirement

**Deliverables**

* [Test plan](https://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html) /strategy document.
* [Effort estimation](https://www.guru99.com/an-expert-view-on-test-estimation.html) document.

## Test Case Development

This phase involves the creation, verification and rework of test cases & test scripts. [Test data](https://www.guru99.com/software-testing-test-data.html), is identified/created and is reviewed and then reworked as well.

**Activities**

* Create test cases, automation scripts (if applicable)
* Review and baseline test cases and scripts
* Create test data (If Test Environment is available)

**Deliverables**

* Test cases/scripts
* Test data

## Test Environment Setup

Test environment decides the software and hardware conditions under which a work product is tested. Test environment set-up is one of the critical aspects of testing process and ***can be done in parallel with Test Case Development Stage***. ***Test team may not be involved in this activity*** if the customer/development team provides the test environment in which case the test team is required to do a readiness check (smoke testing) of the given environment.

**Activities**

* Understand the required architecture, environment set-up and prepare hardware and software requirement list for the Test Environment.
* Setup test Environment and test data
* Perform smoke test on the build

**Deliverables**

* Environment ready with test data set up
* Smoke Test Results.

Test ExecutionDuring this phase, the testers will carry out the testing based on the test plans and the test cases prepared. Bugs will be reported back to the development team for correction and retesting will be performed.

**Activities**

* Execute tests as per plan
* Document test results, and log defects for failed cases
* Map defects to test cases in RTM
* Retest the[Defect](https://www.guru99.com/defect-management-process.html)fixes
* Track the defects to closure

**Deliverables**

* Completed RTM with the execution status
* Test cases updated with results
* Defect reports

## STLC Phases along with Entry and Exit Criteria

| **STLC Stage** | **Entry Criteria** | **Activity** | **Exit Criteria** | **Deliverables** |
| --- | --- | --- | --- | --- |
| Requirement Analysis | * Requirements Document available (both functional and non functional) * Acceptance criteria defined. * Application architectural document available. | * Analyse business functionality to know the business modules and module specific functionalities. * Identify all transactions in the modules. * Identify all the user profiles. * Gather user interface/ authentication, geographic spread requirements. * Identify types of tests to be performed. * Gather details about testing priorities and focus. * Prepare Requirement[Traceability Matrix](https://www.guru99.com/traceability-matrix.html)(RTM). * Identify test environment details where testing is supposed to be carried out. * Automation feasibility analysis (if required). | * Signed off RTM * Test automation feasibility report signed off by the client | * RTM * Automation feasibility report (if applicable) |
| Test Planning | * Requirements Documents * Requirement Traceability matrix. * Test automation feasibility document. | * Analyze various testing approaches available * Finalize on the best-suited approach * Preparation of test plan/strategy document for various types of testing * Test tool selection * Test effort estimation * Resource planning and determining roles and responsibilities. | * Approved test plan/strategy document. * Effort estimation document signed off. | * Test plan/strategy document. * Effort estimation document. |
| Test case development | * Requirements Documents * RTM and test plan * Automation analysis report | * Create test cases, test design, automation scripts (where applicable) * Review and baseline test cases and scripts * Create test data | * Reviewed and signed test Cases/scripts * Reviewed and signed test data | * Test cases/scripts * Test data |
| Test Environment setup | * System Design and architecture documents are available * Environment set-up plan is available | * Understand the required architecture, environment set-up * Prepare hardware and software development requirement list * Finalize connectivity requirements * Prepare environment setup checklist * Setup test Environment and test data * Perform smoke test on the build * Accept/reject the build depending on smoke test result | * Environment setup is working as per the plan and checklist * Test data setup is complete * Smoke test is successful | * Environment ready with test data set up * Smoke Test Results. |
| Test Execution | * Baselined RTM,[Test Plan](https://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html), Test case/scripts are available * Test environment is ready * Test data set up is done * Unit/Integration test report for the build to be tested is available | * Execute tests as per plan * Document test results, and log defects for failed cases * Update test plans/test cases, if necessary * Map defects to test cases in RTM * Retest the defect fixes * [Regression Testing](https://www.guru99.com/regression-testing.html) of application * Track the defects to closure | * All tests planned are executed * Defects logged and tracked to closure | * Completed RTM with execution status * Test cases updated with results * Defect reports |
| Test Cycle closure | * Testing has been completed * Test results are available * Defect logs are available | * Evaluate cycle completion criteria based on - Time,[Test coverage](https://www.guru99.com/test-coverage-in-software-testing.html), Cost, Software Quality, Critical Business Objectives * Prepare test metrics based on the above parameters. * Document the learning out of the project * Prepare Test closure report * Qualitative and quantitative reporting of quality of the work product to the customer. * Test result analysis to find out the defect d * istribution by type and severity | Test Closure report signed off by client | * Test Closure report * Test m |

## 

## Manual Testing Pros and Cons

**Pros of Manual Testing:**

* Get fast and accurate visual feedback
* It is less expensive as you don't need to spend your budget for the automation tools and process
* Human judgment and intuition always benefit the manual element
* While testing a small change, an automation test would require coding which could be time-consuming. While you could test manually on the fly.

**Cons of Manual Testing:**

* Less reliable testing method because it's conducted by a human. Therefore, it is always prone to mistakes & errors.
* The manual testing process can't be recorded, so it is not possible to reuse the manual test.
* In this testing method, certain tasks are difficult to perform manually which may require an additional time of the software testing phase.

## Automated Testing Pros and Cons

**Pros of automated testing:**

* Automated testing helps you to find more bugs compare to a human tester
* As most of the part of the testing process is automated, you can have a speedy and efficient process
* Automation process can be recorded. This allows you to reuse and execute the same kind of testing operations
* Automated testing is conducted using software tools, so it works without tiring and fatigue unlike humans in manual testing
* It can easily increase productivity because it provides fast & accurate testing result
* Automated testing support various applications
* Testing coverage can be increased because of automation testing tool never forget to check even the smallest unit

**Cons of Automated Testing:**

* Without human element, it's difficult to get insight into visual aspects of your UI like colors, font, sizes, contrast or button sizes.
* The tools to run automation testing can be expensive, which may increase the cost of the testing project.
* Automation testing tool is not yet foolproof. Every automation tool has their limitations which reduces the scope of automation.
* Debugging the test script is another major issue in the automated testing. Test maintenance is costly.

## KEY DIFFERENCE

* Manual Testing is done manually by QA analyst (Human) whereas Automation Testing is done with the use of script, code and automation tools (computer) by a tester.
* Manual Testing process is not accurate because of the possibilities of human errors whereas the Automation process is reliable because it is code and script based.
* Manual Testing is a time-consuming process whereas Automation Testing is very fast.
* Manual Testing is possible without programming knowledge whereas Automation Testing is not possible without programming knowledge.
* Manual Testing allows random Testing whereas Automation Testing doesn’t allow random Testing.

## What is Unit Testing?

**UNIT TESTING** is a type of software testing where individual units or components of a software are tested. The purpose is to validate that each unit of the software code performs as expected. Unit Testing is done during the development (coding phase) of an application by the developers. Unit Tests isolate a section of code and verify its correctness. A unit may be an individual function, method, procedure, module, or object.

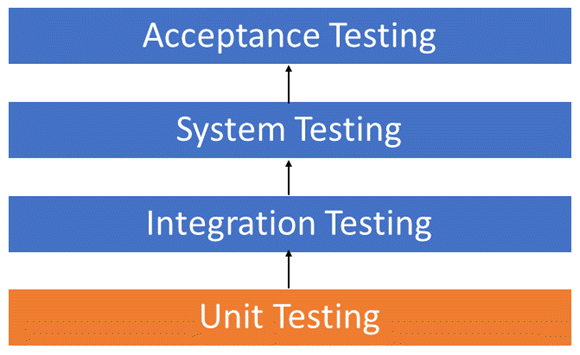
In SDLC, STLC, V Model, Unit testing is first level of testing done before integration testing. Unit testing is a WhiteBox testing technique that is usually performed by the developer. Though, in a practical world due to time crunch or reluctance of developers to tests, QA engineers also do unit testing.

In this tutorial, you will learn-

* [Why Unit Testing?](https://www.guru99.com/unit-testing-guide.html#2)
* [How to do Unit Testing](https://www.guru99.com/unit-testing-guide.html#3)
* [Unit Testing Techniques](https://www.guru99.com/unit-testing-guide.html#4)
* [Unit Testing Tools](https://www.guru99.com/unit-testing-guide.html#5)
* [Test Driven Development (TDD) & Unit Testing](https://www.guru99.com/unit-testing-guide.html#6)
* [Unit Testing Myth](https://www.guru99.com/unit-testing-guide.html#7)
* [Unit Testing Advantage](https://www.guru99.com/unit-testing-guide.html#8)
* [Unit Testing Disadvantages](https://www.guru99.com/unit-testing-guide.html#9)
* [Unit Testing Best Practices](https://www.guru99.com/unit-testing-guide.html#10)

## Why Unit Testing?

Sometimes software developers attempt to save time by doing minimal unit testing. This is a myth because skipping on unit testing leads to higher[Defect](https://www.guru99.com/defect-management-process.html)fixing costs during [System Testing](https://www.guru99.com/system-testing.html), [Integration Testing](https://www.guru99.com/integration-testing.html) and even Beta Testing after the application is completed. Proper unit testing done during the development stage saves both time and money in the end. Here, are key reasons to perform unit testing.

[](https://www.guru99.com/images/1/Unit-Testing.png)Unit Testing Levels

1. Unit tests help to fix bugs early in the development cycle and save costs.
2. It helps the developers to understand the code base and enables them to make changes quickly
3. Good unit tests serve as project documentation
4. Unit tests help with code re-use. Migrate both your code **and**your tests to your new project. Tweak the code until the tests run again.

## How to do Unit Testing

Unit Testing is of two types

* Manual
* Automated

Unit testing is commonly automated but may still be performed manually. Software Engineering does not favor one over the other but automation is preferred. A manual approach to unit testing may employ a step-by-step instructional document.

Under the automated approach-

* A developer writes a section of code in the application just to test the function. They would later comment out and finally remove the test code when the application is deployed.
* A developer could also isolate the function to test it more rigorously. This is a more thorough unit testing practice that involves copy and paste of code to its own testing environment than its natural environment. **Isolating the code helps in revealing unnecessary dependencies between the code being tested and other units or data spaces** in the product. These dependencies can then be eliminated.
* A coder generally uses a UnitTest Framework to develop automated test cases. Using an automation framework, the developer codes criteria into the test to verify the correctness of the code. During execution of the test cases, the framework logs failing test cases. Many frameworks will also automatically flag and report, in summary, these failed test cases. Depending on the severity of a failure, the framework may halt subsequent testing.
* The workflow of Unit Testing is 1) Create Test Cases 2) Review/Rework 3) Baseline 4) Execute Test Cases.

## Unit Testing Techniques

Code coverage techniques used in united testing are listed below:

* Statement Coverage
* Decision Coverage
* Branch Coverage
* Condition Coverage
* Finite State Machine Coverage

For more in refer <https://www.guru99.com/code-coverage.html>

## Unit Testing Example: Mock Objects

Unit testing relies on mock objects being created to test sections of code that are not yet part of a complete application. Mock objects fill in for the missing parts of the program.

For example, you might have a function that needs variables or objects that are not created yet. In unit testing, those will be accounted for in the form of mock objects created solely for the purpose of the unit testing done on that section of code.

## Unit Testing Tools

There are several automated tools available to assist with unit testing. We will provide a few examples below:

1. [Junit](https://www.guru99.com/junit-tutorial.html): Junit is a free to use testing tool used for Java programming language.  It provides assertions to identify test method. This tool test data first and then inserted in the piece of code.
2. [NUnit](https://nunit.org/):  NUnit is widely used unit-testing framework use for all .net languages.  It is an open source tool which allows writing scripts manually. It supports data-driven tests which can run in parallel.
3. [JMockit](http://jmockit.github.io/index.html):  JMockit is open source Unit testing tool.  It is a code coverage tool with line and path metrics. It allows mocking API with recording and verification syntax. This tool offers Line coverage, Path Coverage, and Data Coverage.
4. [EMMA](http://emma.sourceforge.net/):  EMMA is an open-source toolkit for analyzing and reporting code written in Java language. Emma support coverage types like method, line, basic block. It is Java-based so it is without external library dependencies and can access the source code.
5. [PHPUnit](https://phpunit.de/): PHPUnit is a unit testing tool for PHP programmer. It takes small portions of code which is called units and test each of them separately.  The tool also allows developers to use pre-define assertion methods to assert that a system behave in a certain manner.

Those are just a few of the available unit testing tools. There are lots more, especially for C languages and Java, but you are sure to find a unit testing tool for your programming needs regardless of the language you use.

## Test Driven Development (TDD) & Unit Testing

Unit testing in TDD involves an extensive use of testing frameworks. A unit test framework is used in order to create automated unit tests. Unit testing frameworks are not unique to TDD, but they are essential to it. Below we look at some of what TDD brings to the world of unit testing:

* Tests are written before the code
* Rely heavily on testing frameworks
* All classes in the applications are tested
* Quick and easy integration is made possible

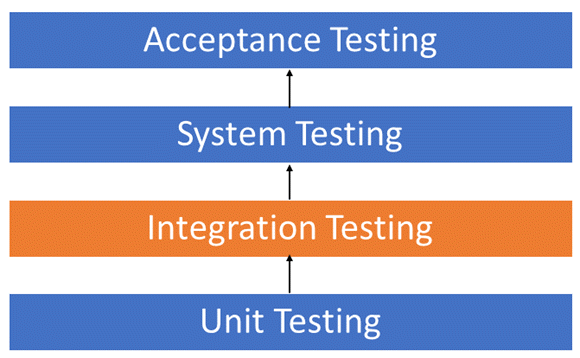
## What is Integration Testing?

**INTEGRATION TESTING** is defined as a type of testing where software modules are integrated logically and tested as a group. A typical software project consists of multiple software modules, coded by different programmers. The purpose of this level of testing is to expose defects in the interaction between these software modules when they are integrated

 Integration Testing focuses on checking data communication amongst these modules. Hence it is also termed as **'I & T'** (Integration and Testing), **'String Testing'** and sometimes **'Thread Testing'**.

* [What is Integration Testing?](https://www.guru99.com/integration-testing.html#1)
* [Why do Integration Testing?](https://www.guru99.com/integration-testing.html#2)
* [Example of Integration Test Case](https://www.guru99.com/integration-testing.html#3)
* [Approaches, Strategies, Methodologies of Integration Testing](https://www.guru99.com/integration-testing.html#4)
* [Big Bang Approach:](https://www.guru99.com/integration-testing.html#5)
* [Incremental Approach](https://www.guru99.com/integration-testing.html#6)
* [What is Stub and Driver?](https://www.guru99.com/integration-testing.html#7)
* [Bottom-up Integration](https://www.guru99.com/integration-testing.html#8)
* [Top-down Integration:](https://www.guru99.com/integration-testing.html#9)
* [Hybrid/ Sandwich Integration](https://www.guru99.com/integration-testing.html#10)
* [How to do Integration Testing?](https://www.guru99.com/integration-testing.html#11)
* [Brief Description of Integration Test Plans:](https://www.guru99.com/integration-testing.html#12)
* [Entry and Exit Criteria of Integration Testing](https://www.guru99.com/integration-testing.html#13)
* [Best Practices/ Guidelines for Integration Testing](https://www.guru99.com/integration-testing.html#14)

## Why do Integration Testing?

[](https://www.guru99.com/images/1/Integration-Testing.png)

Although each software module is unit tested, defects still exist for various reasons like

* A Module, in general, is designed by an individual software developer whose understanding and programming logic may differ from other programmers. Integration Testing becomes necessary to verify the software modules work in unity
* At the time of module development, there are wide chances of change in requirements by the clients. These new requirements may not be unit tested and hence system integration Testing becomes necessary.
* Interfaces of the software modules with the database could be erroneous
* External Hardware interfaces, if any, could be erroneous
* Inadequate exception handling could cause issues.

Click [here](https://www.guru99.com/faq.html#1) if the video is not accessible

## Example of Integration Test Case

Integration[Test Case](https://www.guru99.com/test-case.html)differs from other test cases in the sense it**focuses mainly on the interfaces & flow of data/information between the modules**. Here priority is to be given for the **integrating links** rather than the unit functions which are already tested.

Sample Integration Test Cases for the following scenario: Application has 3 modules say 'Login Page', 'Mailbox' and 'Delete emails' and each of them is integrated logically.

Here do not concentrate much on the Login Page testing as it's already been done in [Unit Testing](https://www.guru99.com/unit-testing-guide.html). But check how it's linked to the Mail Box Page.

Similarly Mail Box: Check its integration to the Delete Mails Module.

| **Test Case ID** | **Test Case Objective** | **Test Case Description** | **Expected Result** |
| --- | --- | --- | --- |
| **1** | Check the interface link between the Login and Mailbox module | Enter login credentials and click on the Login button | To be directed to the Mail Box |
| **2** | Check the interface link between the Mailbox and Delete Mails Module | From Mailbox select the email and click a delete button | Selected email should appear in the Deleted/Trash folder |

## Approaches, Strategies, Methodologies of Integration Testing

Software Engineering defines variety of strategies to execute Integration testing, viz.

* Big Bang Approach :
* Incremental Approach: which is further divided into the following
  + Top Down Approach
  + Bottom Up Approach
  + Sandwich Approach - Combination of Top Down and Bottom Up

Below are the different strategies, the way they are executed and their limitations as well advantages.

## Big Bang Approach:

Here all component are integrated together at **once** and then tested.

**Advantages:**

* Convenient for small systems.

**Disadvantages:**

* Fault Localization is difficult.
* Given the sheer number of interfaces that need to be tested in this approach, some interfaces link to be tested could be missed easily.
* Since the Integration testing can commence only after "all" the modules are designed, the testing team will have less time for execution in the testing phase.
* Since all modules are tested at once, high-risk critical modules are not isolated and tested on priority. Peripheral modules which deal with user interfaces are also not isolated and tested on priority.

## Incremental Approach

In this approach, testing is done by joining two or more modules that are **logically related**. Then the other related modules are added and tested for the proper functioning. The process continues until all of the modules are joined and tested successfully.

Incremental Approach, in turn, is carried out by two different Methods:

* Bottom Up
* Top Down

## What is Stub and Driver?

Incremental Approach is carried out by using dummy programs called **Stubs and Drivers**. Stubs and Drivers do not implement the entire programming logic of the software module but just simulate data communication with the calling module.

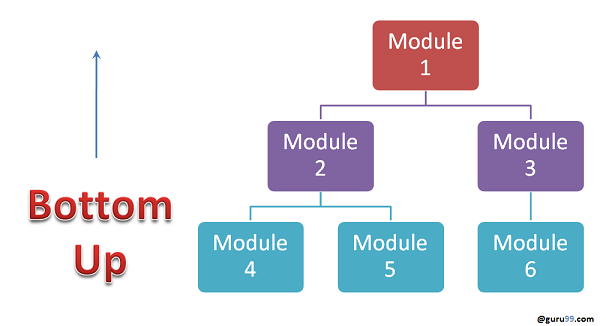
**Stub**: Is called by the Module under Test.

**Driver**: Calls the Module to be tested.

## Bottom-up Integration

In the bottom-up strategy, each module at lower levels is tested with higher modules until all modules are tested. It takes help of Drivers for testing

**Diagrammatic Representation**:

[](https://www.guru99.com/images/bottom-up-integration-testing.png)

**Advantages:**

* Fault localization is easier.
* No time  is wasted waiting for all modules to be developed unlike Big-bang approach

**Disadvantages:**

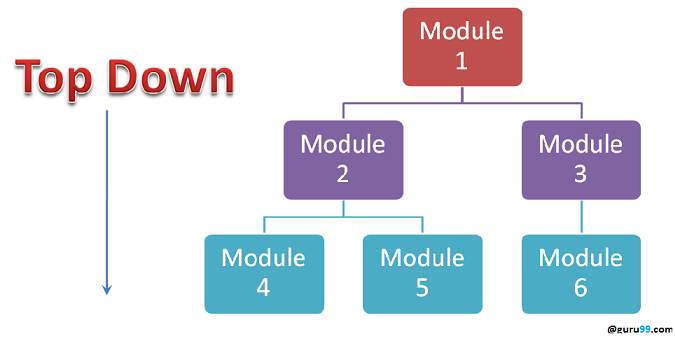
* Critical modules (at the top level of software architecture) which control the flow of application are tested last and may be prone to defects.
* An early prototype is not possible

## Top-down Integration:

In Top to down approach, testing takes place from top to down following the control flow of the software system.

Takes help of stubs for testing.

**Diagrammatic Representation:**

[](https://www.guru99.com/images/top-down-integration-testing.png)

**Advantages:**

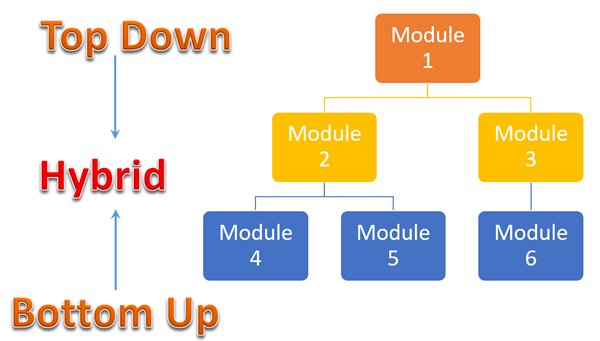
* Fault Localization is easier.
* Possibility to obtain an early prototype.
* Critical Modules are tested on priority; major design flaws could be found and fixed first.

**Disadvantages:**

* Needs many Stubs.
* Modules at a lower level are tested inadequately.

## Hybrid/ Sandwich Integration

In the sandwich/hybrid strategy is a combination of Top Down and Bottom up approaches. Here, top modules are tested with lower modules at the same time lower modules are integrated with top modules and tested. This strategy makes use of stubs as well as drivers.

[](https://www.guru99.com/images/1/Hybrid-Integration.png)

## How to do Integration Testing?

The Integration test procedure irrespective of the Software testing strategies (discussed above):

1. Prepare the Integration Tests Plan
2. Design the Test Scenarios, Cases, and Scripts.
3. Executing the test Cases followed by reporting the defects.
4. Tracking & re-testing the defects.
5. Steps 3 and 4 are repeated until the completion of Integration is successful.

## Brief Description of Integration Test Plans:

It includes the following attributes:

* Methods/Approaches to testing (as discussed above).
* Scopes and Out of Scopes Items of Integration Testing.
* Roles and Responsibilities.
* Pre-requisites for Integration testing.
* Testing environment.
* Risk and Mitigation Plans.

## Entry and Exit Criteria of Integration Testing

Entry and Exit Criteria to Integration testing phase in any software development model

**Entry Criteria:**

* Unit Tested Components/Modules
* All High prioritized bugs fixed and closed
* All Modules to be code completed and integrated successfully.
* Integration tests Plan, test case, scenarios to be signed off and documented.
* Required[Test Environment](https://www.guru99.com/test-environment-software-testing.html)to be set up for Integration testing

**Exit Criteria:**

* Successful Testing of Integrated Application.
* Executed Test Cases are documented
* All High prioritized bugs fixed and closed
* Technical documents to be submitted followed by release Notes.

## Best Practices/ Guidelines for Integration Testing

* First, determine the Integration[Test Strategy](https://www.guru99.com/how-to-create-test-strategy-document.html)that could be adopted and later prepare the test cases and test data accordingly.
* Study the Architecture design of the Application and identify the Critical Modules. These need to be tested on priority.
* Obtain the interface designs from the Architectural team and create test cases to verify all of the interfaces in detail. Interface to database/external hardware/software application must be tested in detail.
* After the test cases, it's the test data which plays the critical role.
* Always have the mock data prepared, prior to executing. Do not select test data while executing the test cases.

## What is System Testing?

**SYSTEM TESTING** is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system. Ultimately, the software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

In this tutorial, we will learn

* [System Testing is Blackbox](https://www.guru99.com/system-testing.html#1)
* [What do you verify in System Testing?](https://www.guru99.com/system-testing.html#2)
* [Software Testing Hierarchy](https://www.guru99.com/system-testing.html#3)
* [Different Types of System Testing](https://www.guru99.com/system-testing.html#4)
* [What Types of System Testing Should Testers Use?](https://www.guru99.com/system-testing.html#5)

## System Testing is Blackbox

Two Category of Software Testing

* Black Box Testing
* White Box Testing

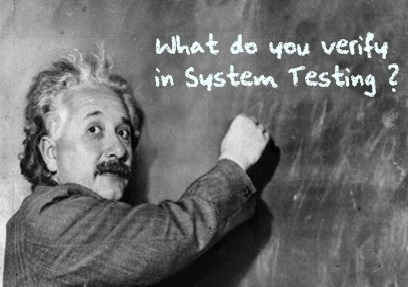
System test falls under the **black box testing** category of software testing.

**White box testing** is the testing of the internal workings or code of a software application. In contrast, black box or System Testing is the opposite. System test involves the external workings of the software from the user's perspective.

Click [here](https://www.guru99.com/faq.html#1) if the video is not accessible

## What do you verify in System Testing?

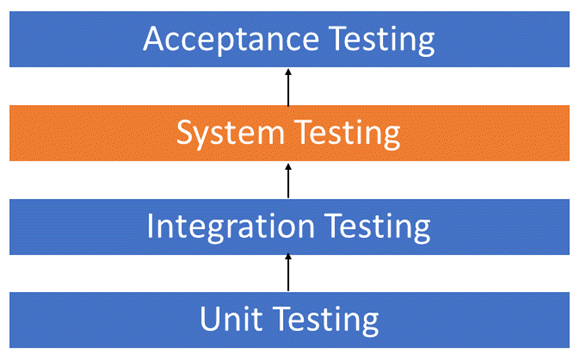
System Testing involves testing the software code for following

[](https://www.guru99.com/images/what%20do%20check%20in%20system%20testing.png)

* Testing the fully integrated applications including external peripherals in order to check how components interact with one another and with the system as a whole. This is also called End to End testing scenario.
* Verify thorough testing of every input in the application to check for desired outputs.
* Testing of the user's experience with the application.

That is a very basic description of what is involved in system testing. You need to build detailed test cases and test suites that test each aspect of the application as seen from the outside without looking at the actual source code.

## Software Testing Hierarchy

[](https://www.guru99.com/images/1/System-Testing.png)

As with almost any software engineering process, software testing has a prescribed order in which things should be done. The following is a list of software testing categories arranged in chronological order. These are the steps taken to fully test new software in preparation for marketing it:

* Unit testing performed on each module or block of code during development. [Unit Testing](https://www.guru99.com/unit-testing-guide.html) is normally done by the programmer who writes the code.

* Integration 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.

* System testing done by a professional testing agent on the completed software product before it is introduced to the market.

* Acceptance testing - beta testing of the product done by the actual end users.

## Different Types of System Testing

There are more than 50 types of System Testing. For an exhaustive list of software testing types click [here](https://www.guru99.com/types-of-software-testing.html). Below we have listed types of system testing a large software development company would typically use

1. [Usability Testing](https://www.guru99.com/usability-testing-tutorial.html)- mainly focuses on the user's ease to use the application, flexibility in handling controls and ability of the system to meet its objectives

1. [Load Testing](https://www.guru99.com/load-testing-tutorial.html)- is necessary to know that a software solution will perform under real-life loads.

1. [Regression Testing](https://www.guru99.com/regression-testing.html)- 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.

1. Recovery testing - is done to demonstrate a software solution is reliable, trustworthy and can successfully recoup from possible crashes.

1. Migration testing- is done to ensure that the software can be moved from older system infrastructures to current system infrastructures without any issues.

1. Functional Testing - Also known as functional completeness testing,[Functional Testing](https://www.guru99.com/functional-testing.html) 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.

1. 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.

## What Types of System Testing Should Testers Use?

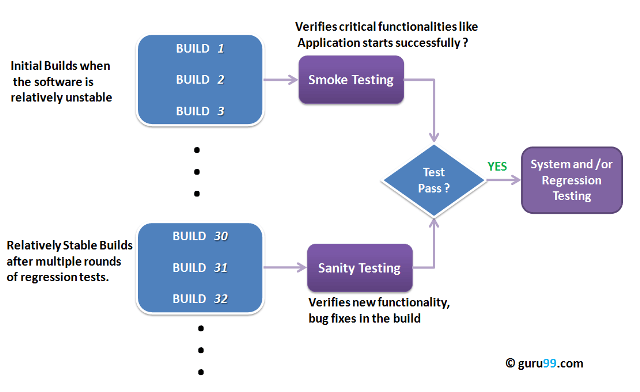
There are over 50 different types of system testing. The specific types used by a tester depend on several variables. Those variables include:

* Who the tester works for - This is a major factor in determining the types of system testing a tester will use. Methods used by large companies are different than that used by medium and small companies.
* Time available for testing - Ultimately, all 50 testing types could be used. Time is often what limits us to using only the types that are most relevant for the software project.
* Resources available to the tester - Of course some testers will not have the necessary resources to conduct a testing type. For example, if you are a tester working for a large software development firm, you are likely to have expensive automated testing software not available to others.
* Software Tester's Education- There is a certain learning curve for each type of software testing available. To use some of the software involved, a tester has to learn how to use it.
* Testing Budget - Money becomes a factor not just for smaller companies and individual software developers but large companies as well.

# Sanity Testing Vs Smoke Testing: Introduction & Differences

Smoke and Sanity testing are the most misunderstood topics in Software Testing. There is an enormous amount of literature on the subject, but most of them are confusing. The following article makes an attempt to address the confusion.

The key differences between Smoke and Sanity Testing can be learned with the help of the following diagram -

[](https://www.guru99.com/images/stories/Sanity_Smoke_Testing.png)

To appreciate the above diagram lets first understand -

## What is a Software Build?

If you are developing a simple computer program which consists of only one source code file, you merely need to compile and link this one file, to produce an executable file. This process is very simple.  
Usually, this is not the case. A typical Software Project consists of hundreds or even thousands of source code files. Creating an executable program from these source files is a complicated and time-consuming task.  
You need to use "build" software to create an executable program and the process is called " Software Build"

## What is Smoke Testing?

[Smoke Testing](https://www.guru99.com/smoke-testing.html) is a kind of Software Testing performed after software build to ascertain that the critical functionalities of the program are working fine. It is executed "before" any detailed functional or regression tests are executed on the software build. The purpose is to reject a badly broken application so that the QA team does not waste time installing and testing the software application.

In Smoke Testing, the test cases chose to cover the most important functionality or component of the system. The objective is not to perform exhaustive testing, but to verify that the critical functionalities of the system are working fine.  
For Example, a typical smoke test would be - Verify that the application launches successfully, Check that the GUI is responsive ... etc.

## What is Sanity Testing?

Sanity testing is a kind of Software Testing performed after receiving a software build, with minor changes in code, or functionality, to ascertain that the bugs have been fixed and no further issues are introduced due to these changes. The goal is to determine that the proposed functionality works roughly as expected. If sanity test fails, the build is rejected to save the time and costs involved in a more rigorous testing.

The objective is "not" to verify thoroughly the new functionality but to determine that the developer has applied some rationality (sanity) while producing the software. For instance, if your scientific calculator gives the result of 2 + 2 =5! Then, there is no point testing the advanced functionalities like sin 30 + cos 50.

## Smoke Testing Vs Sanity Testing - Key Differences

|  |  |
| --- | --- |
| **Smoke Testing** | **Sanity Testing** |
| Smoke Testing is performed to ascertain that the critical functionalities of the program is working fine | Sanity Testing is done to check the new functionality/bugs have been fixed |
| The objective of this testing is to verify the "stability" of the system in order to proceed with more rigorous testing | The objective of the testing is to verify the "rationality" of the system in order to proceed with more rigorous testing |
| This testing is performed by the developers or testers | Sanity testing is usually performed by testers |
| Smoke testing is usually documented or scripted | Sanity testing is usually not documented and is unscripted |
| Smoke testing is a subset of Acceptance testing | Sanity testing is a subset of [Regression Testing](https://www.guru99.com/regression-testing.html) |
| Smoke testing exercises the entire system from end to end | Sanity testing exercises only the particular component of the entire system |
| Smoke testing is like General Health Check Up | Sanity Testing is like specialized health check up |

## Points to note.

* Both sanity tests and smoke tests are ways to avoid wasting time and effort by quickly determining whether an application is too flawed to merit any rigorous testing.
* Sanity Testing is also called tester acceptance testing.
* Smoke testing performed on a particular build is also known as a build verification test.
* One of the best industry practice is to conduct a Daily build and smoke test in software projects.
* Both smoke and sanity tests can be executed manually or using an automation tool.  When automated tools are used, the tests are often initiated by the same process that generates the build itself.
* As per the needs of testing, you may have to execute both Sanity and Smoke Tests in the software build. In such cases, you will first execute Smoke tests and then go ahead with Sanity Testing. In industry, test cases for Sanity Testing are commonly combined with that for smoke tests, to speed up test execution. Hence, it's a common that the terms are often confused and used interchangeably

Click [here](https://www.guru99.com/faq.html#1) if the video is not accessible

Read more about [Smoke Testing](https://www.guru99.com/smoke-testing.html)in software engineering

## KEY DIFFERENCE

* Smoke Testing has a goal to verify “stability” whereas Sanity Testing has a goal to verify “rationality”.
* Smoke Testing is done by both developers or testers whereas Sanity Testing is done by testers.
* Smoke Testing verifies the critical functionalities of the system whereas Sanity Testing verifies the new functionality like bug fixes.
* Smoke testing is a subset of acceptance testing whereas Sanity testing is a subset of Regression Testing.
* Smoke testing is documented or scripted whereas Sanity testing isn’t.
* Smoke testing verifies the entire system from end to end whereas Sanity Testing verifies only a particular component.

# What is Regression Testing? Definition, Test Cases (Example)

## What is Regression Testing?

**REGRESSION TESTING** is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features.

Regression Testing is nothing but a full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine.

This testing is done to make sure that new code changes should not have side effects on the existing functionalities. It ensures that the old code still works once the latest code changes are done.

In this tutorial, we will learn

* [Need of Regression Testing](https://www.guru99.com/regression-testing.html#1)
* [How to do Regression Testing](https://www.guru99.com/regression-testing.html#2)
* [Selecting test cases for regression testing](https://www.guru99.com/regression-testing.html#3)
* [Regression Testing Tools](https://www.guru99.com/regression-testing.html#4)
* [Regression Testing and Configuration Management](https://www.guru99.com/regression-testing.html#5)
* [Difference between Re-Testing and Regression Testing](https://www.guru99.com/regression-testing.html#6)
* [Challenges in Regression Testing](https://www.guru99.com/regression-testing.html#7)
* [Practical Application of Regression Testing Example with a Video](https://www.guru99.com/regression-testing.html#8)

## Need of Regression Testing

Regression Testing is required when there is a

* Change in requirements and code is modified according to the requirement
* New feature is added to the software
* Defect fixing
* Performance issue fix

## How to do Regression Testing

Software maintenance is an activity which includes enhancements, error corrections, optimization and deletion of existing features. These modifications may cause the system to work incorrectly. Therefore, Regression Testing becomes necessary. Regression Testing can be carried out using the following techniques:

[](https://www.guru99.com/images/regressiontestingtypes.png)

### Retest All

* This is one of the methods for Regression Testing in which all the tests in the existing test bucket or suite should be re-executed. This is very expensive as it requires huge time and resources.

### Regression Test Selection

* Instead of re-executing the entire test suite, it is better to select part of the test suite to be run
* Test cases selected can be categorized as 1) Reusable Test Cases 2) Obsolete Test Cases.
* Re-usable Test cases can be used in succeeding regression cycles.
* Obsolete Test Cases can't be used in succeeding cycles.

### Prioritization of Test Cases

* Prioritize the test cases depending on business impact, critical & frequently used functionalities. Selection of test cases based on priority will greatly reduce the regression test suite.

## Selecting test cases for regression testing

It was found from industry data that a good number of the defects reported by customers were due to last minute bug fixes creating side effects and hence selecting the[Test Case](https://www.guru99.com/test-case.html)for regression testing is an art and not that easy.  Effective Regression Tests can be done by selecting the following test cases -

* Test cases which have frequent defects
* Functionalities which are more visible to the users
* Test cases which verify core features of the product
* Test cases of Functionalities which has undergone more and recent changes
* All Integration Test Cases
* All Complex Test Cases
* Boundary value test cases
* A sample of Successful test cases
* A sample of Failure test cases

## Regression Testing Tools

If your software undergoes frequent changes, regression testing costs will escalate.

In such cases, Manual execution of test cases increases test execution time as well as costs.

Automation of regression test cases is the smart choice in such cases.

The extent of automation depends on the number of test cases that remain re-usable for successive regression cycles.

Following are the most important tools used for both functional and regression testing in software engineering.

[**Ranorex Studio**](https://bit.ly/2XolITW): all-in-one regression test automation for desktop, web, and mobile apps with built-in Selenium WebDriver. Includes a full IDE plus tools for codeless automation.

[**Selenium**](https://www.guru99.com/selenium-tutorial.html): This is an open source tool used for automating web applications. Selenium can be used for browser-based regression testing.

[**Quick Test Professional (QTP)**](https://www.guru99.com/quick-test-professional-qtp-tutorial.html): HP Quick Test Professional is automated software designed to automate functional and regression test cases. It uses[VBScript](https://www.guru99.com/vbscript-tutorials-for-beginners.html)language for automation. It is a Data-driven, Keyword based tool.

**Rational Functional Tester (RFT)**: IBM's rational functional tester is a[Java](https://www.guru99.com/java-tutorial.html)tool used to automate the test cases of software applications. This is primarily used for automating regression test cases and it also integrates with Rational Test Manager.

## Regression Testing and Configuration Management

Configuration Management during Regression Testing becomes imperative in Agile Environments where a code is being continuously modified. To ensure effective regression tests, observe the following :

* Code being regression tested should be under a configuration management tool
* No changes must be allowed to code, during the regression test phase.  Regression test code must be kept immune to developer changes.
* The database used for regression testing must be isolated. No database changes must be allowed

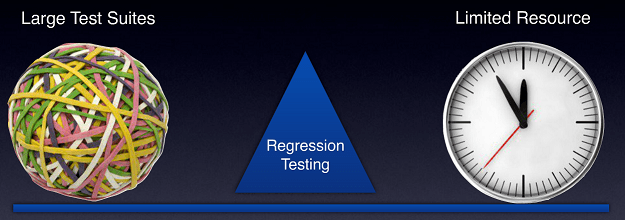
## Difference between Re-Testing and Regression Testing:

Retesting means testing the functionality or bug again to ensure the code is fixed. If it is not fixed,[Defect](https://www.guru99.com/defect-management-process.html)needs to be re-opened. If fixed, Defect is closed.

Regression testing means testing your software application when it undergoes a code change to ensure that the new code has not affected other parts of the software.

Also, Check out the complete list of differences over [here](https://www.guru99.com/re-testing-vs-regression-testing.html).

## Challenges in Regression Testing:

[](https://www.guru99.com/images/regressiontestingchallenges.png)

 Following are the major testing problems for doing regression testing:

* With successive regression runs, test suites become fairly large.  Due to time and budget constraints, the entire regression test suite cannot be executed
* Minimizing the test suite while achieving maximum[Test coverage](https://www.guru99.com/test-coverage-in-software-testing.html)remains a challenge
* Determination of frequency of Regression Tests, i.e., after every modification or every build update or after a bunch of bug fixes, is a challenge.

## Practical Application of Regression Testing Example with a Video

Click [here](https://www.guru99.com/faq.html#1) if the video is not accessible

**Conclusion:**

An effective regression strategy, save organizations both time and money. As per one of the case study in banking domain, regression saves up to 60% time in bug fixes(which would have been caught by regression tests) and 40%  in money

# What is Non Functional Testing? Types with Example

## What is Non-Functional Testing?

**NON-FUNCTIONAL TESTING** is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.

An excellent example of non-functional test would be to check how many people can simultaneously login into a software.

Non-functional testing is equally important as functional testing and affects client satisfaction.

In this tutorial, we will learn

* [Objectives of Non-functional testing](https://www.guru99.com/non-functional-testing.html#1)
* [Characteristics of Non-functional testing](https://www.guru99.com/non-functional-testing.html#2)
* [Non-functional testing Parameters](https://www.guru99.com/non-functional-testing.html#3)
* [Type of Software Testing](https://www.guru99.com/non-functional-testing.html#4)
* [Non-functional Testing Types](https://www.guru99.com/non-functional-testing.html#5)
* [Example Test Cases Non-Functional Testing](https://www.guru99.com/non-functional-testing.html#6)

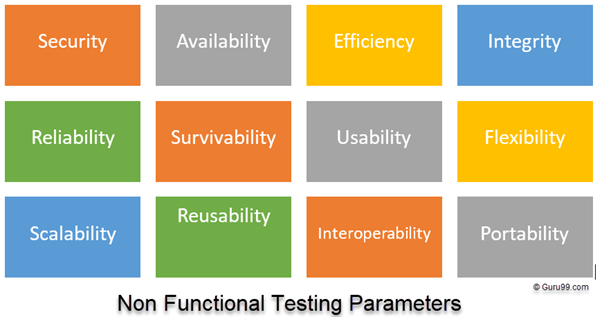
## Objectives of Non-functional testing

* Non-functional testing should increase usability, efficiency, maintainability, and portability of the product.
* Helps to reduce production risk and cost associated with non-functional aspects of the product.
* Optimize the way product is installed, setup, executes, managed and monitored.
* Collect and produce measurements, and metrics for internal research and development.
* Improve and enhance knowledge of the product behavior and technologies in use.

## Characteristics of Non-functional testing

* Non-functional testing should be measurable, so there is no place for subjective characterization like good, better, best, etc.
* Exact numbers are unlikely to be known at the start of the requirement process
* Important to prioritize the requirements
* Ensure that quality attributes are identified correctly in Software Engineering.

## Non-functional testing Parameters

[](https://www.guru99.com/images/1/022218_1114_WhatisNonFu2.png)

### 1) Security:

The parameter defines how a system is safeguarded against deliberate and sudden attacks from internal and external sources. This is tested via [Security Testing](https://www.guru99.com/what-is-security-testing.html).

### 2) Reliability:

The extent to which any software system continuously performs the specified functions without failure. This is tested by [Reliability Testing](https://www.guru99.com/reliability-testing.html)

### 3) Survivability:

The parameter checks that the software system continues to function and recovers itself in case of system failure. This is checked by [Recovery Testing](https://www.guru99.com/recovery-testing.html)

### 4) Availability:

The parameter determines the degree to which user can depend on the system during its operation. This is checked by [Stability Testing.](https://www.guru99.com/stability-testing.html)

### 5) Usability:

The ease with which the user can learn, operate, prepare inputs and outputs through interaction with a system. This is checked by [Usability Testing](https://www.guru99.com/usability-testing-tutorial.html)

### 6) Scalability:

The term refers to the degree in which any software application can expand its processing capacity to meet an increase in demand. This is tested by [Scalability Testing](https://www.guru99.com/scalability-testing.html)

### 7) Interoperability:

This non-functional parameter checks a software system interfaces with other software systems. This is checked by [Interoperability Testing](https://www.guru99.com/interoperability-testing.html)

### 8) Efficiency:

The extent to which any software system can handles capacity, quantity and response time.

### 9) Flexibility:

The term refers to the ease with which the application can work in different hardware and software configurations. Like minimum RAM, CPU requirements.

### 10) Portability:

The flexibility of software to transfer from its current hardware or software environment.

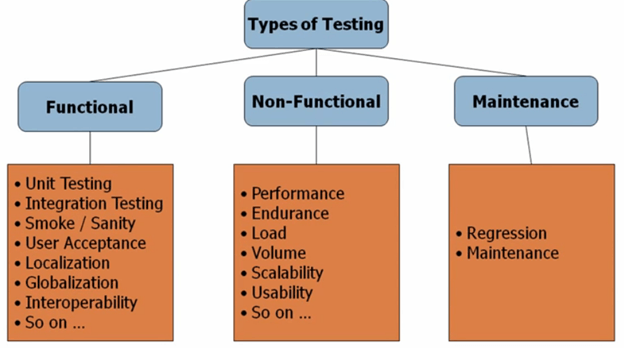
### 11) Reusability:

It refers to a portion of the software system that can be converted for use in another application.

## Type of Software Testing

In general, there are three testing types

* **Functional**
* **Non - Functional**
* **Maintenance**

[](https://www.guru99.com/images/1-2015/012715_0453_NonFunction4.png)

Under these types of testing, you have multiple TESTING Level's, but usually, people call them as Testing Types. You may find some difference in the above classification in different books and reference materials.

The above list is not the complete as there are **more than**[**100 Types of Testing**](https://www.guru99.com/types-of-software-testing.html)and counting. No need to worry, you will pick them up as you age in the testing industry. Also, note that not all testing types apply to all projects but depend on the nature & scope of the project. More on this in a later tutorial.

## Non-functional Testing Types

* Performance Testing
* Load Testing
* Failover Testing
* Compatibility Testing
* Usability Testing
* Stress Testing
* Maintainability Testing
* Scalability Testing
* Volume Testing
* Security Testing
* Disaster Recovery Testing
* Compliance Testing
* Portability Testing
* Efficiency Testing
* Reliability Testing
* Baseline Testing
* Endurance Testing
* Documentation Testing
* Recovery Testing
* Internationalization Testing
* Localization Testing

## Example Test Cases Non-Functional Testing

Following are examples of Non-Functional Testing

|  |  |  |
| --- | --- | --- |
| **Test Case #** | **Test Case** | **Domain** |
| 1 | Application load time should not be more than 5 secs up to 1000 users accessing it simultaneously | Performance Testing |
| 2 | Software should be installable on all versions of Windows and Mac | Compatibility Testing |
| 3 | All web images should have alt tags | Accessibility testing. |

## Examples of Test Documentation

Here, are important Types of Test Documentation:

|  |  |
| --- | --- |
| **Types of Testing** | **Description** |
| Test policy | It is a high-level document which describes principles, methods and all the important testing goals of the organization. |
| Test strategy | A high-level document which identifies the Test Levels (types) to be executed for the project. |
| Test plan | A test plan is a complete planning document which contains the scope, approach, resources, schedule, etc. of testing activities. |
| Requirements Traceability Matrix | This is a document which connects the requirements to the test cases. |
| Test Scenario | Test scenario is an item or event of a software system which could be verified by one or more Test cases. |
| Test case | It is a group of input values, execution preconditions, expected execution postconditions and results. It is developed for a Test Scenario. |
| Test Data | Test Data is a data which exists before a test is executed. It used to execute the test case. |
| Defect Report | Defect report is a documented report of any flaw in a Software System which fails to perform its expected function. |
| Test summary report | Test summary report is a high-level document which summarizes testing activities conducted as well as the test result. |

## Best practice to Achieve Test Documentation

* QA team needs to be involved in the initial phase of the project so that Test Documentation is created in parallel
* Don’t just create and leave the document, but update whenever required
* Use version control to manage and track your documents
* Try to document what is needed for you to understand your work and what you will need to produce to your stakeholders
* You should use a standard template for documentation like excel sheet or doc file
* Store all your project related documents at a single location. It should be accessible to every team member for reference as well as to update when needed
* Not providing enough detail is also a common mistake while creating a test document

## Advantages of Test Documentation

* The main reason behind creating test documentation is to either reduce or remove any uncertainties about the testing activities. Helps you to remove ambiguity which often arises when it comes to the allocation of tasks
* Documentation not only offers a systematic approach to software testing, but it also acts as training material to freshers in the software testing process
* It is also a good marketing & sales strategy to showcase Test Documentation to exhibit a mature testing process
* Test documentation helps you to offer a quality product to the client within specific time limits
* In Software Engineering, Test Documentation also helps to configure or set-up the program through the configuration document and operator manuals
* Test documentation helps you to improve transparency with the client

## Disadvantages of Test Documentation

* The cost of the documentation may surpass its value as it is very time-consuming
* Many times, it is written by people who can't write well or who don't know the material
* Keeping track of changes requested by the client and updating corresponding documents is tiring.
* Poor documentation directly reflects the quality of the product as a misunderstanding between the client and the organization can occur

## Summary

* Test documentation is documentation of artifacts created before or during the testing of software.
* The degree of test formality depends on 1) the type of application under test 2) standards followed by your organization 3) the maturity of the development process.
* Important types of Test Documents are Test policy, Test strategy, Test plan, Test case etc.
* QA team needs to be involved in the initial phase of the project so that Test Documentation is created in parallel
* The main reason behind creating test documentation is to either reduce or remove any uncertainties about the testing activities.
* The cost of the documentation may surpass its value as it is very time-consuming

## What is a Test Scenario?

A **TEST SCENARIO** is defined as any functionality that can be tested. It is also called *Test Condition* or *Test Possibility*. As a tester, you should put yourself in the end user’s shoes and figure out the real-world scenarios and use cases of the Application Under Test.

### What is Scenario Testing?

Scenario Testing is a variant of Software Testing where Scenarios are Used for Testing. Scenarios help in an Easier Way of Testing of the more complicated Systems

Let’s study this with the help of the video below -

## Why create Test Scenarios?

Test Scenarios are created for the following reasons,

* Creating Test Scenarios ensures complete Test Coverage
* Test Scenarios can be approved by various stakeholders like Business Analyst, Developers, Customers to ensure the Application Under Test is thoroughly tested. It ensures that the software is working for the most common use cases.
* They serve as a quick tool to determine the testing work effort and accordingly create a proposal for the client or organize the workforce.
* They help determine the most important end-to-end transactions or the real use of the software applications.
* For studying the end-to-end functioning of the program, Test Scenario is critical.

## When not create Test Scenario?

Test Scenarios may not be created when

* The Application Under Test is complicated, unstable and there is a time crunch in the project.
* Projects that follow Agile Methodology like Scrum, Kanban may not create Test Scenarios.
* Test Scenario may not be created for a new bug fix or [Regression Testing](https://www.guru99.com/regression-testing.html). In such cases, Test Scenarios must be already heavily documented in the previous test cycles. This is especially true for Maintenance projects.

## How to Write Test Scenarios

As a tester, you can follow these five steps to create Test Scenarios-

* **Step 1**: Read the Requirement Documents like BRS, SRS, FRS, of the System Under Test (SUT).  You could also refer uses cases, books, manuals, etc. of the application to be tested.
* **Step 2**: For each requirement, figure out possible users actions and objectives. Determine the technical aspects of the requirement. Ascertain possible scenarios of system abuse and evaluate users with hacker's mindset.
* **Step 3:** After reading the Requirements Document and doing your due Analysis, list out different test scenarios that verify each feature of the software.
* **Step 4:** Once you have listed all possible Test Scenarios, a[Traceability Matrix](https://www.guru99.com/traceability-matrix.html)is created to verify that each & every requirement has a corresponding Test Scenario
* **Step 5:**The scenarios created are reviewed by your supervisor. Later, they are also reviewed by other Stakeholders in the project.

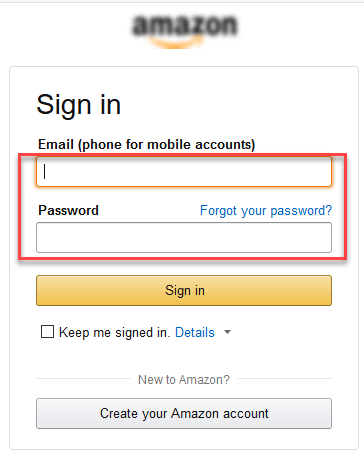
## Tips to Create Test Scenarios

* Each Test Scenario should be tied to a minimum of one Requirement or User Story as per the Project Methodology.
* Before creating a Test Scenario that verifies multiple Requirements at once, ensure you have a Test Scenario that checks that requirement in isolation.
* Avoid creating overly complicated Test Scenarios spanning multiple Requirements.
* The number of scenarios may be large, and it is expensive to run them all. Based on customer priorities only run selected Test Scenarios

## Example 1: Test Scenario for eCommerce Application

For an eCommerce Application, a few test scenarios would be

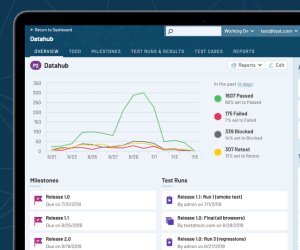
**Test Scenario 1:**Check the Login Functionality

[](https://www.guru99.com/images/1/test_scenario_01.png)

In order to help you understand the difference Test Scenario and Test Cases, specific test cases for this Test Scenario would be

1. Check system behavior when valid email id and password is entered.
2. Check system behavior when invalid email id and valid password is entered.
3. Check system behavior when valid email id and invalid password is entered.
4. Check system behavior when invalid email id and invalid password is entered.
5. Check system behavior when email id and password are left blank and Sign in entered.
6. Check Forgot your password is working as expected
7. Check system behavior when valid/invalid phone number and password is entered.
8. Check system behavior when "Keep me signed" is checked

# How to Write Test Cases: Sample Template with Examples

[](http://guru99.link/recommends-test-case)

## What is a Test Case?

A **TEST CASE** is a set of actions executed to verify a particular feature or functionality of your software application. A Test Case contains test steps, test data, precondition, postcondition developed for specific test scenario to verify any requirement. The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer.

### Test Scenario Vs Test Case

Test scenarios are rather vague and cover a wide range of possibilities. Testing is all about being very specific.

For a [Test Scenario](https://www.guru99.com/test-scenario.html): Check Login Functionality there many possible test cases are:

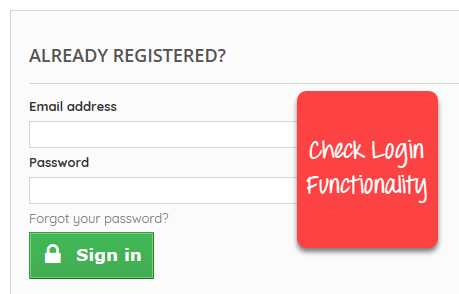
* Test Case 1: Check results on entering valid User Id & Password
* Test Case 2: Check results on entering Invalid User ID & Password
* Test Case 3: Check response when a User ID is Empty & Login Button is pressed, and many more

This is nothing but a Test Case.

Click [here](https://www.guru99.com/faq.html#1) if the video is not accessible

## How to Create a Test Case

Let’s create a Test Case for the scenario: Check Login Functionality

[](https://www.guru99.com/images/1/test-cases_01.png)

**Step 1)** A simple test case for the scenario would be

|  |  |
| --- | --- |
| **Test Case #** | **Test Case Description** |
| 1 | Check response when valid email and password is entered |

**Step 2)** In order to execute the test case, you would need Test Data. Adding it below

|  |  |  |
| --- | --- | --- |
| **Test Case #** | **Test Case Description** | **Test Data** |
| 1 | Check response when valid email and password is entered | Email: [guru99@email.com](mailto:guru99@email.com) Password: lNf9^Oti7^2h |

Identifying test data can be time-consuming and may sometimes require creating test data afresh. The reason it needs to be documented.

**Step 3)** In order to execute a test case, a tester needs to perform a specific set of actions on the AUT. This is documented as below:

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case #** | **Test Case Description** | **Test Steps** | **Test Data** |
| 1 | Check response when valid email and password is entered | 1) Enter Email Address  2) Enter Password  3) Click Sign in | Email: [guru99@email.com](mailto:guru99@email.com)  Password: lNf9^Oti7^2h |

Many times the Test Steps are not simple as above, hence they need documentation. Also, the author of the test case may leave the organization or go on a vacation or is sick and off duty or is very busy with other critical tasks. A recently hire may be asked to execute the test case. Documented steps will help him and also facilitate reviews by other stakeholders.

**Step 4)** The goal of test cases is to check behavior the AUT for an expected result. This needs to be documented as below

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case #** | **Test Case Description** | **Test Data** | **Expected Result** |
| 1 | Check response when valid email and password is entered | Email: [guru99@email.com](mailto:guru99@email.com) Password: lNf9^Oti7^2h | Login should be successful |

During test execution time, the tester will check expected results against actual results and assign a pass or fail status

[](http://guru99.link/recommends-test-case)

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Test Case #** | **Test Case Description** | **Test Data** | **Expected Result** | **Actual Result** | **Pass/Fail** |
| 1 | Check response when valid email and password is entered | Email: [guru99@email.com](mailto:guru99@email.com) Password: lNf9^Oti7^2h | Login should be successful | Login was successful | Pass |

**Step 5)** That apart your test case -may have a field like, Pre - Condition which specifies things that must in place before the test can run. For our test case, a pre-condition would be to have a browser installed to have access to the site under test. A test case may also include Post - Conditions which specifies anything that applies after the test case completes. For our test case, a postcondition would be time & date of login is stored in the database

## The format of Standard Test Cases

Below is a format of a standard login Test case

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Test Case ID** | **Test Scenario** | **Test Steps** | **Test Data** | **Expected Results** | **Actual Results** | **Pass/Fail** |
| TU01 | Check Customer Login with valid Data | 1. Go to site [http://demo.guru99.com](http://demo.guru99.com/) 2. Enter UserId 3. Enter Password 4. Click Submit | Userid = guru99 Password = pass99 | User should Login into an application | As Expected | Pass |
| TU02 | Check Customer Login with invalid Data | 1. Go to site [http://demo.guru99.com](http://demo.guru99.com/) 2. Enter UserId 3. Enter Password 4. Click Submit | Userid = guru99 Password = glass99 | User should not Login into an application | As Expected | Pass |

This entire table may be created in Word, Excel or any other Test management tool. That's all to Test Case Design

**While drafting a test case to include the following information**

* The description of what requirement is being tested
* The explanation of how the system will be tested
* The test setup like a version of an application under test, software, data files, operating system, hardware, security access, physical or logical date, time of day, prerequisites such as other tests and any other setup information pertinent to the requirements being tested
* Inputs and outputs or actions and expected results
* Any proofs or attachments
* Use active case language
* Test Case should not be more than 15 steps
* An automated test script is commented with inputs, purpose and expected results
* The setup offers an alternative to pre-requisite tests
* With other tests, it should be an incorrect business scenario order

## Best Practice for writing good Test Case Example.

**1. Test Cases need to be simple and transparent:**

Create test cases that are as simple as possible. They must be clear and concise as the author of the test case may not execute them.

Use assertive language like go to the home page, enter data, click on this and so on. This makes the understanding the test steps easy and tests execution faster.

**2. Create Test Case with End User in Mind**

The ultimate goal of any software project is to create test cases that meet customer requirements and is easy to use and operate. A tester must create test cases keeping in mind the end user perspective

**3. Avoid test case repetition.**

Do not repeat test cases. If a test case is needed for executing some other test case, call the test case by its test case id in the pre-condition column

**4. Do not Assume**

Do not assume functionality and features of your software application while preparing test case. Stick to the Specification Documents.

**5. Ensure 100% Coverage**

Make sure you write test cases to check all software requirements mentioned in the specification document. Use[Traceability Matrix](https://www.guru99.com/traceability-matrix.html)to ensure no functions/conditions is left untested.

**6. Test Cases must be identifiable.**

Name the test case id such that they are identified easily while tracking defects or identifying a software requirement at a later stage.

**7. Implement Testing Techniques**

It's not possible to check every possible condition in your software application. Software Testing techniques help you select a few test cases with the maximum possibility of finding a defect.

* **Boundary Value Analysis (BVA):** As the name suggests it's the technique that defines the testing of boundaries for a specified range of values.
* **Equivalence Partition (EP):**This technique partitions the range into equal parts/groups that tend to have the same behavior.
* **State Transition Technique**: This method is used when software behavior changes from one state to another following particular action.
* **Error Guessing Technique:** This is guessing/anticipating the error that may arise while doing manual testing. This is not a formal method and takes advantages of a tester's experience with the application

8. **Self-cleaning**

The test case you create must return the[Test Environment](https://www.guru99.com/test-environment-software-testing.html)to the pre-test state and should not render the test environment unusable. This is especially true for configuration testing.

9. **Repeatable** **and self-standing**

The test case should generate the same results every time no matter who tests it

**10. Peer Review.**

After creating test cases, get them reviewed by your colleagues. Your peers can uncover defects in your test case design, which you may easily miss.

## Test Case Management Tools

Test management tools are the automation tools that help to manage and maintain the Test Cases. Main Features of a test case management tool are

1. **For documenting Test Cases:**With tools, you can expedite Test Case creation with use of templates
2. **Execute the Test Case and Record the results:**Test Case can be executed through the tools and results obtained can be easily recorded.
3. **Automate the Defect Tracking:**Failed tests are automatically linked to the bug tracker, which in turn can be assigned to the developers and can be tracked by email notifications.
4. **Traceability:**Requirements, Test cases, Execution of Test cases are all interlinked through the tools, and each case can be traced to each other to check test coverage.
5. **Protecting Test Cases:** Test cases should be reusable and should be protected from being lost or corrupted due to poor version control. Test Case Management Tools offer features like

* Naming and numbering conventions
* Versioning
* Read-only storage
* Controlled access
* Off-site backup

Popular Test Management tools are: [Quality Center](https://www.guru99.com/hp-alm-free-tutorial.html)and [JIRA](https://www.guru99.com/jira-tutorial-a-complete-guide-for-beginners.html)

# What is Test Analysis (Test Basis) in Software Testing?

## What is Test Analysis?

Test Analysis is the process of looking into test artifacts to base your test conditions/test cases. Hence, it is also called Test Basis.

The source from which you derive test information could be

* SRS (Software Requirement Specification)
* BRS (Business Requirement Specification)
* Functional Design Documents

Testers can create Test Conditions by looking into the Application Under Test or use their experience. But mostly, test cases are derived from test artifacts.

#### Let's understand Test Analysis with the help of a case study

Consider a scenario, where the client sends the following

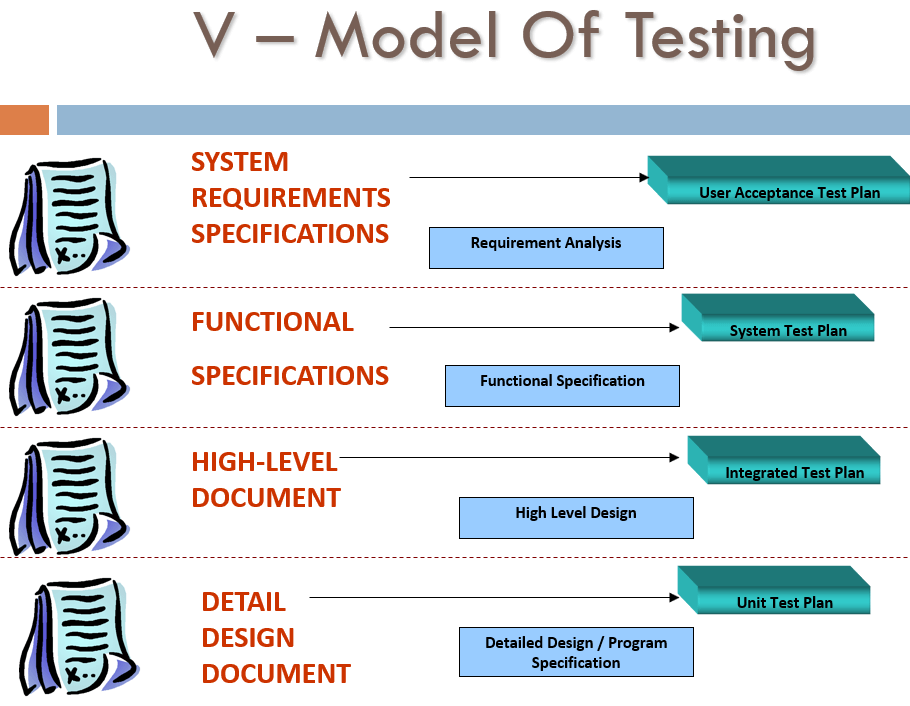
Add search functionality to an eCommerce Store

Even though the application is yet to be developed, try and develop a few test cases for this requirement. Pause here, do your homework and move the solution ahead:

A few test cases among the many you could have thought of are listed below

* Check the search results when no keyword is entered
* Check the search results when no corresponding product is available for the keyword searched
* Check the search results when a number of corresponding products are available for the keyword searched

Here, you look into Test Basis (requirement send by the client), analyze it, and convert it into Test Conditions.  
This is what happens during the different phases of V- Model. Test Plan/Cases are created using the corresponding documents available at different phases.

[](https://www.guru99.com/images/1/test-analysis-v-model.png)Test Analysis in V Model of Testing

# What is Requirements Traceability Matrix (RTM)? Example Template

### What is Traceability Matrix? (TM)

A Traceability Matrix is a document that co-relates any two-baseline documents that require a many-to-many relationship to check the completeness of the relationship.

It is used to track the requirements and to check the current project requirements are met.

## What is Requirement Traceability Matrix?

**Requirement Traceability Matrix (RTM)** is a document that maps and traces user requirement with test cases. It captures all requirements proposed by the client and requirement traceability in a single document, delivered at the conclusion of the Software devlopement life cycle. The main purpose of Requirement Traceability Matrix is to validate that all requirements are checked via test cases such that no functionality is unchecked during Software testing.

In this tutorial, you will learn more about-

* [Why RTM is Important?](https://www.guru99.com/traceability-matrix.html#8)
* [Which Parameters to include in Requirement Traceability Matrix?](https://www.guru99.com/traceability-matrix.html#3)
* [Types of Traceability Test Matrix](https://www.guru99.com/traceability-matrix.html#4)
* [How to create Requirement Traceability Matrix](https://www.guru99.com/traceability-matrix.html#5)
* [Advantage of Requirement Traceability Matrix](https://www.guru99.com/traceability-matrix.html#6)
* [Requirements Traceability Matrix (RTM) Template](https://www.guru99.com/traceability-matrix.html#7)

## Why RTM is Important?

The main agenda of every tester should be to understand the client’s requirement and make sure that the output product should be defect-free. To achieve this goal, every QA should understand the requirement thoroughly and create positive and negative test cases.

This would mean that the software requirements provided by the client have to be further split into different scenarios and further to test cases. Each of this case has to be executed individually.

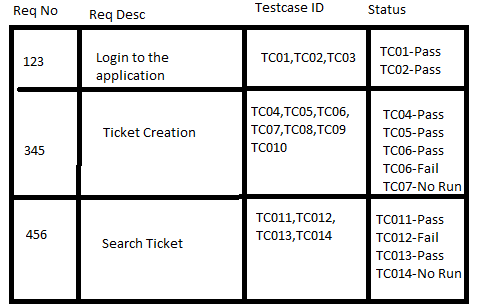
A question arises here on how to make sure that the requirement is tested considering all possible scenarios/cases? How to ensure that any requirement is not left out of the testing cycle?

A simple way is to trace the requirement with its corresponding test scenarios and test cases. This merely is termed as ‘Requirement Traceability Matrix.'

The traceability matrix is typically a worksheet that contains the requirements with its all possible test scenarios and cases and their current state, i.e. if they have been passed or failed. This would help the testing team to understand the level of testing activities done for the specific product.

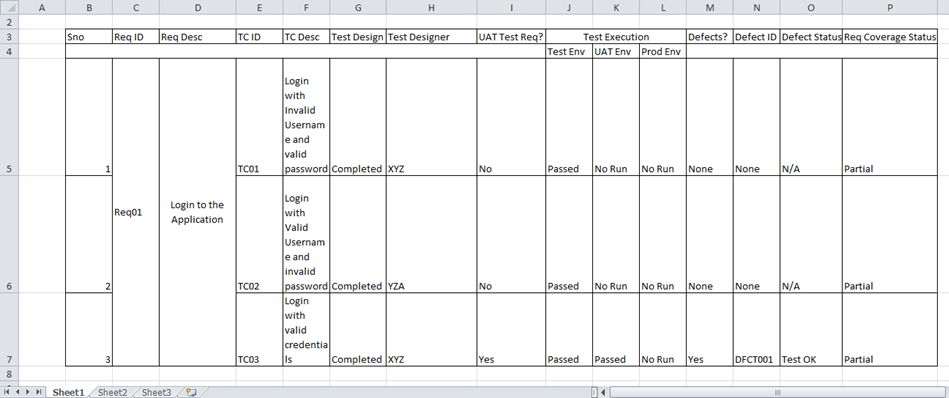
## Which Parameters to include in Requirement Traceability Matrix?

* Requirement ID
* Requirement Type and Description
* Test Cases with Status

[](https://www.guru99.com/images/1/requirements-traceability-matrix.png)

Above is a sample requirement traceability matrix.

But in a typical software testing project, the traceability matrix would have more than these parameters.

[](https://www.guru99.com/images/1/requirements-traceability-matrix002.png)

As illustrated above, a requirement traceability matrix can:

* Show the requirement coverage in the number of test cases
* Design status as well as execution status for the specific test case
* If there is any User Acceptance test to be done by the users, then UAT status can also be captured in the same matrix.
* The related defects and the current state can also be mentioned in the same matrix.

This kind of matrix would be providing **One Stop Shop** for all the testing activities.

Apart from maintaining an excel separately. A testing team can also opt for requirements tracing available Test Management Tools.

## Types of Traceability Test Matrix

In Software Engineering, traceability matrix can be divided into three major component as mentioned below:

* **Forward traceability**: This matrix is used to check whether the project progresses in the desired direction and for the right product. It makes sure that each requirement is applied to the product and that each requirement is tested thoroughly. It maps requirements to test cases.
* **Backward or reverse traceability:** It is used to ensure whether the current product remains on the right track. The purpose behind this type of traceability is to verify that we are not expanding the scope of the project by adding code, design elements, test or other work that is not specified in the requirements. It maps test cases to requirements.
* **Bi-directional traceability ( Forward+Backward):**This traceability matrix ensures that all requirements are covered by test cases. It analyzes the impact of a change in requirements affected by the[Defect](https://www.guru99.com/defect-management-process.html)in a work product and vice versa.

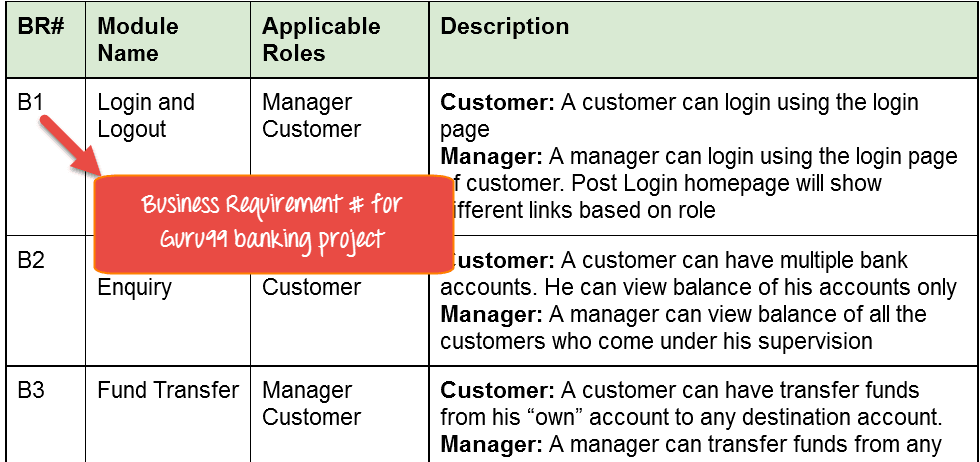
## How to create Requirement Traceability Matrix

Let's understand the concept of Requirement Traceability Matrix through a Guru99 banking project.

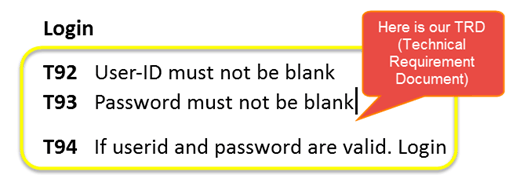
On the basis of **the Business Requirement Document (BRD)** and **Technical Requirement Document (TRD)**, testers start writing test cases.

Let suppose, the following table is our Business Requirement Document or [BRD](https://docs.google.com/document/d/1FsLnZ4thNQF0MhFIFw2Q1KzOTAX89MB8BCvnpsPu5Lc/edit?usp=sharing) for **Guru99 banking project**.

Here the scenario is that the customer should be able to login to Guru99 banking website with the correct password and user#id while manager should be able to login to the website through customer login page.

[](https://www.guru99.com/images/1-2015/012615_1111_Requirement1.png)

While the below table is our **Technical Requirement Document (TRD)**.

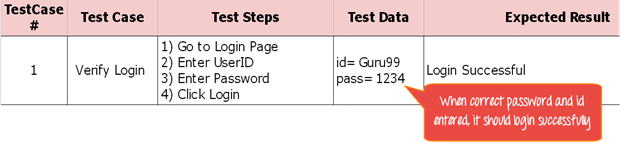
[](https://www.guru99.com/images/1-2015/012615_1111_Requirement2.png)

**Note:** QA teams do not document the BRD and TRD. Also, some companies use **Function Requirement Documents (FRD)** which are similar to Technical Requirement Document but the process of creating Traceability Matrix remains the same.

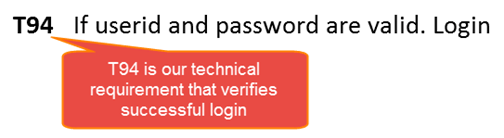
Let's Go Ahead and create RTM in Testing

**Step 1:**Our[sample Test Case](https://www.guru99.com/test-case.html)is

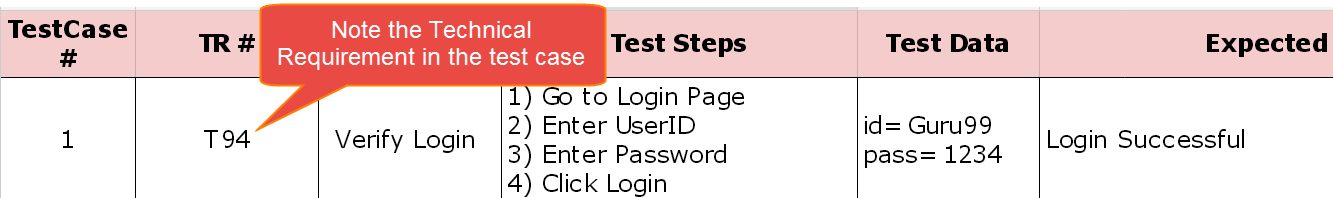
"Verify Login, when correct ID and Password is entered, it should log in successfully"

[](https://www.guru99.com/images/1-2015/012615_1111_Requirement3.png)

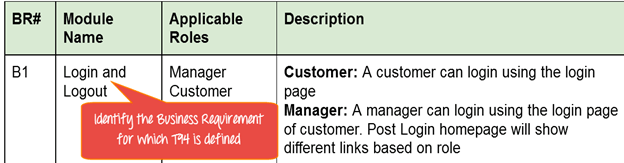
**Step 2**: Identify the Technical Requirement that this test case is verifying. For our test case, the technical requirement is T94 is being verified.

[](https://www.guru99.com/images/1-2015/012615_1111_Requirement4.png)

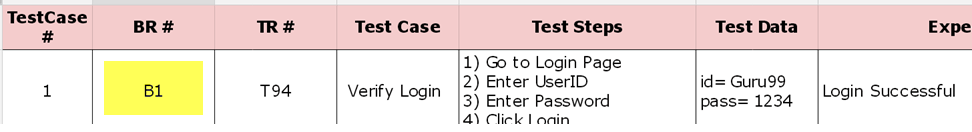
**Step 3:**Note this Technical Requirement (T94) in the Test Case.

[](https://www.guru99.com/images/1-2015/012615_1111_Requirement5.png)

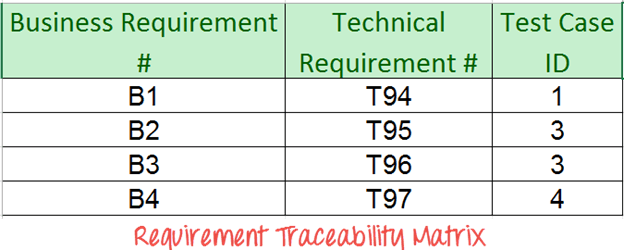
**Step 4:** Identify the Business Requirement for which this TR (Technical Requirement-T94) is defined

[](https://www.guru99.com/images/1-2015/012615_1111_Requirement6.png)

**Step 5:** Note the BR (Business Requirement) in Test Case

[](https://www.guru99.com/images/1-2015/012615_1111_Requirement7.png)

**Step 6:** Do above for all Test Cases. Later Extract the First 3 Columns from your Test Suite. RTM in testing is Ready!

[](https://www.guru99.com/images/1-2015/012615_1111_Requirement8.png)

## Advantage of Requirement Traceability Matrix

* It confirms 100% test coverage
* It highlights any requirements missing or document inconsistencies
* It shows the overall defects or execution status with a focus on business requirements
* It helps in analyzing or estimating the impact on the QA team's work with respect to revisiting or re-working on the test cases

# Test Data Generation: What is, How to, Example, Tools

Everybody knows that testing is a process that produces and consumes large amounts of data. Data used in testing describes the initial conditions for a test and represents the medium through which the tester influences the software. It is a crucial part of most[Functional Testing](https://www.guru99.com/functional-testing.html). But what actually is the test data? Why is it used? Maybe you would wonder ‘Designing Test cases is challenging enough, then why to bother about something as trivial as Test Data’ The purpose of this tutorial is to introduce you to Test Data, its importance and give practical tips and tricks to generate test data quickly. So, Let's Begin!

### What is Test Data? Why is it Important?

Test data is actually the input given to a software program. It represents data that affects or is affected by the execution of the specific module. Some data may be used for positive testing, typically to verify that a given set of input to a given function produces an expected result. Other data may be used for negative testing to test the ability of the program to handle unusual, extreme, exceptional, or unexpected input. Poorly designed testing data may not test all possible test scenarios which will hamper the quality of the software.

[](https://www.guru99.com/images/testdata.jpg)

### What is Test Data Generation?**Why test data should be created before test execution?**

Depending on your testing environment you may need to CREATE Test Data (Most of the times) or at least identify a suitable test data for your test cases (is the test data is already created).

Typically test data is created in-sync with the test case it is intended to be used for.

Test Data can be Generated -

* Manually
* Mass copy of data from production to testing environment
* Mass copy of test data from legacy client systems
* Automated Test Data Generation Tools

Typically sample data should be generated before you begin test execution because it is difficult to handle test data management otherwise. Since in **many testing environments creating test data takes many pre-steps or test environment configurations which is very time-consuming**. Also If test data generation is done ***while***you are in test execution phase you may exceed your testing deadline.

Below are described several testing types together with some suggestions regarding their testing data needs.

### Test Data for White Box Testing

In [White Box Testing](https://www.guru99.com/white-box-testing.html), test data Management is derived from direct examination of the code to be tested. Test data may be selected by taking into account the following things:

* It is desirable to cover as many branches as possible; testing data can be generated such that all branches in the program source code are tested at least once
* Path testing: all paths in the program source code are tested at least once - test data preparation can done to cover as many cases as possible
* Negative [API Testing](https://www.guru99.com/api-testing.html):
  + Testing data may contain invalid parameter types used to call different methods
  + Testing data may consist in invalid combinations of arguments which are used to call the program's methods

### Test Data for Performance Testing

[Performance Testing](https://www.guru99.com/performance-testing.html) is the type of testing which is performed in order to determine how fast system responds under a particular workload. The goal of this type of testing is not to find bugs, but to eliminate bottlenecks.  An important aspect of Performance Testing is that the set of sample data used must be very close to ***'real' or 'live'***data which is used on production. The following question arises: ‘Ok, it’s good to test with real data, but how do I obtain this data?’ The answer is pretty straightforward: from the people who know the best –**the customers**. They may be able to provide some data they already have or, if they don’t have an existing set of data, they may help you by giving feedback regarding how the real-world data might look like. In case you are in a **maintenance testing** project you could copy data from the production environment into the testing bed. It is a good practice to **anonymize**(scramble) sensitive customer data like Social Security Number, Credit Card Numbers, Bank Details etc. while the copy is made.

### Test Data for Security Testing

[Security Testing](https://www.guru99.com/what-is-security-testing.html) is the process that determines if an information system protects data from malicious intent. The set of data that need to be designed in order to fully test a software security must cover the following topics:

* **Confidentiality:** All the information provided by clients is held in the strictest confidence and is not shared with any outside parties. As a short example, if an application uses SSL, you can design a set of test data which verifies that the encryption is done correctly.
* **Integrity:** Determine that the information provided by the system is correct. To design suitable test data you can start by taking an in-depth look at the design, code, databases and file structures.
* **Authentication:** Represents the process of establishing the identity of a user. Testing data can be designed as a different combination of usernames and passwords and its purpose is to check that only the authorized people are able to access the software system.
* **Authorization:** Tells what are the rights of a specific user. Testing data may contain a different combination of users, roles and **operations**in order to check only users with sufficient privileges are able to perform a particular operation.

### Test Data for Black Box Testing

In Black Box Testing the code is not visible to the tester. Your functional test cases can have test data meeting following criteria -

* **No data**: Check system response when no data is submitted
* **Valid data**: Check system response when Valid  test data is submitted
* **Invalid data**: Check system response when InValid  test data is submitted
* **Illegal data format**: Check system response when test data is in an invalid format
* **Boundary Condition Dataset**: Test data meeting boundary value conditions
* **Equivalence Partition Data Set**: Test data qualifying your equivalence partitions.
* **Decision Table Data Set**: Test data qualifying your decision table testing strategy
* **State Transition Test Data Set**: Test data meeting your state transition testing strategy
* **Use Case Test Data**: Test Data in-sync with your use cases.

# Download Sample Test Case Template: Example Excel, Word Formats

A good[Test Case](https://www.guru99.com/test-case.html)template maintains test artifact consistency for the test team and makes it easy for all stakeholders to understand the test cases. Writing test case in a standard format lessen the test effort and the error rate. Test cases format are more desirable in case if you are reviewing test case from experts.

The template chosen for your project depends on your test policy. Many organizations create test cases in Microsoft Excel while some in Microsoft Word. Some even use test management tools like HP ALM to document their test cases.

Click Below to download Test Case XLS

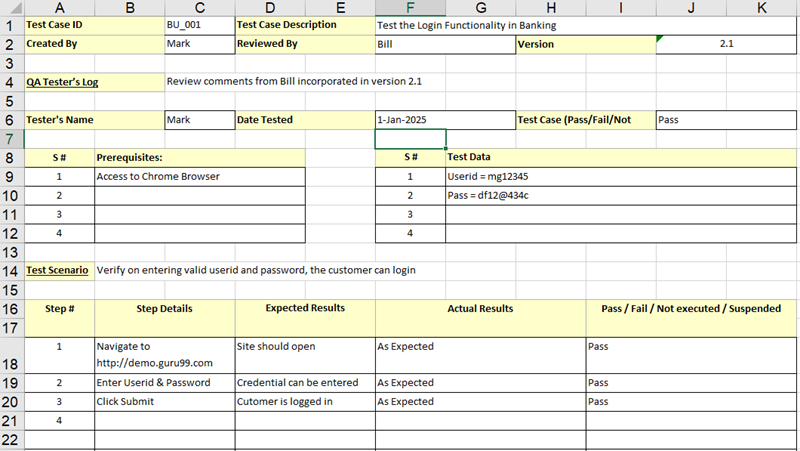
[**Download Test Case Template(.xls)**](https://drive.google.com/uc?export=download&id=0ByI5-ZLwpo25eXFlcU5ZMTJsT28)

Irrespective of the test case documentation method chosen, any good test case template must have the following fields

|  |  |
| --- | --- |
| **Test Case Field** | **Description** |
| **Test case ID:** | * Each test case should be represented by a unique ID. To indicate test types follow some convention like "TC\_UI\_1" indicating "User Interface Test Case#1." |
| **Test Priority:** | * It is useful while executing the test.   + Low   + Medium   + High |
| **Name of the Module**: | * Determine the name of the main module or sub-module being tested |
| **Test Designed by**: | * Tester's Name |
| **Date of test designed**: | * Date when test was designed |
| **Test Executed by**: | * Who executed the test- tester |
| **Date of the Test Execution**: | * Date when test needs to be executed |
| **Name or Test Title**: | * Title of the test case |
| **Description/Summary of Test**: | * Determine the summary or test purpose in brief |
| **Pre-condition**: | * Any requirement that needs to be done before execution of this test case. To execute this test case list all pre-conditions |
| **Dependencies**: | * Determine any dependencies on test requirements or other test cases |
| **Test Steps**: | * Mention all the test steps in detail and write in the order in which it requires to be executed. While writing test steps ensure that you provide as much detail as you can |
| **Test Data**: | * Use of test data as an input for the test case. Deliver different data sets with precise values to be used as an input |
| **Expected Results**: | * Mention the expected result including error or message that should appear on screen |
| **Post-Condition**: | * What would be the state of the system after running the test case? |
| **Actual Result**: | * After test execution, actual test result should be filled |
| **Status (Fail/Pass):** | * Mark this field as failed, if actual result is not as per the estimated result |
| **Notes**: | * If there are some special condition which is left in above field |

**Optionally you can have the following fields depending on the project requirements**

* **Link / Defect ID**: Include the link for[Defect](https://www.guru99.com/defect-management-process.html)or determine the defect number if test status is fail
* **Keywords / Test Type**: To determine tests based on test types this field can be used. Eg: Usability, functional, business rules, etc.
* **Requirements**: Requirements for which this test case is being written
* **References / Attachments**: It is useful for complicated test scenarios, give the actual path of the document or diagram
* **Automation ( Yes/No)**: To track automation status when test cases are automated
* **Custom Fields**: Fields particular your project being tested due to client/project requirements.

[](https://www.guru99.com/images/6-2015/052615_1220_DownloadSam1.png)

## **Testing Techniques:**

# Software Testing Techniques with Test Case Design Examples

## What is Software Testing Technique?

Software Testing Techniques help you design better test cases. Since exhaustive testing is not possible; Manual Testing Techniques help reduce the number of test cases to be executed while increasing test coverage. They help identify test conditions that are otherwise difficult to recognize.

In this tutorial, you will learn 5 important software testing techniques:

* [Boundary Value Analysis (BVA)](https://www.guru99.com/software-testing-techniques.html#1)
* [Equivalence Class Partitioning](https://www.guru99.com/software-testing-techniques.html#2)
* [Decision Table based testing.](https://www.guru99.com/software-testing-techniques.html#3)
* [State Transition](https://www.guru99.com/software-testing-techniques.html#4)
* [Error Guessing](https://www.guru99.com/software-testing-techniques.html#5)

### Boundary Value Analysis (BVA)

Boundary value analysis is based on testing at the boundaries between partitions. It includes maximum, minimum, inside or outside boundaries, typical values and error values.

It is generally seen that a large number of errors occur at the boundaries of the defined input values rather than the center. It is also known as BVA and gives a selection of test cases which exercise bounding values.

This black box testing technique complements equivalence partitioning. This software testing technique base on the principle that, if a system works well for these particular values then it will work perfectly well for all values which comes between the two boundary values.

**Guidelines for Boundary Value analysis**

* If an input condition is restricted between values x and y, then the test cases should be designed with values x and y as well as values which are above and below x and y.
* If an input condition is a large number of values, the test case should be developed which need to exercise the minimum and maximum numbers. Here, values above and below the minimum and maximum values are also tested.
* Apply guidelines 1 and 2 to output conditions. It gives an output which reflects the minimum and the maximum values expected. It also tests the below or above values.

**Example:**

Input condition is valid between 1 to 10

Boundary values 0,1,2 and 9,10,11

### Equivalence Class Partitioning

Equivalent Class Partitioning allows you to divide set of test condition into a partition which should be considered the same. This software testing method divides the input domain of a program into classes of data from which test cases should be designed.

The concept behind this technique is that test case of a representative value of each class is equal to a test of any other value of the same class. It allows you to Identify valid as well as invalid equivalence classes.

**Example:**

Input conditions are valid between

1 to 10 and 20 to 30

Hence there are five equivalence classes

--- to 0 (invalid)

1 to 10 (valid)

11 to 19 (invalid)

20 to 30 (valid)

31 to --- (invalid)

You select values from each class, i.e.,

-2, 3, 15, 25, 45

### Decision Table Based Testing.

A decision table is also known as to Cause-Effect table. This software testing technique is used for functions which respond to a combination of inputs or events. For example, a submit button should be enabled if the user has entered all required fields.

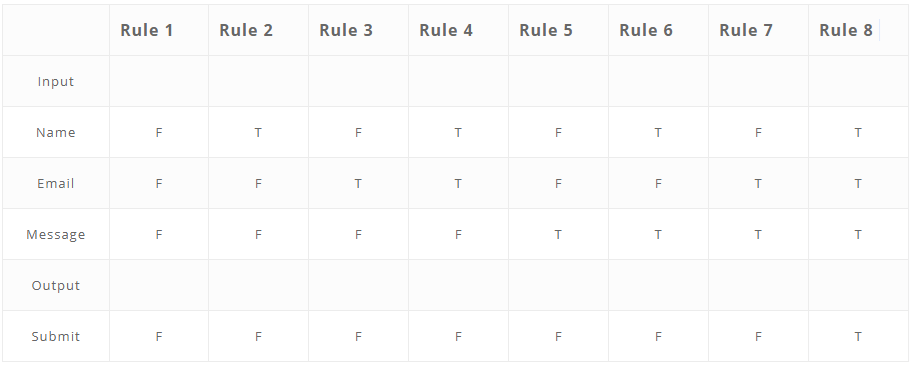
The first task is to identify functionalities where the output depends on a combination of inputs. If there are large input set of combinations, then divide it into smaller subsets which are helpful for managing a decision table.

For every function, you need to create a table and list down all types of combinations of inputs and its respective outputs. This helps to identify a condition that is overlooked by the tester.

**Following are steps to create a decision table:**

* Enlist the inputs in rows
* Enter all the rules in the column
* Fill the table with the different combination of inputs
* In the last row, note down the output against the input combination.

**Example**: A submit button in a contact form is enabled only when all the inputs are entered by the end user.

[](https://www.guru99.com/images/1/053018_0554_SoftwareTes1.png)

### State Transition

In State Transition technique changes in input conditions change the state of the Application Under Test (AUT). This testing technique allows the tester to test the behavior of an AUT. The tester can perform this action by entering various input conditions in a sequence. In State transition technique, the testing team provides positive as well as negative input test values for evaluating the system behavior.

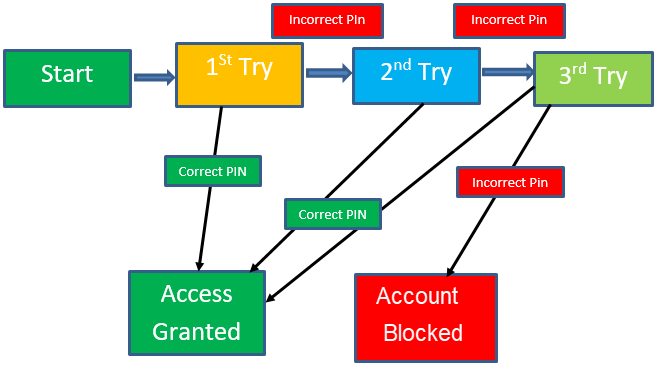
**Guideline for State Transition:**

* State transition should be used when a testing team is testing the application for a limited set of input values.
* The technique should be used when the testing team wants to test sequence of events which happen in the application under test.

**Example:**

In the following example, if the user enters a valid password in any of the first three attempts the user will be able to log in successfully. If the user enters the invalid password in the first or second try, the user will be prompted to re-enter the password. When the user enters password incorrectly 3rd time, the action has taken, and the account will be blocked.

### State transition diagram

[](https://www.guru99.com/images/1/053018_0554_SoftwareTes2.png)

In this diagram when the user gives the correct PIN number, he or she is moved to Access granted state. Following Table is created based on the diagram above-

### State Transition Table

|  |  |  |
| --- | --- | --- |
|  | **Correct PIN** | **Incorrect PIN** |
| **S1) Start** | **S5** | **S2** |
| **S2) 1st attempt** | **S5** | **S3** |
| **S3) 2nd attempt** | **S5** | **S4** |
| **S4) 3rd attempt** | **S5** | **S6** |
| **S5) Access Granted** | **-** | **-** |
| **S6) Account blocked** | **-** | **-** |

In the above-given table when the user enters the correct PIN, the state is transitioned to Access granted. And if the user enters an incorrect password, he or she is moved to next state. If he does the same 3rd time, he will reach the account blocked state.

### Error Guessing

Error guessing is a software testing technique which is based on guessing the error which can prevail in the code. It is an experience-based technique where the test analyst uses his/her or experience to guess the problematic part of the testing application.

The technique counts a list of possible errors or error-prone situations. Then tester writes a test case to expose those errors. To design test cases based on this software testing technique, the analyst can use the past experiences to identify the conditions.

**Guidelines for Error Guessing:**

* The test should use the previous experience of testing similar applications
* Understanding of the system under test
* Knowledge of typical implementation errors
* Remember previously troubled areas
* Evaluate Historical data & Test results

**Conclusion**

* Software testing Techniques allow you to design better cases. There are five primarily used techniques.
* Boundary value analysis is testing at the boundaries between partitions.
* Equivalent Class Partitioning allows you to divide set of test condition into a partition which should be considered the same.
* Decision Table software testing technique is used for functions which respond to a combination of inputs or events.
* In State Transition technique changes in input conditions change the state of the Application Under Test (AUT)
* Error guessing is a software testing technique which is based on guessing the error which can prevail in the code.

# Boundary Value Analysis & Equivalence Partitioning with Examples

Practically, due to time and budget considerations, it is not possible to perform exhausting testing for each set of test data, especially when there is a large pool of input combinations.

* We need an easy way or special techniques that can select test cases intelligently from the pool of test-case, such that all test scenarios are covered.
* We use two techniques - **Equivalence Partitioning & Boundary Value Analysis testing techniques** to achieve this.

In this tutorial, we will learn

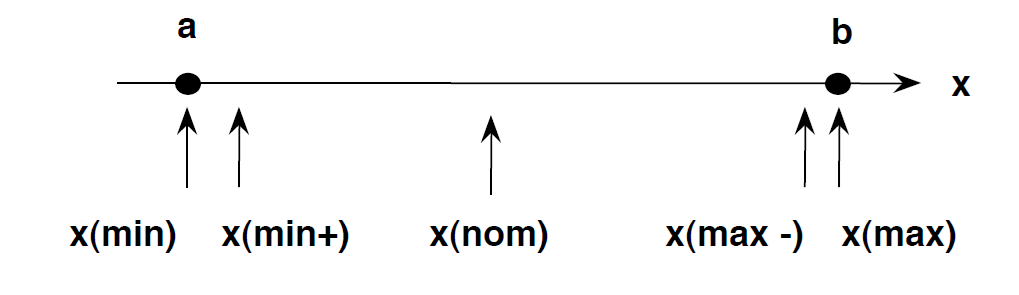
* [What is Boundary Testing?](https://www.guru99.com/equivalence-partitioning-boundary-value-analysis.html#1)
* [What is Equivalent Class Partitioning?](https://www.guru99.com/equivalence-partitioning-boundary-value-analysis.html#2)
* [Example 1: Equivalence and Boundary Value](https://www.guru99.com/equivalence-partitioning-boundary-value-analysis.html#3)
* [Example 2: Equivalence and Boundary Value](https://www.guru99.com/equivalence-partitioning-boundary-value-analysis.html#4)
* [Why Equivalence & Boundary Analysis Testing](https://www.guru99.com/equivalence-partitioning-boundary-value-analysis.html#5)

### What is Boundary Testing?

Boundary testing is the process of testing between extreme ends or boundaries between partitions of the input values.

* So these extreme ends like Start- End, Lower- Upper, Maximum-Minimum, Just Inside-Just Outside values are called boundary values and the testing is called "boundary testing".
* The basic idea in boundary value testing is to select input variable values at their:

1. Minimum
2. Just above the minimum
3. A nominal value
4. Just below the maximum
5. Maximum

[](https://www.guru99.com/images/3-2016/032316_0620_Equivalence1.png)

* In Boundary Testing, Equivalence Class Partitioning plays a good role
* Boundary Testing comes after the Equivalence Class Partitioning.

### What is Equivalent Class Partitioning?

Equivalent Class Partitioning is a black box technique (code is not visible to tester) which can be applied to all levels of testing like unit, integration, system, etc. In this technique, you divide the set of test condition into a partition that can be considered the same.

* It divides the input data of software into different equivalence data classes.
* You can apply this technique, where there is a range in the input field.

### Example 1: Equivalence and Boundary Value

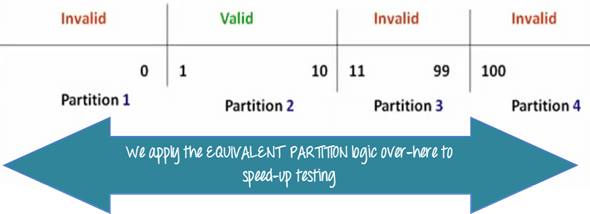
* Let's consider the behavior of Order Pizza Text Box Below
* Pizza values 1 to 10 is considered valid. A success message is shown.
* While value 11 to 99 are considered invalid for order and an error message will appear, **"Only 10 Pizza can be ordered"**

**Order Pizza:**     

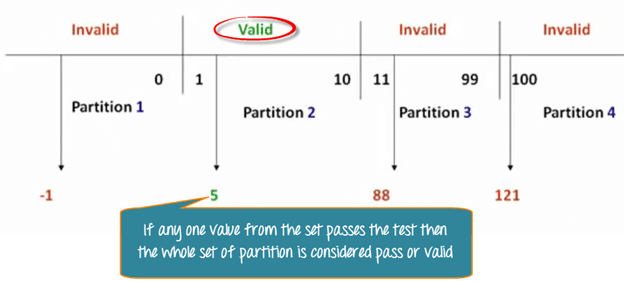
**Here is the test condition**

1. Any Number greater than 10 entered in the Order Pizza field(let say 11) is considered invalid.
2. Any Number less than 1 that is 0 or below, then it is considered invalid.
3. Numbers 1 to 10 are considered valid
4. Any 3 Digit Number say -100 is invalid.

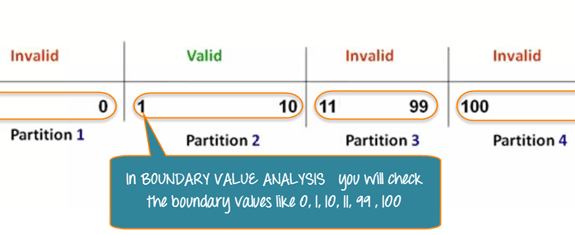
We cannot test all the possible values because if done, the number of test cases will be more than 100. To address this problem, we use equivalence partitioning hypothesis where we divide the possible values of tickets into groups or sets as shown below where the system behavior can be considered the same.

[](https://www.guru99.com/images/3-2016/032316_0620_Equivalence4.png)

The divided sets are called Equivalence Partitions or Equivalence Classes. Then we pick only one value from each partition for testing. The hypothesis behind this technique is **that if one condition/value in a partition passes all others will also pass**. Likewise**, if one condition in a partition fails, all other conditions in that partition will fail**.

[](https://www.guru99.com/images/3-2016/032316_0620_Equivalence5.png)

**Boundary Value Analysis**- in Boundary Value Analysis, you test boundaries between equivalence partitions

[](https://www.guru99.com/images/3-2016/032316_0620_Equivalence6.png)

In our earlier example instead of checking, one value for each partition you will check the values at the partitions like 0, 1, 10, 11 and so on. As you may observe, you test values at**both valid and invalid boundaries**. Boundary Value Analysis is also called**range checking**.

Equivalence partitioning and boundary value analysis(BVA) are closely related and can be used together at all levels of testing.

### Example 2: Equivalence and Boundary Value

Following password field accepts minimum 6 characters and maximum 10 characters

That means results for values in partitions 0-5, 6-10, 11-14 should be equivalent

**Enter Password:**      

|  |  |  |
| --- | --- | --- |
| **Test Scenario #** | **Test Scenario Description** | **Expected Outcome** |
| 1 | Enter 0 to 5 characters in password field | System should not accept |
| 2 | Enter 6 to 10 characters in password field | System should accept |
| 3 | Enter 11 to 14 character in password field | System should not accept |

### Examples 3: Input Box should accept the Number 1 to 10

Here we will see the Boundary Value Test Cases

|  |  |
| --- | --- |
| **Test Scenario Description** | **Expected Outcome** |
| Boundary Value = 0 | System should NOT accept |
| Boundary Value = 1 | System should accept |
| Boundary Value = 2 | System should accept |
| Boundary Value = 9 | System should accept |
| Boundary Value = 10 | System should accept |
| Boundary Value = 11 | System should NOT accept |

### Why Equivalence & Boundary Analysis Testing

1. This testing is used to reduce a very large number of test cases to manageable chunks.
2. Very clear guidelines on determining test cases without compromising on the effectiveness of testing.
3. Appropriate for calculation-intensive applications with a large number of variables/inputs

**Summary:**

* Boundary Analysis testing is used when practically it is impossible to test a large pool of test cases individually
* Two techniques - Equivalence Partitioning & Boundary Value Analysis testing techniques are used
* In Equivalence Partitioning, first, you divide a set of test condition into a partition that can be considered.
* In Boundary Value Analysis you then test boundaries between equivalence partitions
* Appropriate for calculation-intensive applications with variables that represent physical quantities

# Decision Table Testing: Learn with Example

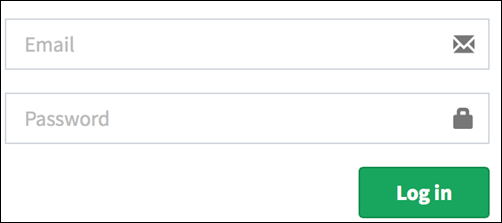
## What is Decision Table Testing?

Decision table testing is a software testing technique used to test system behavior for different input combinations. This is a systematic approach where the different input combinations and their corresponding system behavior (Output) are captured in a tabular form. That is why it is also called as a **Cause-Effect** table where Cause and effects are captured for better test coverage.

A Decision Table is a tabular representation of inputs versus rules/cases/test conditions. Let's learn with an example.

## Example 1: How to make Decision Base Table for Login Screen

Let's create a decision table for a login screen.

[](https://www.guru99.com/images/1/120817_0759_DecisionTab1.png)

The condition is simple if the user provides correct username and password the user will be redirected to the homepage. If any of the input is wrong, an error message will be displayed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Conditions** | **Rule 1** | **Rule 2** | **Rule 3** | **Rule 4** |
| **Username (T/F)** | F | T | F | T |
| **Password (T/F)** | F | F | T | T |
| **Output (E/H)** | E | E | E | H |

Legend:

* T – Correct username/password
* F – Wrong username/password
* E – Error message is displayed
* H – Home screen is displayed

Interpretation:

* Case 1 – Username and password both were wrong. The user is shown an error message.
* Case 2 – Username was correct, but the password was wrong. The user is shown an error message.
* Case 3 – Username was wrong, but the password was correct. The user is shown an error message.
* Case 4 – Username and password both were correct, and the user navigated to homepage

While converting this to test case, we can create 2 scenarios ,

* Enter correct username and correct password and click on login, and the expected result will be the user should be navigated to homepage

And one from the below scenario

* Enter wrong username and wrong password and click on login, and the expected result will be the user should get an error message
* Enter correct username and wrong password and click on login, and the expected result will be the user should get an error message
* Enter wrong username and correct password and click on login, and the expected result will be the user should get an error message

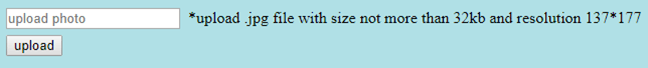
As they essentially test the same rule.

## Example 2: How to make Decision Table for Upload Screen

Now consider a dialogue box which will ask the user to upload photo with certain conditions like –

1. You can upload only '.jpg' format image
2. file size less than 32kb
3. resolution 137\*177.

If any of the conditions fails the system will throw corresponding error message stating the issue and if all conditions are met photo will be updated successfully

[](https://www.guru99.com/images/1/120817_0759_DecisionTab2.png)

Let's create the decision table for this case.

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Conditions** | **Case 1** | **Case 2** | **Case 3** | **Case 4** | **Case 5** | **Case 6** | **Case 7** | **Case 8** |
| **Format** | .jpg | .jpg | .jpg | .jpg | Not .jpg | Not .jpg | Not .jpg | Not .jpg |
| **Size** | Less than 32kb | Less than 32kb | >= 32kb | >= 32kb | Less than 32kb | Less than 32kb | >= 32kb | >= 32kb |
| **resolution** | 137\*177 | Not 137\*177 | 137\*177 | Not 137\*177 | 137\*177 | Not 137\*177 | 137\*177 | Not 137\*177 |
| **Output** | Photo uploaded | Error message resolution mismatch | Error message size mismatch | Error message size and resolution mismatch | Error message for format mismatch | Error message format and resolution mismatch | Error message for format and size mismatch | Error message for format, size, and resolution mismatch |

For this condition, we can create 8 different test cases and ensure complete coverage based on the above table.

1. Upload a photo with format '.jpg', size less than 32kb and resolution 137\*177 and click on upload. Expected result is Photo should upload successfully
2. Upload a photo with format '.jpg', size less than 32kb and resolution not 137\*177 and click on upload. Expected result is Error message resolution mismatch should be displayed
3. Upload a photo with format '.jpg', size more than 32kb and resolution 137\*177 and click on upload. Expected result is Error message size mismatch should be displayed
4. Upload a photo with format '.jpg', size more than equal to 32kb and resolution not 137\*177 and click on upload. Expected result is Error message size and resolution mismatch should be displayed
5. Upload a photo with format other than '.jpg', size less than 32kb and resolution 137\*177 and click on upload. Expected result is Error message for format mismatch should be displayed
6. Upload a photo with format other than '.jpg', size less than 32kb and resolution not 137\*177 and click on upload. Expected result is Error message format and resolution mismatch should be displayed
7. Upload a photo with format other than '.jpg', size more than 32kb and resolution 137\*177 and click on upload. Expected result is Error message for format and size mismatch should be displayed
8. Upload a photo with format other than '.jpg', size more than 32kb and resolution not 137\*177 and click on upload. Expected result is Error message for format, size and resolution mismatch should be displayed

## Why is Decision Table Testing is important?

This testing technique becomes important when it is required to test different combination. It also helps in better test coverage for complex business logic.

In Software Engineering, boundary value and equivalent partition are other similar techniques used to ensure better coverage. They are used if the system shows the **same**behavior for a large set of inputs. However, in a system where for each set of input values the system behavior is **different**, boundary value and equivalent partitioning technique are not effective in ensuring good test coverage.

In this case, decision table testing is a good option. This technique can make sure of good coverage, and the representation is simple so that it is easy to interpret and use.

This table can be used as the reference for the requirement and for the functionality development since it is easy to understand and cover all the combinations.

The significance of this technique becomes immediately clear as the number of inputs increases. Number of possible Combinations is given by 2 ^ n , where n is the number of Inputs. For n = 10, which is very common in the web based testing, having big input forms, the number of combinations will be 1024. Obviously, you cannot test all but you will choose a rich sub-set of the possible combinations using decision based testing technique.

## Advantages of Decision Table Testing

* When the system behavior is different for different input and not same for a range of inputs, both equivalent partitioning, and boundary value analysis won't help, but decision table can be used.
* The representation is simple so that it can be easily interpreted and is used for development and business as well.
* This table will help to make effective combinations and can ensure a better coverage for testing
* Any complex business conditions can be easily turned into decision tables
* In a case we are going for 100% coverage typically when the input combinations are low, this technique can ensure the coverage.

## Disadvantages of Decision Table Testing

The main disadvantage is that when the number of input increases the table will become more complex

# What is State Transition Testing? Diagram, Technique, Example

## What is State Transition in Testing?

State Transition testing is defined as the software testing technique in which changes in input conditions cause's state changes in the Application under Test (AUT).

It is a black box testing technique in which the tester analyzes the behavior of an application under test for different input conditions in a sequence. In this technique, tester provides both positive and negative input test values and record the system behavior.

It is the model on which the system and the tests are based. Any system where you get a different output for the same input, depending on what has happened before, is a finite state system.

**State Transition Testing Technique**is helpful where you need to **test different system transitions.**

In this tutorial, you will learn-

* [What is State Transition in Testing?](https://www.guru99.com/state-transition-testing.html#1)
* [When to Use State Transition?](https://www.guru99.com/state-transition-testing.html#2)
* [When to Not Rely On State Transition?](https://www.guru99.com/state-transition-testing.html#3)
* [Four Parts Of State Transition Diagram](https://www.guru99.com/state-transition-testing.html#4)
* [State Transition Diagram and State Transition Table](https://www.guru99.com/state-transition-testing.html#5)
* [How to Make a State Transition (Examples of a State Transition)](https://www.guru99.com/state-transition-testing.html#6)
* [Advantages and Disadvantages of State Transition Technique](https://www.guru99.com/state-transition-testing.html#7)

## When to Use State Transition?

* This can be used when a tester is testing the application for a finite set of input values.
* When the tester is trying to test sequence of events that occur in the application under test. I.e., this will allow the tester to test the application behavior for a sequence of input values.
* When the system under test has a dependency on the events/values in the past.

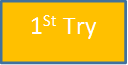
## When to Not Rely On State Transition?

* When the testing is not done for sequential input combinations.
* If the testing is to be done for different functionalities like exploratory testing

## Four Parts Of State Transition Diagram

There are 4 main components of the State Transition Model as below

**1) States** that the software might get

[](https://www.guru99.com/images/1/103017_0527_WhatIsState1.png)

**2) Transition** from one state to another

[](https://www.guru99.com/images/1/103017_0527_WhatIsState2.png)

**3) Events** that origin a transition like closing a file or withdrawing money

[](https://www.guru99.com/images/1/103017_0527_WhatIsState3.png)

**4) Actions** that result from a transition (an error message or being given the cash.)

[](https://www.guru99.com/images/1/103017_0527_WhatIsState4.png)

## State Transition Diagram and State Transition Table

There are two main ways to represent or design state transition, State transition diagram, and state transition table.

In state transition diagram the states are shown in boxed texts, and the transition is represented by arrows. It is also called State Chart or Graph. It is useful in identifying valid transitions.

In state transition table all the states are listed on the left side, and the events are described on the top. Each cell in the table represents the state of the system after the event has occurred. It is also called State Table. It is useful in identifying invalid transitions.

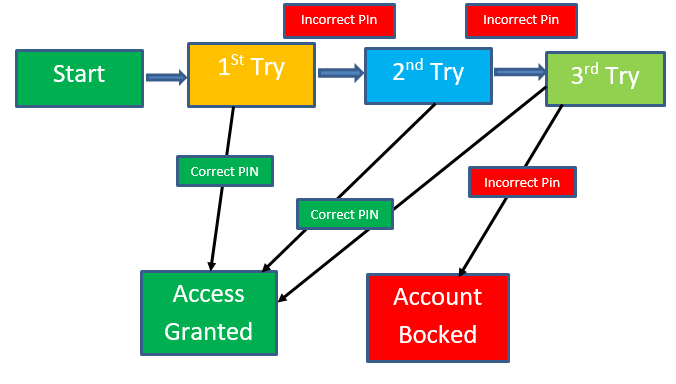
## How to Make a State Transition (Examples of a State Transition)

### Example 1:

Let's consider an ATM system function where if the user enters the invalid password three times the account will be locked.

In this system, if the user enters a valid password in any of the first three attempts the user will be logged in successfully. If the user enters the invalid password in the first or second try, the user will be asked to re-enter the password. And finally, if the user enters incorrect password 3rd time, the account will be blocked.

### State transition diagram

[](https://www.guru99.com/images/1/103017_0527_WhatIsState7.png)

In the diagram whenever the user enters the correct PIN he is moved to Access granted state, and if he enters the wrong password he is moved to next try and if he does the same for the 3rd time the account blocked state is reached.

### State Transition Table

|  |  |  |
| --- | --- | --- |
|  | **Correct PIN** | **Incorrect PIN** |
| **S1) Start** | **S5** | **S2** |
| **S2) 1st attempt** | **S5** | **S3** |
| **S3) 2nd attempt** | **S5** | **S4** |
| **S4) 3rd attempt** | **S5** | **S6** |
| **S5) Access Granted** | **-** | **-** |
| **S6) Account blocked** | **-** | **-** |

In the table when the user enters the correct PIN, state is transitioned to S5 which is Access granted. And if the user enters a wrong password he is moved to next state. If he does the same 3rd time, he will reach the account blocked state.

# What is Use Case Testing? Technique, Examples

Before we Learn What Use Case Testing is, let’s understand

### What is a Use Case?

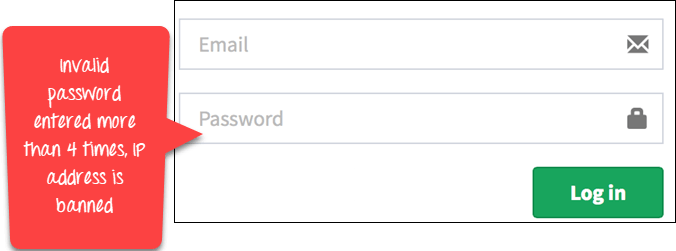
A use case is a description of a particular use of the system by an actor or user. It is used widely in developing tests at system or acceptance level.

### What is Use Case Testing?

Use Case Testing is defined as a software testing technique, that helps identify test cases that cover the entire system, on a transaction by transaction basis from start to the finishing point.

## How to do Use Case Testing: Example

In a use-case, an actor is represented by "A" and system by "S". We create Use for a login functionality of a Web Application as shown below

[](https://www.guru99.com/images/1/use-case-testing-01.png)

|  |  |  |
| --- | --- | --- |
| **Main Success Scenario** | **Step** | **Description** |
| A:Actor S:System | 1 | **A: Enter Agent Name & Password** |
| 2 | **S: Validate Password** |
| 3 | **S: Allow Account Access** |
| Extensions | 2a | **Password not valid** S: Display Message and ask for re-try 4 times |
| 2b | **Password not valid 4 times** S: Close Application |

* Consider the first step of an end to end scenario for a login functionality for our web application where the Actor enters email and password.
* In the next step, the system will validate the password
* Next, if the password is correct, the access will be granted
* There can be an extension of this use case. In case password is not valid system will display a message and ask for re-try four times
* If Password, not valid four times system will ban the IP address.

Here we will test the success scenario and one case of each extension.

This is USE-Case testing in Software Engineering

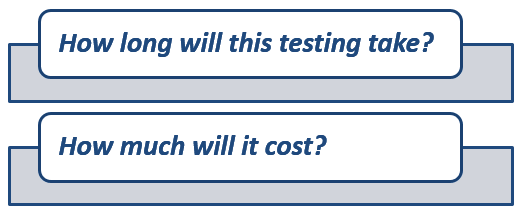
**Test Management & Control**

### What is Software Test Estimation?

Test Estimation is a management activity which approximates**how long** a Task would take to complete. Estimating effort for the test is one of the **major** and **important** tasks in Test Management.

## Why Test Estimation?

Two questions you can expect from your clients when discussing potential test engagements are

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_2_1.png)

For small projects, these questions are relatively easy to answer. But for the big project like[Testing](https://www.guru99.com/software-testing.html)Guru99 Bank website, you must think hard to answer those questions.

In this tutorial, you will learn-

* [What is Software Test Estimation?](https://www.guru99.com/an-expert-view-on-test-estimation.html#1)
* [Why Test Estimation?](https://www.guru99.com/an-expert-view-on-test-estimation.html#2)
* [What to Estimate?](https://www.guru99.com/an-expert-view-on-test-estimation.html#3)
* [How to estimate?](https://www.guru99.com/an-expert-view-on-test-estimation.html#4)
  + [Step 1) Divide the whole project task into subtasks](https://www.guru99.com/an-expert-view-on-test-estimation.html#5)
  + [Step 2) Allocate each task to team member](https://www.guru99.com/an-expert-view-on-test-estimation.html#6)
  + [Step 3) Effort Estimation For Tasks](https://www.guru99.com/an-expert-view-on-test-estimation.html#7)
    - [Method 1) Function Point Method](https://www.guru99.com/an-expert-view-on-test-estimation.html#8)
    - [Method 2) Three Point Estimation](https://www.guru99.com/an-expert-view-on-test-estimation.html#9)
  + [Step 4) Validate the estimation](https://www.guru99.com/an-expert-view-on-test-estimation.html#10)
* [Test estimation best practices](https://www.guru99.com/an-expert-view-on-test-estimation.html#11)
* [Other Techniques](https://www.guru99.com/an-expert-view-on-test-estimation.html#12)

## What to Estimate?

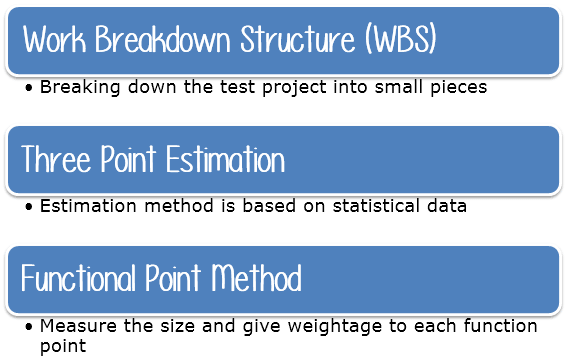
[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_2_2.jpg)

* **Resources:**Resources are required to **carry out** any project tasks. They can be people, equipment, facilities, funding, or anything else capable of definition required for the completion of a project activity.
* **Times :**Time is the most valuable resource in a project. Every project has a  deadline to delivery.
* **Human Skills :**Human skills mean the **knowledge** and the **experience** of the Team members. They affect to your estimation. For example, a team, whose members have low testing skills, will take more time to finish the project than the one which has high testing skills.
* **Cost:**Cost is the project **budget**. Generally speaking, it means **how much** **money** it takes to finish the project.

## How to estimate?

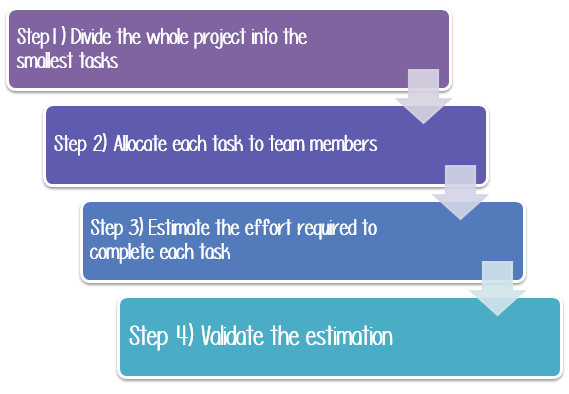
List of Software Test Estimation Techniques

* Work Breakdown Structure
* 3-Point Software Testing Estimation Technique
* Wideband Delphi technique
* Function Point/Testing Point Analysis
* Use – Case Point Method
* Percentage distribution
* Ad-hoc method

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_2_3.png)

Following is the 4 Step process to arrive at an estimate

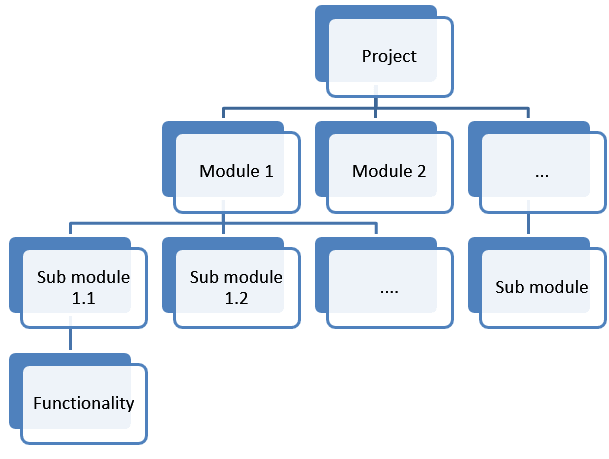
You will learn how to combine these techniques to find the estimate for Guru99 Bank case study.

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_2_4.png)

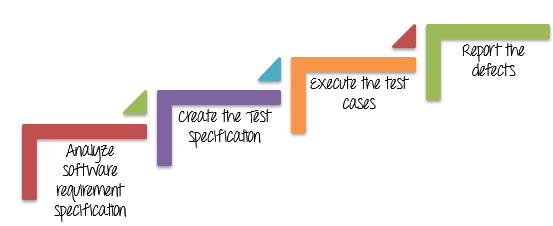
### Step1) Divide the whole project task into subtasks

Task is a piece of work that has been given to someone. To do this, you can use the **Work Breakdown Structure** technique.

In this technique, a complex project is divided into modules. The modules are divided into sub-modules. Each sub-module is further divided into functionality. It means divide the whole project task into the **smallest** tasks.

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_2_5.png)

Use the Work Break Down structure to break out the Guru99 Bank project into 5 smaller tasks-

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_2_6.png)

After that, you can break out each task to the **subtask.**The purpose of this activity is create task as **detailed** as **possible**.

|  |  |
| --- | --- |
| **Task** | **Sub task** |
| Analyze software requirement specification | Investigate the soft requirement specs |
| Interview with the developer & other stakeholders  to know more about the website |
| Create the Test Specification | Design test scenarios |
| Create test cases |
| Review and revise test cases |
| Execute the test cases | Build up the test environment |
| Execute the test cases |
| Review test execution results |
| Report the defects |  |
| Create the[Defect](https://www.guru99.com/defect-management-process.html)reports |
| Report the defects |

### Step 2) Allocate each task to team member

In this step, each task is assigned to the **appropriate** member in the project team. You can assigned task as follows

|  |  |
| --- | --- |
| **Task** | **Members** |
| Analyze software requirement specification | All the members |
| Create the test specification | Tester/Test Analyst |
| Build up the test environment | Test Administrator |
| Execute the test cases | Tester, Test Administrator |
| Report defects | Tester |

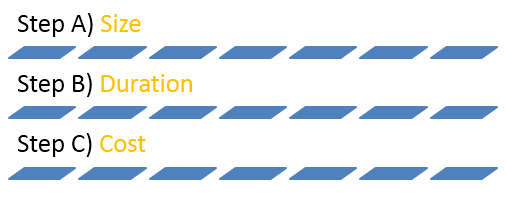
### Step 3) Effort Estimation For Tasks

There are 2 techniques which you can apply to estimate the effort for tasks

1. **Functional Point Method**
2. **Three Point Estimation**

### Method 1) Function Point Method

In this method, the Test Manager estimates Size, Duration, and Cost for the tasks

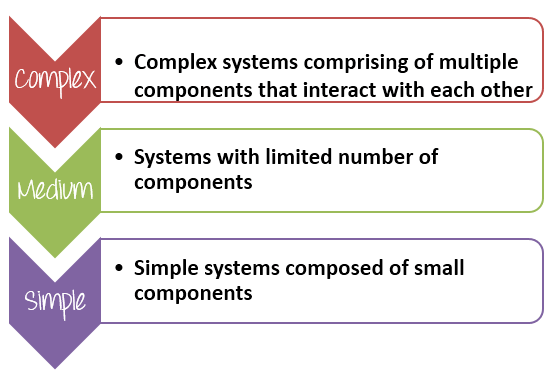
[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_2_7.png)

**Step A) Estimate size for the task**

In [Step 1](https://www.guru99.com/an-expert-view-on-test-estimation.html#Step1), you already have broken the whole project task into small task by using WBS method. Now you estimate the size of those tasks. Let’s practice with a particular task “**Create the test specification**”

The size of this task depends on the functional size of the system under test. The functional size reflects the **amount** of functionality that is relevant to the user. The more **number** of functionality, the more **complex** system is.

Prior to start actual estimating tasks effort, functional points are divided into three groups like **Complex**, **Medium** **Simple** as following:

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_2_8.png)

Based on the complex of software functions, the Test Manger has to give enough **weightage** to each functional point. For example

|  |  |
| --- | --- |
| **Group** | **Weightage** |
| Complex | 5 |
| Medium | 3 |
| Simple | 1 |

Let’s take a simple example exercise to get clearer:

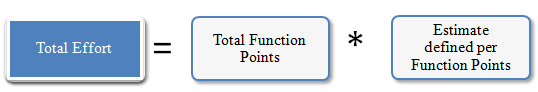
Take a look the software specification of website Guru99 Bank over [here](https://docs.google.com/document/d/1PZQZKt7hqS417QjYRMppPnTwfj8V54XUA7nZUnYvumE/edit), the software engineer have already described the software modules in detail, can you determine the **complexity** of website’s features by giving the weightage for each modules?

More complex the function point, more is the effort to test it is. The website is divided into **12 function** points, you can determine the **complexity** of each function points as follows-

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **No.** | **Module Name** | **Applicable Roles** | **Description** | **Weightage** |
| 1. | Balance Enquiry | Manager Customer | **Customer:**A customer can have multiple bank accounts. He can view balance of his accounts only **Manager:**A manager can view balance of all the customers who come under his supervision | 3 |
| 2. | Fund Transfer | Manager Customer | **Customer:**A customer can have transfer funds from his “own” account to any destination account. **Manager:**A manager can transfer funds from any source bank account to destination account | 5 |
| 3. | Mini Statement | Manager Customer | A Mini statement will show last 5 transactions of an account **Customer:**A customer can see mini-statement of only his “own” accounts **Manager:**A manager can see mini-statement of any account | 3 |
| 4. | Customized Statement | Manager Customer | A customized statement allows you to filter and display transactions in an account based on date, transaction value **Customer:**A customer can see Customized- statement of only his “own” accounts **Manager:**A manager can see Customized -statement of any account | 5 |
| 5. | Change Password | Manager Customer | **Customer:**A customer can change password of only his account. **Manager:**A manager can change password of only his account. He cannot change passwords of his customers | 1 |
| 6. | New Customer | Manager | **Manager:**A manager can add a new customer. **Manager:**A manager can edit details like address, email, telephone of a customer. | 3 |
| 7. | New Account | Manager | Currently system provides 2 types of accounts   * Saving * Current   A customer can have multiple saving accounts (one in his name, other in a joint name etc). He can have multiple current accounts for different companies he owns. Or he can have a multiple current and saving accounts. **Manager:**A manager can add a new account for an existing customer. | 5 |
| 8. | Edit Account | Manager | **Manager:**A manager can add an edit account details for an existing account | 1 |
| 9. | Delete Account | Manager | **Manager:**A manager can add a delete an account for a customer. | 1 |
| 10. | Delete Customer | Manager | A customer can be deleted only if he/she has  no active current or saving accounts **Manager:**A manager can delete a customer. | 1 |
| 11. | Deposit | Manager | **Manager:**A manager can deposit money into any account. Usually done when cash is deposited at a bank branch. | 3 |
| 12. | Withdrawal | Manager | **Manager:**A manager can withdraw money from any account. Usually done when cash is withdrawn at a bank branch. | 3 |

**STEP B) Estimate duration for the task**

After classifying the **complexity** of the function points, you have to estimate the **duration** to test them. Duration means **how much** time needs to finish the task.

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_2_9.png)

* **Total Effort**: The effort to completely test all the functions of the website
* **Total Function Points**: Total modules of the website
* **Estimate defined per Function Points**: The average effort to complete one function points. This value depends on the **productivity** of the member who will take in charge this task.

Suppose your project team has estimated defined per Function Points of **5 hours/points**. You can estimate the total effort to test all the features of website Guru99 Bank as follows:

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Weightage** | **# of Function Points** | **Total** |
| Complex | 5 | 3 | 15 |
| Medium | 3 | 5 | 15 |
| Simple | 1 | 4 | 4 |
| **Function Total Points** | | | **34** |
| **Estimate define per point** | | | **5** |
| **Total Estimated Effort (Person Hours)** | | | **170** |

So the total effort to complete the task “Create the test specification” of Guru99 Bank is around 170 man-hours

Once you understand the effort that is required, you can assign resources to determine how long the task will take (duration), and then you can estimate labor and non-labor costs.

Above example also shows the importance of the member in your team. If you have **talented** and **experienced** members, you can finish the assigned task in the **small** time, and your project will finish at the deadline or sooner.

**STEP C) Estimate the cost for the tasks**

This step helps you to answer the last question of customer “**How much does it cost?”**

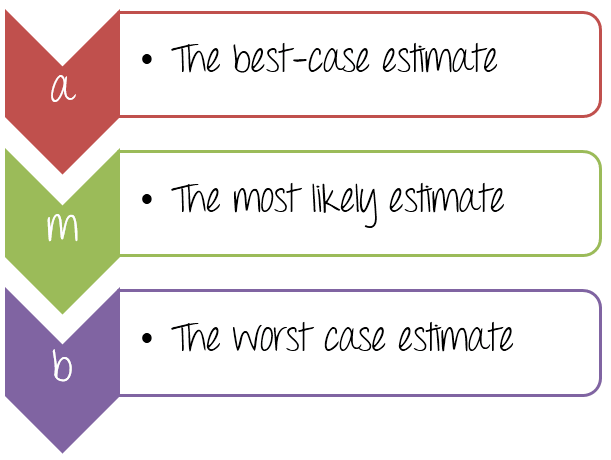
Suppose, on average your team salary is $5 per hour. The time required for “Create Test Specs” task is 170 hours. Accordingly, the cost for the task is 5\*170= $850. Now you can calculate budget for other activities in WBS and arrive at overall budget for the project.

As a project manager, you have to decide how to get the **most return** for your company’s investment. The more **accurate** your estimate of project cost is, the **better** able you will be to manage your project’s budget.

### METHOD 2) Three Point Estimation

Three-Point estimation is one of the techniques that could be used to estimate a task. The simplicity of the Three-point estimation makes it a very useful tool for a Project Manager that who wants to estimate.

In three-point estimation, **three** values are produced initially for every task based on **prior experience** or **best-guesses** as follows

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_2_10.png)

When estimating a task, the Test Manager needs to provide three values, as specified above. The three values identified, estimate what happens in an **optimal state**, what is the **most likely**, or what we think it would be the **worst case** scenario.

Let’s see how to use the above three values in the following example

For the task “**Create the test specification**”, can you estimate the test effort? Remember that you have to **cover all**the modules of the Guru99 Bank website as done in [Function Point Method](https://www.guru99.com/an-expert-view-on-test-estimation.html#8)

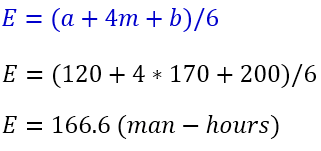
You can estimate as following

* The **best case** to complete this task is **120** man-hours (around 15 days). In this case, you have a talented team, they can finish the task in smallest time.
* The **most likely** case to complete this task is **170** man-hours (around 21 days). This is a normal case, you have enough resource and ability to complete the task
* The **worst case** to complete this task is **200** man-hours (around 25 days). You need to perform much more work because your team members are not experienced.

Now, assign the value to each parameter as below

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_2_11.png)

The effort to complete the task can be calculated using **double-triangular distribution** formula as follows-

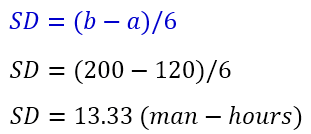
[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_2_12.png)

In the above formula, parameter E is known as **Weighted Average. It is the estimation of the task “Create the test specification”.**

But your boss may ask you

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_2_13.jpg)

In the above estimation, you just determine a **possible** and not a **certain** value, we must know about the **probability** that the estimation is correct. You can use the other formula:

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_2_14.png)

In above formula, the SD mean Standard Deviation, this value could give you the information about the **probability** that the estimation is correct.

Now you can conclude the estimation for the task “Create the test specification”

To complete the task “Create the test specification” of Guru99 Bank website, you need **166.6 ± 13.33** Man-hour (153.33 to 179.99 man-hour)

### Step 4) Validate the estimation

Once you create an aggregate estimate for all the tasks mentioned in the WBS, you need to forward it to the **management board**, who will **review** and **approve** it.

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_2_15.jpg)

The member of management board could comprise of the CEO, Project Manager & other stakeholders.

The management board will review and discuss your estimation plan with you. You may explain them your estimation **logically** and **reasonably**so that they can approve your estimation plan.

## Test estimation best practices

This topic introduces general tips on how to estimate Testing accuracy.

* **Add some buffer time:** Many unpredictable things may happen to your project, such as a talented team member quits his job suddenly, the testing takes more time than estimated to complete… etc. That why you need include some buffer in your estimation. Having a buffer in the estimation enables to cope for any delays that may occur.
* **Account Resource planning in estimation:** What should you do if some members in your team take long leaves? It may delay the project.Resource planning in estimation plays a key role. The availability of resources will help to make sure that the estimations are realistic. Here you have to consider the leaves for your team member, generally long leaves.
* **Use the past experience as reference:**Experiences from past projects play a vital role while preparing the time estimates. Because some project may be some similarity, you can reuse the past estimation. For example, if you use to do a project like testing a website, you can learn from that experience, try to avoid all the difficulties or issues that were faced in past projects.
* **Stick to your estimation:**Estimation is just estimate because it may go **wrong**.In early stages of the project, you should frequently **re-check the test estimations and make modification**if needed. We should not extend the estimation after we fix it, unless there are major changes in requirement, or you have to negotiate with customer about the re-estimation

### Software Test Estimation Template

[**Download the Software Test Estimation Excel(.xlsx)**](https://drive.google.com/uc?export=download&id=1VIAzFdKXMWFckFB9CDKUHkappXAXQ3fi)

## Other Techniques

Wideband Delphi Technique, Use – Case Point Method, Percentage distribution, Ad-hoc method are other estimation techniques in Software Engineering.

## What is a Test Plan?

A **TEST PLAN** is a detailed document that describes the test strategy, objectives, schedule, estimation and deliverables and resources required for testing. Test Plan helps us determine the effort needed to validate the quality of the application under test. The test plan serves as a blueprint to conduct software testing activities as a defined process which is minutely monitored and controlled by the test manager.

Let’s start with following scenario

In a meeting, you want to discuss the Test Plan with the team members, but they are not interested - .

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_4_1.jpg)

In such case, what will you do? Select your answer as following figure

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_4_2.jpg)

 A) I am Manager do everything as I said  
  
 B) OK, let's me explain why we need a Test Plan

## Importance of Test Plan

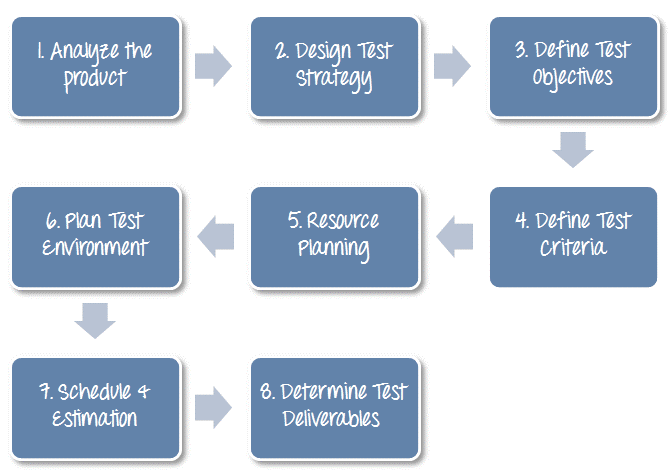
Making Test Plan has multiple benefits

* Help people outside the test team such as developers, business managers, customers **understand** the details of testing.
* Test Plan **guides** our thinking. It is like a rule book, which needs to be followed.
* Important aspects like test estimation, test scope,[Test Strategy](https://www.guru99.com/how-to-create-test-strategy-document.html)are **documented** in Test Plan, so it can be reviewed by Management Team and re-used for other projects.

## How to write a Test Plan

You already know that making a **Test Plan** is the most important task of Test Management Process. Follow the seven steps below to create a test plan as per IEEE 829

1. Analyze the product
2. Design the Test Strategy
3. Define the Test Objectives
4. Define Test Criteria
5. Resource Planning
6. Plan Test Environment
7. Schedule & Estimation
8. Determine Test Deliverables

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_4_3.png)

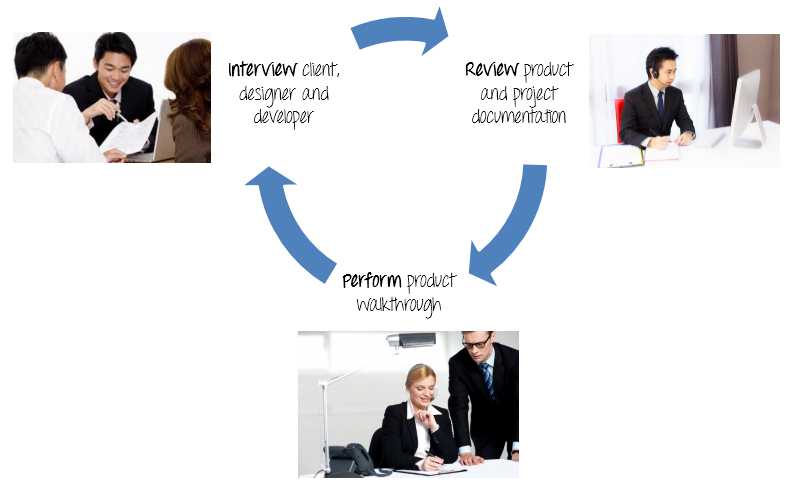
### Step 1) Analyze the product

How can you test a product **without** any information about it? The answer is **Impossible.**You must learn a product **thoroughly**before testing it.

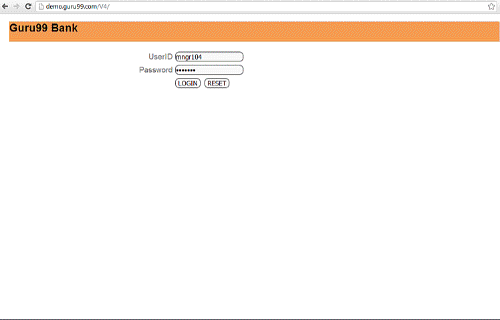
The product under test is Guru99 banking website. You should research clients and the end users to know their needs and expectations from the application

* Who will use the website?
* What is it used for?
* How will it work?
* What are software/ hardware the product uses?

You can use the following approach to analyze the site

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_4_4.jpg)

Now let’s apply above knowledge to a real product: **Analyze** the banking website <http://demo.guru99.com/V4>.

[](https://www.guru99.com/images/TestManagement/Guru99TourFrame.gif)

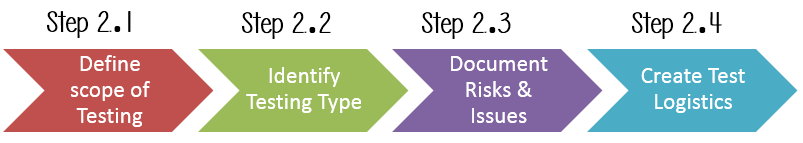
You should take a **look around** this website and also **review**[product documentation](https://docs.google.com/document/d/1PZQZKt7hqS417QjYRMppPnTwfj8V54XUA7nZUnYvumE/edit?usp=sharing). Review of product documentation helps you to understand all the features of the website as well as how to use it. If you are unclear on any items, you might **interview** customer, developer, designer to get more information.

### Step 2) Develop Test Strategy

Test Strategy is a **critical step**in making a Test Plan. A Test Strategy document, is a high-level document, which is usually developed by Test Manager. This document defines:

* The project’s **testing objectives** and the means to achieve them
* Determines testing **effort** and **costs**

Back to your project, you need to develop Test Strategy for testing that banking website. You should follow steps below

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_4_6.png)

### Step 2.1) Define Scope of Testing

Before the start of any test activity, scope of the testing should be known. You must think hard about it.

* The components of the system to be tested (hardware, software, middleware, etc.) are defined as "**in scope**"
* The components of the system that will not be tested also need to be clearly defined as being "**out of scope**."

Defining the scope of your testing project is very important for all stakeholders. A precise scope helps you

* Give everyone a **confidence & accurate information** of the testing you  are doing
* All project members will have a **clear** understanding about what is tested and what is not

**How do you determine scope your project?**

To determine scope, you must –

* Precise customer requirement
* Project Budget
* Product Specification
* Skills & talent of your test team

Now should clearly define the "in scope" and "out of scope" of the testing.

* As the software requirement [specs](https://docs.google.com/document/d/1rPW5DV82VJT6vtA1VDSrfxaCBuAduxW0zb1yfTh_VMk/edit?pli=1#heading=h.ftgetk7f23qj), the project Guru99 Bank only focus on testing all the **functions** and external interface of website **Guru99** Bank (**in scope** testing)
* Nonfunctional testing such as **stress**, **performance** or **logical database** currently will not be tested. (**out of** scope)

**Problem Scenario**

The customer wants you to test his API. But the project budget does not permit to do so. In such a case what will you do?

Well, in such case you need to convince the customer that [Api Testing](https://www.guru99.com/api-testing.html) is extra work and will consume significant resources. Give him data supporting your facts. Tell him if Api Testing is included in-scope the budget will increase by XYZ amount.

The customer agrees and accordingly the new scopes, out of scope items are

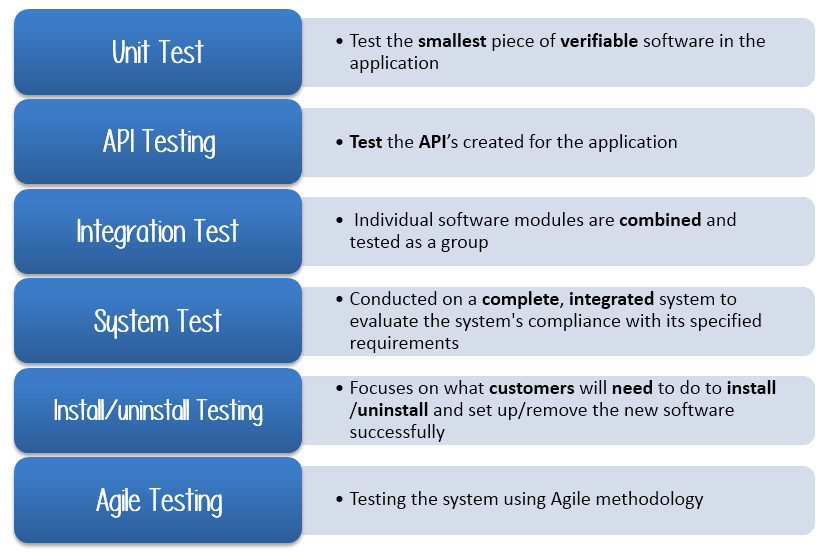
* In-scope items:[Functional Testing](https://www.guru99.com/functional-testing.html), Api Testing
* Out of scope items: [Database Testing](https://www.guru99.com/data-testing.html), hardware & any other external interfaces

### Step 2.2) Identify Testing Type

A **Testing Type** is a standard test procedure that gives an expected test outcome.

Each testing type is formulated to identify a specific type of product bugs. But, all Testing Types are aimed at achieving one common goal “**Early detection of** all the defects before releasing the product to the customer”

The **commonly used** testing types are described as following figure

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_4_7.png)Commonly Used Testing Types

There are **tons of Testing Types** for testing software product. Your team **cannot have** enough efforts to handle all kind of testing. As Test Manager, you must set **priority** of the Testing Types

* Which Testing Types should be **focused** for web application testing?
* Which Testing Types should be **ignored** for saving cost?

**Now let's practice with your project. The product you want to test is a banking website.  
  
Which Testing Types should you focus in this case?  
  
Select All that Apply**

 A) Unit Testing  
  
 B) API Testing  
  
 C) Integration Testing  
  
 D) System Testing  
  
 E) Install/Uninstall Testing  
  
 F) Agile testing

### Step 2.3) Document Risk & Issues

Risk is future’s **uncertain event** with a probability of **occurrence** and a **potential** for loss. When the risk actually happens, it becomes the ‘**issue’.**

In the article [Risk Analysis and Solution](https://www.guru99.com/how-precaution-becomes-cure-risk-analysis-and-solutions-in-test-management.html), you have already learned about the ‘Risk’ analysis in detail and identified potential risks in the project.

In the QA Test Plan, you will document those risks

|  |  |
| --- | --- |
| **Risk** | **Mitigation** |
| Team member lack the required skills for website testing. | Plan **training course** to skill up your members |
| The project schedule is too tight; it's hard to complete this project on time | Set **Test Priority** for each of the test activity. |
| Test Manager has poor management skill | Plan **leadership training** for manager |
| A lack of cooperation negatively affects your employees' productivity | **Encourage**each team member in his task, **and inspire** them to greater efforts. |
| Wrong budget estimate and cost overruns | Establish the **scope** before beginning work, pay a lot of attention to project planning and constantly track and measure the progress |

### Step 2.4) Create Test Logistics

 In Test Logistics, the Test Manager should answer the following questions:

* **Who**will test?
* **When**will the test occur?

**Who will test?**

You may not know exact names of the tester who will test, but the **type of tester** can be defined.

To select the right member for specified task, you have to consider if his skill is qualified for the task or not, also estimate the project budget. Selecting wrong member for the task may cause the project to**fail** or **delay**.

Person having the following skills is most ideal for performing software testing:

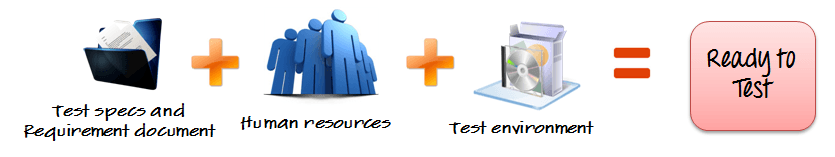
* Ability to **understand** customers point of view
* Strong **desire** for quality
* **Attention** to detail
* Good **cooperation**

In your project, the member who will take in charge for the test execution is the **tester.** Base on the project budget, you can choose in-source or outsource member as the tester.

**When will the test occur?**

Test activities must be matched with associated development activities.

You will start to test when you have **all required items** shown in following figure

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_4_8.png)

### Step 3) Define Test Objective

Test Objective is the overall goal and achievement of the test execution. The objective of the testing is finding as many software defects as possible; ensure that the software under test is **bug free** before release.

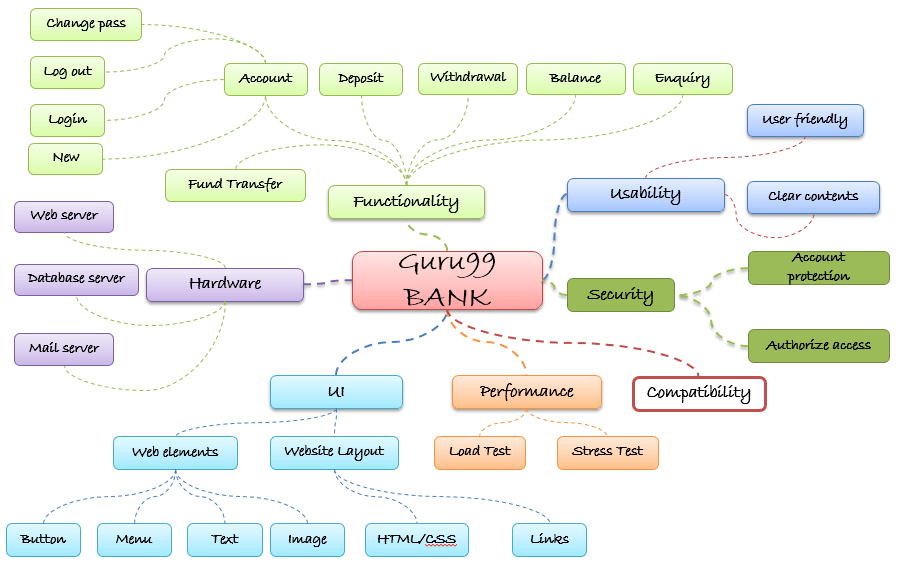
To define the test objectives, you should do 2 following steps

1. List all the software features (functionality, performance, GUI…) which may need to test.
2. Define the **target** or the **goal** of  the test based on  above features

Let’s apply these steps to find the test objective of your Guru99 Bank testing project

You can choose the ‘**TOP-DOWN’**method to find the website’s features which may need to test. In this method, you break down the application under test to **component** and **sub-component**.

In the previous topic, you have already analyzed the requirement specs and walk through the website, so you can create a **Mind-Map** to find the website features as following

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_4_9.png)

This figure shows all the features which the Guru99 website may have.

Based on above features, you can define the Test Objective of the project Guru99 as following

* Check that whether website Guru99 **functionality**(Account, Deposit…) is working as expected without any error or bugs in real business environment
* Check that the external interface of the website such as **UI** is working as expected and & meet the customer need
* Verify the **usability** of the website. Are those functionalities convenient for user or not?

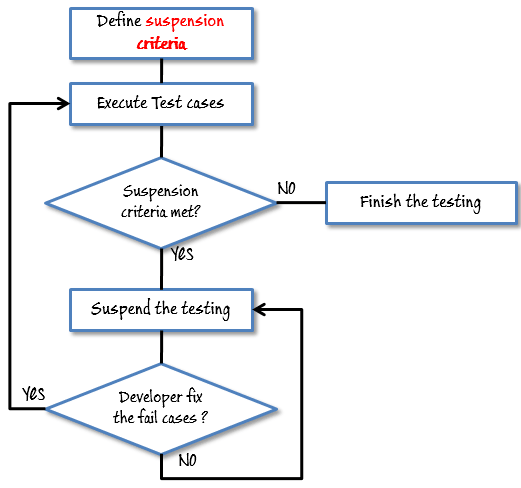
### Step 4) Define Test Criteria

Test Criteria is a standard or rule on which a test procedure or test judgment can be based. There’re 2 types of test criteria as following

### Suspension Criteria

Specify the critical suspension criteria for a test. If the suspension criteria are met during testing, the active test cycle will be **suspended** until the criteria are **resolved**.

Example: If your team members report that there are **40%** of test cases failed, you should **suspend** testing until the development team fixes all the failed cases.

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_4_10.png)

### Exit Criteria

It specifies the criteria that denote a **successful** completion of a test phase. The exit criteria are the targeted results of the test and are necessary before proceeding to the next phase of development. Example: **95%** of all critical test cases must pass.

Some methods of defining exit criteria are by specifying a targeted **run rate** and **pass rate**.

* Run rate is ratio between **number test cases executed/total test cases** of test specification. For example, the test specification has total 120 TCs, but the tester only executed 100 TCs, So the run rate is 100/120 = 0.83 (83%)
* Pass rate is ratio between **numbers test cases passed / test cases executed**. For example, in above 100 TCs executed, there’re 80 TCs that passed, so the pass rate is 80/100 = 0.8 (80%)

This data can be retrieved in Test Metric documents.

* **Run** rate is mandatory to be **100%**unless a clear reason is given.
* **Pass** rate is dependent on project scope, but **achieving high pass rate** is a goal.

**Example:**Your Team has already done the test executions. They report the test result to you, and they want you to confirm the **Exit Criteria.**

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_4_11.jpg)

In above case, the Run rate is mandatory is **100%,**but the test team only completed 90% of test cases. It means the Run rate is not satisfied, so do NOT confirm the Exit Criteria

### Step 5) Resource Planning

Resource plan is a **detailed summary** of all types of resources required to complete project task. Resource could be human, equipment and materials needed to complete a project

The resource planning is important factor of the test planning because helps in **determining** the **number** of resources (employee, equipment…) to be used for the project. Therefore, the Test Manager can make the correct schedule & estimation for the project.

This section represents the recommended resources for your project.

### Human Resource

The following table represents various members in your project team

|  |  |  |
| --- | --- | --- |
| **No.** | **Member** | **Tasks** |
| **1.** | Test Manager | **Manage** the whole project  Define project **directions**  Acquire appropriate resources |
| **2.** | Tester | Identifying and describing appropriate test techniques/tools/automation architecture  Verify and assess the Test Approach  **Execute** the tests, **Log** results, **Report** the defects.  Tester could be in-sourced or out-sourced members, base on the project budget  For the task which required **low** skill, I recommend you choose **outsourced** members to **save** project cost. |
| **3.** | Developer in Test | **Implement**the test cases, test program, test suite etc. |
| **4.** | Test Administrator | Builds up and ensures[Test Environment](https://www.guru99.com/test-environment-software-testing.html)and assets are **managed** and **maintained**  **Support**Tester to use the test environment for test execution |
| **5.** | SQA members | Take in charge of quality assurance  Check  to confirm whether the testing process is meeting specified requirements |

### System Resource

For testing, a web application, you should plan the resources as following tables:

|  |  |  |
| --- | --- | --- |
| **No.** | **Resources** | **Descriptions** |
| **1.** | Server | Install the web application under test  This includes a separate web server, database server, and application server if applicable |
| **2.** | Test tool | The testing tool is to automate the testing, simulate the user operation, generate the test results  There are tons of test tools you can use for this project such as Selenium, QTP…etc. |
| **3.** | Network | You need a Network include LAN and Internet to simulate the real business and user environment |
| **4.** | Computer | The PC which users often use to connect the web server |

### Step 6) Plan Test Environment

### What is the Test Environment

A testing environment is a setup of software and hardware on which the testing team is going to execute test cases. The test environment consists of **real business** and **user** environment, as well as physical environments, such as server, front end running environment.

### How to setup the Test Environment

Back to your project, how do you set up **test environment** for this banking website?

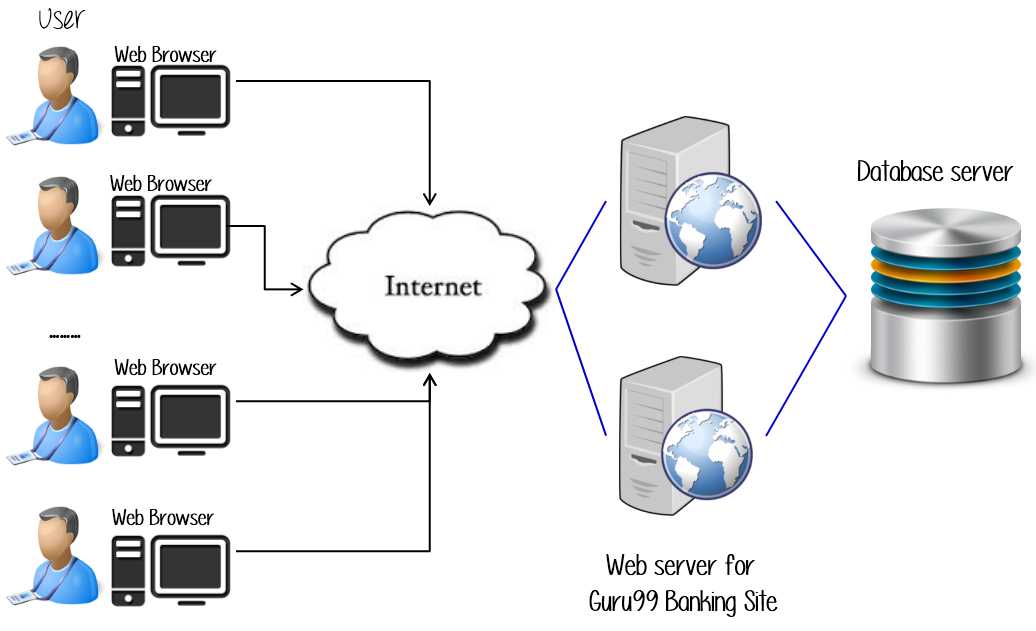
To finish this task, you need **a strong cooperation** between Test Team and Development Team

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_4_12.png)

You should ask the developer some questions to understand the web application under test **clearly**. Here’re some recommended questions. Of course, you can ask the other questions if you need.

* What is the maximum user connection which this website can handle at the same time?
* What are hardware/software requirements to install this website?
* Does the user's computer need any particular setting to browse the website?

Following figure describes the test environment of the banking website [www.demo.guru99.com/V4](http://www.demo.guru99.com/V4)

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_4_13.jpg)

### Step 7) Schedule & Estimation

In the article [Test estimation](https://www.guru99.com/an-expert-view-on-test-estimation.html), you already used some techniques to estimate the effort to complete the project. Now you should include that estimation as well as the schedule to the Test Planning

In the Test Estimation phase, suppose you break out the whole project into small tasks and add the estimation for each task as below

|  |  |  |
| --- | --- | --- |
| **Task** | **Members** | **Estimate effort** |
| **Create the test specification** | Test Designer | 170 man-hour |
| **Perform Test Execution** | Tester, Test Administrator | 80 man-hour |
| **Test Report** | Tester | 10 man-hour |
| **Test Delivery** |  | 20 man-hour |
| **Total** |  | **280 man-hour** |

Then you create the **schedule** to complete these tasks.

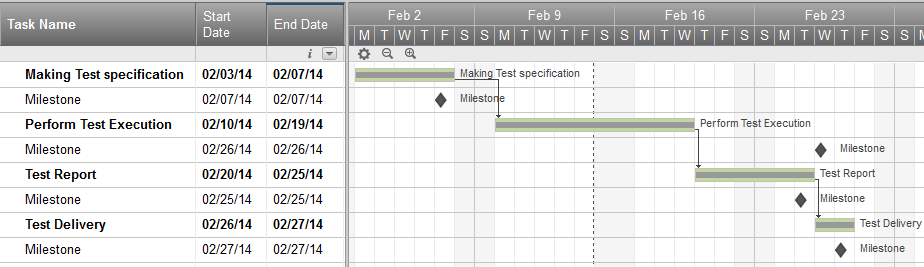
Making schedule is a common term in project management. By creating a solid schedule in the Test Planning, the Test Manager can use it as tool for monitoring the project progress, control the cost overruns.

To create the project schedule, the Test Manager needs several types of input as below:

* **Employee and project deadline**: The working days, the project deadline, resource availability are the factors which affected to the schedule
* **Project estimation**:  Base on the estimation, the Test Manager knows how long it takes to complete the project. So he can make the appropriate project schedule
* **Project Risk**: Understanding the risk helps Test Manager add enough extra time to the project schedule to deal with the risks

Let’s practice with an example:

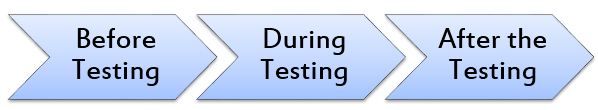
Suppose the boss wants to complete the project Guru99 in **one** month, you already estimated the effort for each tasks in Test Estimation. You can create the schedule as below

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_4_14.png)

### Step 8) Test Deliverables

Test Deliverables is a list of all the documents, tools and other components that has to be developed and maintained in support of the testing effort.

There are different test deliverables at every phase of the software development lifecycle.

[](https://www.guru99.com/images/TestManagement/testmanagement_article_2_4_15.png)

Test deliverables are provided **before** testing phase.

* Test plans document.
* Test cases documents
* Test Design specifications.

Test deliverables are provided **during** the testing

* Test Scripts
* Simulators.
* Test Data
* Test Traceability Matrix
* Error logs and execution logs.

Test deliverables are provided **after** the testing cycles is over.

* **Test Results/reports**
* Defect Report
* Installation/ Test procedures guidelines
* **Release notes**

## What is test plan template?

**TEST PLAN TEMPLATE** is a detailed document that describes the test strategy, objectives, schedule, estimation and deliverables, and resources required for testing. Test Plan helps us determine the effort needed to validate the quality of the application under test. The test plan serves as a blueprint to conduct software testing activities as a defined process which is minutely monitored and controlled by the test manager.

Creating a[Test Plan](https://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html)is mandatory to ensure success of your Software testing project.If you are new to Test Planning refer this tutorial on [How to Create a Test Plan](https://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html)

[**Download Sample Test Plan Template**](https://drive.google.com/uc?export=download&id=0ByI5-ZLwpo25LThJNUZzUzdkQXM)

Below find important constituents of a test plan-

* [1 Introduction](https://www.guru99.com/test-plan-for-project.html#1)
* [1.1 Scope](https://www.guru99.com/test-plan-for-project.html#11)
* [1.1.1 In Scope](https://www.guru99.com/test-plan-for-project.html#111)
* [1.1.2 Out of Scope](https://www.guru99.com/test-plan-for-project.html#112)
* [1.2 Quality Objective](https://www.guru99.com/test-plan-for-project.html#12)
* [1.3 Roles and Responsibilities](https://www.guru99.com/test-plan-for-project.html#13)
* [2 Test Methodology](https://www.guru99.com/test-plan-for-project.html#2)
* [2.1 Overview](https://www.guru99.com/test-plan-for-project.html#21)
* [2.2 Test Levels](https://www.guru99.com/test-plan-for-project.html#22)
* [2.3 Bug Triage](https://www.guru99.com/test-plan-for-project.html#23)
* [2.4 Suspension Criteria and Resumption Requirements](https://www.guru99.com/test-plan-for-project.html#24)
* [2.5 Test Completeness](https://www.guru99.com/test-plan-for-project.html#25)
* [3 Test Deliverables](https://www.guru99.com/test-plan-for-project.html#3)
* [4 Resource & Environment Needs](https://www.guru99.com/test-plan-for-project.html#4)
* [4.1 Testing Tools](https://www.guru99.com/test-plan-for-project.html#41)
* [4.2 Test Environment](https://www.guru99.com/test-plan-for-project.html#42)

## 1) Introduction

Brief introduction of the test strategies, process, workflow and methodologies used for the project

## 1.1) Scope

**1.1.1) In Scope**

Scope defines the features, functional or non-functional requirements of the software that **will be** tested

**1.1.2) Out of Scope**

Out Of Scope defines the features, functional or non-functional requirements of the software that **will NOT be** tested

## 1.2) Quality Objective

Here make a mention of the overall objective that you plan to achive with your manual testing and automation testing.

Some objectives of your testing project could be

* Ensure the Application Under Test conforms to functional and non-functional requirements
* Ensure the AUT meets the quality specifications defined by the client
* Bugs/issues are identified and fixed before go live

## 1.3) Roles and Responsibilities

Detail description of the Roles and responsibilities of different team members like

* QA Analyst
* Test Manager
* Configuration Manager
* Developers
* Installation Team

Amongst others

## 2) Test Methodology

### 2.1) Overview

Mention the reason of adopting a particular test methodology for the project. The test methodology selected for the project could be

* WaterFall
* Iterative
* Agile
* Extreme Programming

The methodology selected depends on multiple factors. You can read about Test Methodology [here](https://www.guru99.com/testing-methodology.html)

### 2.2) Test Levels

**Test Levels define the Types of Testing to be executed on the Application Under Test (AUT**). The Testing Levels primarily depends on the scope of the project, time and budget constraints.

### 2.3) Bug Triage

The goal of the triage is to

* To define the type of resolution for each bug
* To prioritize bugs and determine a schedule for all "To Be Fixed Bugs'.

### 2.4) Suspension Criteria and Resumption Requirements

Suspension criteria define the criteria to be used to suspend all or part of the testing procedure while Resumption criteria determine when testing can resume after it has been suspended

### 2.5) Test Completeness

Here you define the criterias that will deem your testing complete.

For instance, a few criteria to check Test Completeness would be

* 100% test coverage
* All Manual & Automated Test cases executed
* All open bugs are fixed or will be fixed in next release

## 3) Test Deliverables

Here mention all the Test Artifacts that will be delivered during different phases of the testing lifecycle.

Here are the simple deliverables

|  |
| --- |
| * Test Plan * Test Cases * Requirement Traceability Matrix * Bug Reports * Test Strategy * Test Metrics * Customer Sign Off |

## 4) Resource & Environment Needs

### 4.1) Testing Tools

Make a list of Tools like

* Requirements Tracking Tool
* Bug Tracking Tool
* Automation Tools

Required to test the project

### 4.2) Test Environment

It mentions the minimum **hardware** requirements that will be used to test the Application.

Following **software's** are required in addition to client-specific software.

* Windows 8 and above
* Office 2013 and above
* MS Exchange, etc.

## 5) Terms/Acronyms

Make a mention of any terms or acronyms used in the project

|  |  |
| --- | --- |
| **TERM/ACRONYM** | **DEFINITION** |
| API | Application Program Interface |
| AUT | Application Under Test |

[**Download the above Test Plan Template Format**](https://drive.google.com/uc?export=download&id=0ByI5-ZLwpo25LThJNUZzUzdkQXM)

## Sample Test Plan Document Banking Web Application Example

### 1 Introduction

The Test Plan is designed to prescribe the scope, approach, resources, and schedule of all testing activities of the project Guru99 Bank.

The plan identify the items to be tested, the features to be tested, the types of testing to be performed, the personnel responsible for testing, the resources and schedule required to complete testing, and the risks associated with the plan.

#### 1.1 Scope

#### 1.1.1 In Scope

All the feature of websiteGuru99 Bank which were defined in software requirement [specs](https://docs.google.com/document/d/1rPW5DV82VJT6vtA1VDSrfxaCBuAduxW0zb1yfTh_VMk/edit?pli=1) are need to bested

|  |  |  |
| --- | --- | --- |
| Module Name | Applicable Roles | Description |
| Balance Enquiry | Manager Customer | **Customer**: A customer can have multiple bank accounts. He can view balance of his accounts only **Manager**: A manager can view balance of all the customers who come under his supervision |
| Fund Transfer | Manager Customer | **Customer:** A customer can have transfer funds from his “own” account to any destination account. **Manager**: A manager can transfer funds from any source bank account to destination account |
| Mini Statement | Manager Customer | A Mini statement will show last 5 transactions of an account **Customer:** A customer can see mini-statement of only his “own” accounts **Manager:** A manager can see mini-statement of any account |
| Customized Statement | Manager Customer | A customized statement allows you to filter and display transactions in an account based on date, transaction value **Customer:** A customer can see Customized- statement of only his “own” accounts **Manager**: A manager can see Customized -statement of any account |
| Change Password | Manager Customer | **Customer:**A customer can change password of only his account. **Manager**: A manager can change password of only his account. He cannot change passwords of his customers |
| New Customer | Manager | **Manager**: A manager can add a new customer. |
|  | Manager | **Manager:** A manager can edit details like address, email, telephone of a customer. |
| New Account | Manager | Currently system provides 2 types of accounts • Saving • Current A customer can have multiple saving accounts (one in his name, other in a joint name etc). He can have multiple current accounts for different companies he owns. Or he can have a multiple current and saving accounts. **Manager:** A manager can add a new account for an existing customer. |
| Edit Account | Manager | Manager: A manager can add a edit account details for an existing account |
| Delete Account | Manager | Manager: A manager can add a delete an account for a customer. |
| Delete Customer | Manager | A customer can be deleted only if he/she has no active current or saving accounts **Manager:** A manager can delete a customer. |
| Deposit | Manager | **Manager:** A manager can deposit money into any account. Usually done when cash is deposited at a bank branch. |
| Withdrawal | Manager | **Manager:** A manager can withdraw money from any account. Usually done when cash is withdrawn at a bank branch. |

#### 1.1.2 Out of Scope

These feature are not be tested because they are not included in the software requirement specs

* User Interfaces
* Hardware Interfaces
* Software Interfaces
* Database logical
* Communications Interfaces
* Website Security and Performance

### 1.2 Quality Objective

The test objectives are to **verify** the Functionality of website Guru99 Bank, the project should focus on testing the **banking operation** such as Account Management, Withdrawal, and Balance…etc. to **guarantee** all these operation can work **normally** in real business environment.

### 1.3 Roles and Responsibilities

The project should use **outsource** members as the tester to save the project cost.

|  |  |  |
| --- | --- | --- |
| **No.** | **Member** | **Tasks** |
| **1.** | Test Manager | Manage the whole project Define project directions Acquire appropriate resources |
| **2.** | Test | Identifying and describing appropriate test techniques/tools/automation architecture Verify and assess the Test Approach Execute the tests, Log results, Report the defects. Outsourced members |
| **3.** | Developer in Test | Implement the test cases, test program, test suite etc. |
| **4.** | Test Administrator | Builds up and ensures test environment and assets are managed and maintained Support Tester to use the test environment for test execution |
| **5.** | SQA members | Take in charge of quality assurance Check to confirm whether the testing process is meeting specified requirements |

### 2 Test Methodology

### 2.1 Overview

### 2.2 Test Levels

In the project Guru99 Bank, there're 3 types of testing should be conducted.

* **Integration** Testing (Individual software modules are combined and tested as a group)
* **System** Testing: Conducted on a **complete**, **integrated** system to evaluate the system's compliance with its specified requirements
* **API testing:**Test all the APIs create for the software under tested

### 2.3 Bug Triage

### 2.4 Suspension Criteria and Resumption Requirements

If the team members report that there are **40%** of test cases **failed**, suspend testing until the development team fixes all the failed cases.

### 2.5 Test Completeness

* Specifies the criteria that denote a **successful** completion of a test phase
* **Run** rate is mandatory to be **100%**unless a clear reason is given.
* **Pass** rate is **80%,**achieving the pass rate is **mandatory**

### 2.6 Project task and estimation and schedule

|  |  |  |
| --- | --- | --- |
| **Task** | **Members** | **Estimate effort** |
| **Create the test specification** | Test Designer | 170 man-hour |
| **Perform Test Execution** | Tester, Test Administrator | 80 man-hour |
| **Test Report** | Tester | 10 man-hour |
| **Test Delivery** |  | 20 man-hour |
| **Total** |  | **280 man-hour** |

**Schedule to complete these tasks**

### 3 Test Deliverables

Test deliverables are provided as below

**Before testing phase**

* Test plans document.
* Test cases documents
* Test Design specifications.

**During the testing**

- Test Tool Simulators.

- Test Data

- Test Trace-ability Matrix - Error logs and execution logs.

**After the testing cycles is over**

* Test Results/reports
* Defect Report
* Installation/ Test procedures guidelines
* Release notes

### 4 Resource & Environment Needs

### 4.1 Testing Tools

|  |  |  |
| --- | --- | --- |
| **No.** | **Resources** | **Descriptions** |
| **1.** | Server | Need a Database server which install MySQL server Web server which install Apache Server |
| **2.** | Test tool | Develop a Test tool which can auto generate the test result to the predefined form and automated test execution |
| **3.** | Network | Setup a LAN Gigabit and 1 internet line with the speed at least 5 Mb/s |
| **4.** | Computer | At least 4 computer run Windows 7, Ram 2GB, CPU 3.4GHZ |

### 4.2 Test Environment

Test Environment to be setup as per figure below

**Defects**

## What is a Test Environment?

A testing environment is a setup of software and hardware for the testing teams to execute test cases. In other words, it supports test execution with hardware, software and network configured.

Test bed or test environment is configured as per the need of the Application Under Test. On a few occasion, test bed could be the combination of the test environment and the test data it operates.

Setting up a right test environment ensures software testing success. Any flaws in this process may lead to extra cost and time to the client.

In this tutorial, you will learn-

* [What is a Test Environment](https://www.guru99.com/test-environment-software-testing.html#1)
* [Key areas to set up in Test Environment](https://www.guru99.com/test-environment-software-testing.html#2)
* [Process of Software Test environment set up](https://www.guru99.com/test-environment-software-testing.html#3)
* [Test Environment Management](https://www.guru99.com/test-environment-software-testing.html#4)
* [Test Environment Checklist](https://www.guru99.com/test-environment-software-testing.html#5)
* [Challenges in setting up Test Environment Management](https://www.guru99.com/test-environment-software-testing.html#6)
* [Best practices for setting up a Test Environment Management](https://www.guru99.com/test-environment-software-testing.html#7)

## Key areas to set up in Test Environment

For the test environment, a key area to set up includes

* System and applications
* Test data
* Database server
* Front-end running environment
* Client operating system
* Browser
* Hardware includes Server Operating system
* Network
* Documentation required like reference documents/configuration guides/installation guides/ user manuals

## Process of Software Test environment setup

Tests are limited to what can be tested and what not should be tested.

Following people are involved in test environment setup

* System Admins,
* Developers
* Testers
* Sometimes users or techies with an affinity for testing.

The test environment requires setting up of various number of distinct areas like,

**Setup of Test Server**

Every test may not be executed on a local machine. It may need establishing a test server, which can support applications.

For example, Fedora set up for PHP, Java-based applications with or without mail servers, cron set up, Java-based applications, etc.

**Network**

Network set up as per the test requirement. It includes,

* Internet setup
* LAN Wifi setup
* Private network setup

It ensures that the congestion that occurs during testing doesn't affect other members. (Developers, designers, content writers, etc.)

**Test PC setup**

For web testing, you may need to set up different browsers for different testers. For desktop applications, you need various types of OS for different testers PCs.

For example, windows phone app testing may require

* Visual Studio installation
* Windows phone emulator
* Alternatively, assigning a windows phone to the tester.

**Bug Reporting**

Bug reporting tools should be provided to testers.

**Creating Test Data for the Test Environment**

Many companies use a separate test environment to test the software product. The common approach used is to copy production data to test. This helps the tester, to detect the same issues as a live production server, without corrupting the production data.

The approach for copying production data to test data includes,

* Set up production jobs to copy the data to a common test environment
* All PII (Personally Identifiable Information) is modified along with other sensitive data. The PII is replaced with logically correct, but non-personal data.
* Remove data that is irrelevant to your test.

Testers or developers can copy this to their individual test environment. They can modify it as per their requirement.

Privacy is the main issue in copy production data. To overcome privacy issues you should look into obfuscated and anonymized test data.

For Anonymization of data two approaches can be used,

* BlackList: In this approach, all the data fields are left unchanged. Except those fields specified by the users.
* WhiteList: By default, this approach, anonymizes all data fields. Except for a list of fields which are allowed to be copied. A whitelisted field implies that it is okay to copy the data as it is and anonymization is not required.

Also, if you are using production data, you need to be smart about how to source data. Querying the database using[SQL](https://www.guru99.com/sql.html)script is an effective approach.

## Test Environment Management

Test Environment Management deals with the maintenance and upkeep of the test bed.

List of activities by the Test environment management function include,

1. Maintenance of a central repository with all the updated version of test environments.
2. Test environment management as per the test team demands.
3. As per the new requirements creating new environments
4. Monitoring of the environments
5. Updating/deleting outdated test-environments
6. Investigation of issues on the environment
7. Co-ordination till an issue resolution.

## Test Environment Checklist

|  |  |  |
| --- | --- | --- |
| **Hardware** | | |
| 1 | Check whether required equipment for testing is available? | If this is not the case, analyze the supply time! |
|  | Check whether peripheral equipment is available? | Such as scanners, special printers, handhelds, etc. |
| **Software / connections** | | |
| 2 | Are the needed applications specified? | An application such as excel, word, drawings, etc. |
|  | For the new software does the test environment exist for the organization? | Has the organization experience with use and maintenance of the software? |
| **Environmental data** | | |
| 3 | Check whether the standard test data sets are available? | With the regression test set, consider the[Defect](https://www.guru99.com/defect-management-process.html)administration to collect test data. |
|  | Do agreements with the test data owners about the test data exist? | Consider functional maintenance. |
| **Maintenance tools/processes** | | |
| 4 | Check whether a single point of contact exists for test environment maintenance? | If no, prepare a list of all possible members involved in keeping the test environment running. It should include their contact information as well. |
|  | Does the agreement reached about the readiness and quality of the test environment? | For instance, acceptance criteria, maintenance requirements, etc. Also, check whether other/extra quality attributes for environments are there in agreement. |
|  | Do all members involved in the maintenance process are known? |  |

Besides these, there are a few more questions to answer before setting up the test environment.

* Whether to develop an internal Test Environment or to outsource?
* Whether to follow an internal company standard or follow any External (IEE, ISO, etc.)?
* How long the test environment is required?
* Differences between the test and production systems and their impact on test validity must be determined.
* Can you re-use an existing setup for other projects in the company?

## Challenges in setting up Test Environment Management

1. **Proper planning on resource usage**

Ineffective planning for resource usage can affect the actual output. Also, it may lead to conflict between teams.

1. **Remote environment**

It is possible that a Test environment is located geographically apart. In such a case, the testing team has to rely on the support team for various test assets. (Software, hardware, and other issues).

1. **Elaborate setup time**

Sometimes test set up gets too elaborated in cases of [Integration Testing](https://www.guru99.com/integration-testing.html).

1. **Shared usage by teams**

If the testing environment is used by development & testing team simultaneously, test results will be corrupted.

1. **Complex test configuration**

Certain test requires complex test environment configuration. It may pose a challenge to the test team.

## Best practices for setting up a Test Environment Management

1. Understand the test requirements thoroughly and educate the test team members.
2. Connectivity should be checked before the initiation of the testing
3. Check for the required hardware and software, licenses
4. Browsers and versions
5. Planning out the Scheduled use of the test environment.
6. Automation tools and their configurations.

## What is a Test bed?

In general, a test bed is a software development environment. It allows the developers to test their modules without affecting the live production servers. The test bed is not confined to developers only but also used by testers. It is referred as a test environment as well.

**Summary**:

* A testing environment is a setup of software and hardware on which the test team will conduct the testing
* For the test environment, a key area to set up includes
  + System and applications
  + Test data
  + Database server
  + Front-end running environment, etc.
* Few challenges while setting up a test environment include,
  + Remote environment
  + Combined usage between teams
  + Elaborate setup time
  + Ineffective planning for resource usage for integration
  + Complex test configuration

### What is Bug?

A bug is the consequence/outcome of a coding fault

## What is Defect?

A defect is a variation or deviation from the original business requirements

These two terms have very thin line of differnce, In the Industry both are faults that need to be fixed and so interchangebaly used by some of the[Testing](https://www.guru99.com/software-testing.html)teams.

When a tester executes the test cases, he might come across the test result which is contradictory to expected result. This variation in the test result is referred as a **Software Defect**. These defects or variation are referred by different names in a different organization like **issues, problem, bug or incidents**.

In this tutorial, you will learn-

* [Bug Report](https://www.guru99.com/defect-management-process.html#2)
* [Defect Management Process](https://www.guru99.com/defect-management-process.html#3)
  + [Discovery](https://www.guru99.com/defect-management-process.html#4)
  + [Categorization](https://www.guru99.com/defect-management-process.html#5)
  + [Resolution](https://www.guru99.com/defect-management-process.html#6)
  + [Verification](https://www.guru99.com/defect-management-process.html#7)
  + [Closure](https://www.guru99.com/defect-management-process.html#8)
  + [Reporting](https://www.guru99.com/defect-management-process.html#9)
* [Important Defect Metrics](https://www.guru99.com/defect-management-process.html#10)

## Bug Report

While reporting the bug to developer, your Bug Report should contain the following information

* **Defect\_ID** - Unique identification number for the defect.
* **Defect Description** - Detailed description of the Defect including information about the module in which Defect was found.
* **Version** - Version of the application in which defect was found.
* **Steps** - Detailed steps along with screenshots with which the developer can reproduce the defects.
* **Date Raised** - Date when the defect is raised
* **Reference**-  where in you Provide reference to the documents like . requirements, design, architecture or maybe even screenshots of the error   to help understand the defect
* **Detected By** - Name/ID of the tester who raised the defect
* **Status** - Status of the defect , more on this later
* **Fixed by** - Name/ID of the developer who fixed it
* **Date Closed** - Date when the defect is closed
* **Severity** which describes the impact of the defect on the application
* **Priority** which is related to defect fixing urgency. Severity Priority could be High/Medium/Low based on the impact urgency at which the defect should be fixed respectively

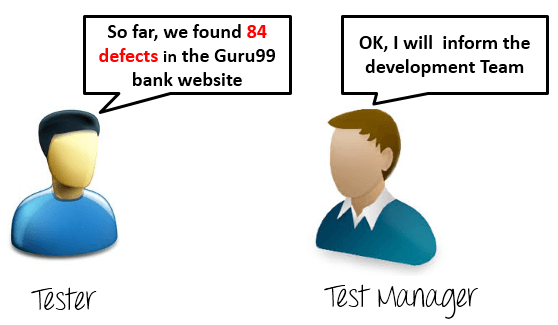
Click [here](https://www.guru99.com/faq.html#1) if the video is not accessible

Resources

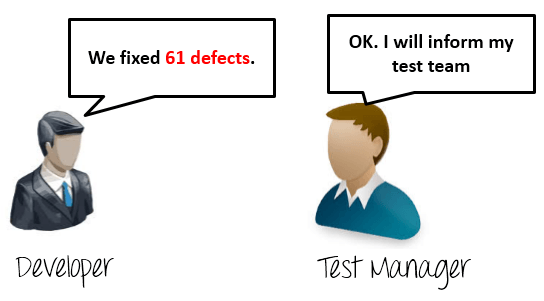
[**Download a sample Defect Reporting Template**](https://drive.google.com/uc?export=download&id=0B_vqvT0ovzHcZllqY2Q0TVpwaDQ)

## Consider the following as a Test Manager

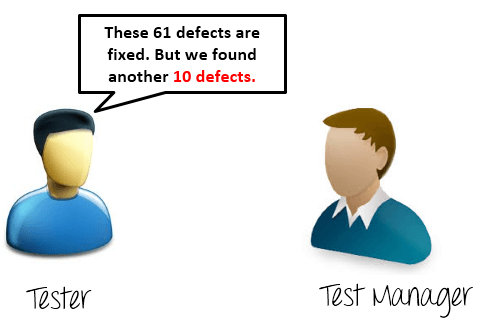
Your team found bugs while testing the Guru99 Banking project.

[](https://www.guru99.com/images/TestManagement/testmanagement_article_4_1.png)

After a week the developer responds -

[](https://www.guru99.com/images/TestManagement/testmanagement_article_4_2.png)

In next week the tester responds

[](https://www.guru99.com/images/TestManagement/testmanagement_article_4_3.png)

## What is Defect Life Cycle?

**DEFECT LIFE CYCLE** or Bug Life Cycle is the specific set of states that a Bug goes through in its entire life. The purpose of the Defect life cycle is to easily coordinate bug status changes to various assignees and make the bug fixing process systematic.

### Bug Life Cycle Status

The number of states that a defect goes through varies from project to project. Below lifecycle diagram, covers all possible states

* **New:** When a new defect is logged and posted for the first time. It is assigned a status as NEW.
* **Assigned:** Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to the developer team
* **Open**: The developer starts analyzing and works on the defect fix
* **Fixed**: When a developer makes a necessary code change and verifies the change, he or she can make bug status as "Fixed."
* **Pending retest**: Once the defect is fixed the developer gives a particular code for retesting the code to the tester. Since the software testing remains pending from the testers end, the status assigned is "pending retest."
* **Retest**: Tester does the retesting of the code at this stage to check whether the defect is fixed by the developer or not and changes the status to "Re-test."

[](https://www.guru99.com/images/1-2015/012715_0802_BugLifeCycl1.png)

* **Verified**: The tester re-tests the bug after it got fixed by the developer. If there is no bug detected in the software, then the bug is fixed and the status assigned is "verified."
* **Reopen**: If the bug persists even after the developer has fixed the bug, the tester changes the status to "reopened". Once again the bug goes through the life cycle.
* **Closed**: If the bug is no longer exists then tester assigns the status "Closed."
* **Duplicate**: If the defect is repeated twice or the defect corresponds to the same concept of the bug, the status is changed to "duplicate."
* **Rejected**: If the developer feels the defect is not a genuine defect then it changes the defect to "rejected."
* **Deferred**: If the present bug is not of a prime priority and if it is expected to get fixed in the next release, then status "Deferred" is assigned to such bugs
* **Not a bug**:If it does not affect the functionality of the application then the status assigned to a bug is "Not a bug".

## Defect Life Cycle Explained

[](https://www.guru99.com/images/defectcyclechart.png)

* 1. Tester finds the defect
  2. Status assigned to defect- New
  3. A defect is forwarded to Project Manager for analyze
  4. Project Manager decides whether a defect is valid
  5. Here the defect is not valid- a status is given "Rejected."
  6. So, project manager assigns a status **rejected**. If the defect is not rejected then the next step is to check whether it is in scope. Suppose we have another function- email functionality for the same application, and you find a problem with that. But it is not a part of the current release when such defects are assigned as a **postponed or deferred**status.
  7. Next, the manager verifies whether a similar defect was raised earlier. If yes defect is assigned a status **duplicate**.
  8. If no the defect is assigned to the developer who starts fixing the code. During this stage, the defect is assigned a status **in- progress.**
  9. Once the code is fixed. A defect is assigned a status **fixed**
  10. Next, the tester will re-test the code. In case, the[Test Case](https://www.guru99.com/test-case.html)passes the defect is **closed.** If the test cases fail again, the defect is **re-opened** and assigned to the developer.
  11. Consider a situation where during the 1st release of Flight Reservation a defect was found in Fax order that was fixed and assigned a status closed. During the second upgrade release the same defect again re-surfaced. In such cases, a closed defect will be **re-opened.**

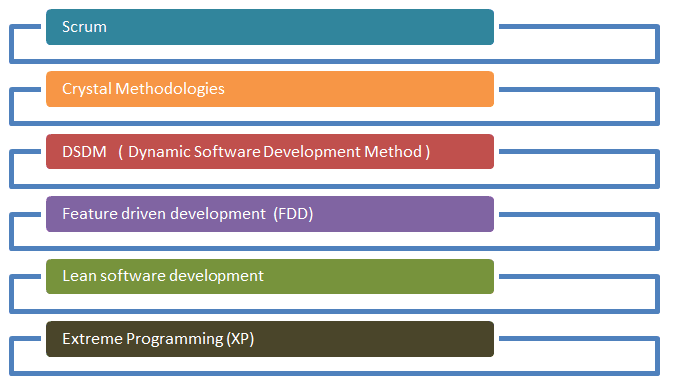
**Agile:**

## Agile Vs Waterfall Method

Agile and Waterfall model are two different methods for software development process. Though they are different in their approach, both methods are useful at times, depending on the requirement and the type of the project.

|  |  |
| --- | --- |
| **Agile Model** | **Waterfall Model** |
| * Agile method proposes incremental and iterative approach to software design | * Development of the software flows sequentially from start point to end point. |
| * The **agile process** is broken into individual models that designers work on | * The design process is not broken into an individual models |
| * The customer has early and frequent opportunities to look at the product and make decision and changes to the project | * The customer can only see the product at the end of the project |
| * Agile model is considered unstructured compared to the waterfall model | * Waterfall model are more secure because they are so plan oriented |
| * Small projects can be implemented very quickly. For large projects, it is difficult to estimate the development time. | * All sorts of project can be estimated and completed. |
| * Error can be fixed in the middle of the project. | * Only at the end, the whole product is tested. If the requirement error is found or any changes have to be made, the project has to start from the beginning |
| * Development process is iterative, and the project is executed in short (2-4) weeks iterations. Planning is very less. | * The development process is phased, and the phase is much bigger than iteration. Every phase ends with the detailed description of the next phase. |
| * Documentation attends less priority than software development | * Documentation is a top priority and can even use for training staff and upgrade the software with another team |
| * Every iteration has its own testing phase. It allows implementing regression testing every time new functions or logic are released. | * Only after the development phase, the testing phase is executed because separate parts are not fully functional. |
| * In agile testing when an iteration end, shippable features of the product is delivered to the customer. New features are usable right after shipment. It is useful when you have good contact with customers. | * All features developed are delivered at once after the long implementation phase. |
| * Testers and developers work together | * Testers work separately from developers |
| * At the end of every sprint, user acceptance is performed | * User acceptance is **performed** at the end of the project. |
| * It requires close communication with developers and together analyze requirements and planning | * Developer does not involve in requirement and planning process. Usually, time delays between tests and coding |

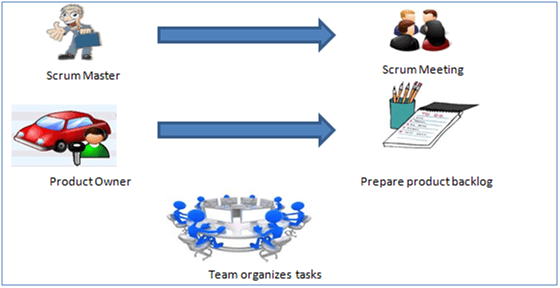
### Agile Methodology

[](https://www.guru99.com/images/11-2014/agile_Processesv1_2.png)

There are various **methods** present in agile testing, and those are listed below:

## Scrum

SCRUM is an agile development method which concentrates specifically on how to manage tasks within a team-based development environment. Basically, Scrum is derived from activity that occurs during a rugby match. Scrum believes in empowering the development team and advocates working in small teams (say- 7 to 9 members). It consists of three roles, and their responsibilities are explained as follows:

[](https://www.guru99.com/images/11-2014/agile_Processesv1_3.png)

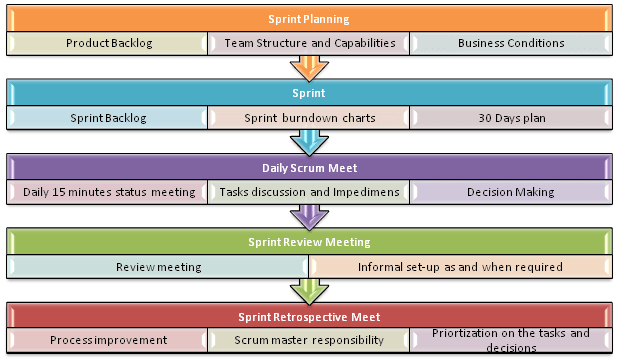
* Scrum Master
  + Master is responsible for setting up the team, sprint meeting and removes obstacles to progress
* Product owner
  + The Product Owner creates product backlog, prioritizes the backlog and is responsible for the delivery of the functionality at each iteration
* Scrum Team
  + Team manages its own work and organizes the work to complete the sprint or cycle

## Product Backlog

This is a repository where requirements are tracked with details on the no of requirements(user stories) to be completed for each release. It should be maintained and prioritized by Product Owner, and it should be distributed to the scrum team. Team can also request for a new requirement addition or modification or deletion

## Scrum Practices

Practices are described in detailed:

[](https://www.guru99.com/images/11-2014/agile_Processesv1_4.png)

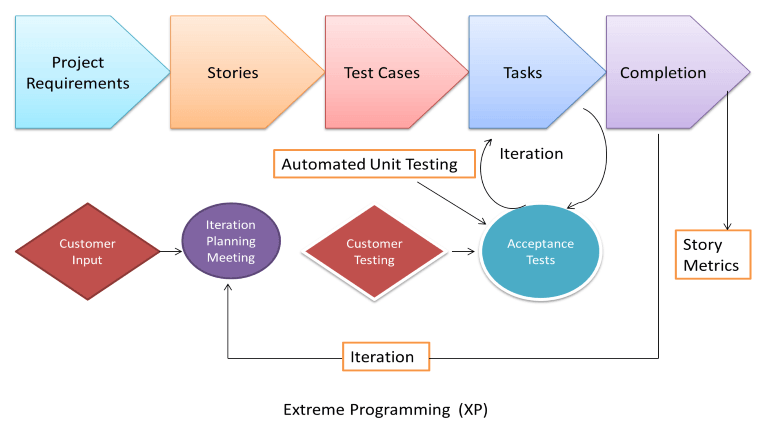
## Process flow of Scrum Methodologies:

Process flow of scrum testing is as follows:

* Each iteration of a scrum is known as Sprint
* Product backlog is a list where all details are entered to get the end-product
* During each Sprint, top user stories of Product backlog are selected and turned into Sprint backlog
* Team works on the defined sprint backlog
* Team checks for the daily work
* At the end of the sprint, team delivers product functionality

## eXtreme Programming (XP)

Extreme Programming technique is very helpful when there is constantly changing demands or requirements from the customers or when they are not sure about the functionality of the system. It advocates frequent "releases" of the product in short development cycles, which inherently improves the productivity of the system and also introduces a checkpoint where any customer requirements can be easily implemented. The XP develops software keeping customer in the target.

[](https://www.guru99.com/images/11-2014/agile_Processesv1_5.png)

Business requirements are gathered in terms of stories. All those stories are stored in a place called the parking lot.

In this type of methodology, releases are based on the shorter cycles called Iterations with span of 14 days time period. Each iteration includes phases like coding, unit testing and system testing where at each phase some minor or major functionality will be built in the application.

## Phases of eXtreme programming:

There are 6 phases available in Agile XP method, and those are explained as follows:

### Planning

* Identification of stakeholders and sponsors
* Infrastructure Requirements
* [Security](https://www.guru99.com/ethical-hacking-tutorials.html) related information and gathering
* Service Level Agreements and its conditions

### Analysis

* Capturing of Stories in Parking lot
* Prioritize stories in Parking lot
* Scrubbing of stories for estimation
* Define Iteration SPAN(Time)
* Resource planning for both Development and QA teams

### Design

* Break down of tasks
* Test Scenario preparation for each task
* Regression Automation Framework

### Execution

* Coding
* Unit Testing
* Execution of Manual test scenarios
* Defect Report generation
* Conversion of Manual to Automation regression test cases
* Mid Iteration review
* End of Iteration review

### Wrapping

* Small Releases
* Regression Testing
* Demos and reviews
* Develop new stories based on the need
* Process Improvements based on end of iteration review comments

### Closure

* Pilot Launch
* Training
* Production Launch
* SLA Guarantee assurance
* Review SOA strategy
* Production Support

There are two storyboards available to track the work on a daily basis, and those are listed below for reference.

* Story Cardboard
  + This is a traditional way of collecting all the stories in a board in the form of stick notes to track daily XP activities. As this manual activity involves more effort and time, it is better to switch to an online form.
* Online Storyboard
  + Online tool Storyboard can be used to store the stories. **Several teams can use it** for different purposes.

## Crystal Methodologies

Crystal Methodology is based on three concepts

1. **Chartering:** Various activities involved in this phase are creating a development team, performing a preliminary feasibility analysis, developing an initial plan and fine-tuning the development methodology
2. **Cyclic delivery:** The main development phase consists of two or more delivery cycles, during which the
   1. Team updates and refines the release plan
   2. Implements a subset of the requirements through one or more program test integrate iterations
   3. Integrated product is delivered to real users
   4. Review of the project plan and adopted development methodology
3. **Wrap Up:** The activities performed in this phase are deployment into the user environment, post- deployment reviews and reflections are performed.

## Dynamic Software Development Method (DSDM)

DSDM is a Rapid Application Development (RAD) approach to software development and provides an agile project delivery framework. The important aspect of DSDM is that the users are required to be involved actively, and the teams are given the power to make decisions. Frequent delivery of product becomes the active focus with DSDM. The techniques used in DSDM are

1. Time Boxing
2. MoSCoW Rules
3. Prototyping

The DSDM project consists of 7 phases

1. Pre-project
2. Feasibility Study
3. Business Study
4. Functional Model Iteration
5. Design and build Iteration
6. Implementation
7. Post-project

## Feature Driven Development (FDD)

This method is focused around "designing & building" features. Unlike other agile methods, FDD describes very specific and short phases of work that has to be accomplished separately per feature. It includes domain walkthrough, design inspection, promote to build, code inspection and design. FDD develops product keeping following things in the target

1. Domain object Modeling
2. Development by feature
3. Component/ Class Ownership
4. Feature Teams
5. Inspections
6. Configuration Management
7. Regular Builds
8. Visibility of progress and results

## Lean Software Development

Lean software development method is based on the principle "Just in time production". It aims at increasing speed of software development and decreasing cost. Lean development can be summarized in seven steps.

1. Eliminating Waste
2. Amplifying learning
3. Defer commitment (deciding as late as possible)
4. Early delivery
5. Empowering the team
6. Building Integrity
7. Optimize the whole

## Kanban

Kanban originally emerged from Japanese word that means, a card containing all the information needed to be done on the product at each stage along its path to completion. This framework or method is quite adopted in software testing method especially in agile testing.

### Scrum Vs Kanban

|  |  |
| --- | --- |
| **Scrum** | **Kanban** |
| * In scrum technique, test must be broken down so that they can be completed within one sprint | * No particular item size is prescribed |
| * Prescribes a prioritized product backlog | * Prioritization is optional |
| * Scrum team commits to a particular amount of work for the iteration | * Commitment is optional |
| * Burndown chart is prescribed | * No particular item size is prescribed |
| * Between each sprint, a scrum board is reset | * A Kanban board is persistent. It limits the number of items in workflow state |
| * It cannot add items to ongoing iteration | * It can add items whenever capacity is available |
| * WIP limited indirectly | * WIP limited directly |
| * Timeboxed iterations prescribed | * Timeboxed iterations optional |

## Agile metrics:

Metrics that can be collected for effective usage of Agile is:

* Drag Factor
  + Effort in hours which do not contribute to sprint goal
  + Drag factor can be improved by reducing number of shared resources, reducing the amount of non-contributing work
  + New estimates can be increased by percentage of drag factor -New estimate = (Old estimate+drag factor)
* Velocity
  + Amount of backlog(user stories) converted to shippable functionality of sprint
* No of Unit Tests added
* Time interval taken to complete daily build
* Bugs detected in an iteration or in previous iterations
* Production defect leakage

## What is Agile Testing?

**AGILE TESTING** is a testing practice that follows the rules and principles of agile software development. Unlike the Waterfall method, Agile Testing can begin at the start of the project with continuous integration between development and testing. Agile Testing is not sequential (in the sense it's executed only after coding phase) but continuous.

In this article, we will discuss

* [Test Plan for Agile.](https://www.guru99.com/agile-testing-a-beginner-s-guide.html#1)
* [Agile Testing Strategies.](https://www.guru99.com/agile-testing-a-beginner-s-guide.html#2)
* [The Agile Testing Quadrant.](https://www.guru99.com/agile-testing-a-beginner-s-guide.html#3)
* [QA challenges with agile software development.](https://www.guru99.com/agile-testing-a-beginner-s-guide.html#4)
* [Risk of Automation in Agile Process.](https://www.guru99.com/agile-testing-a-beginner-s-guide.html#5)

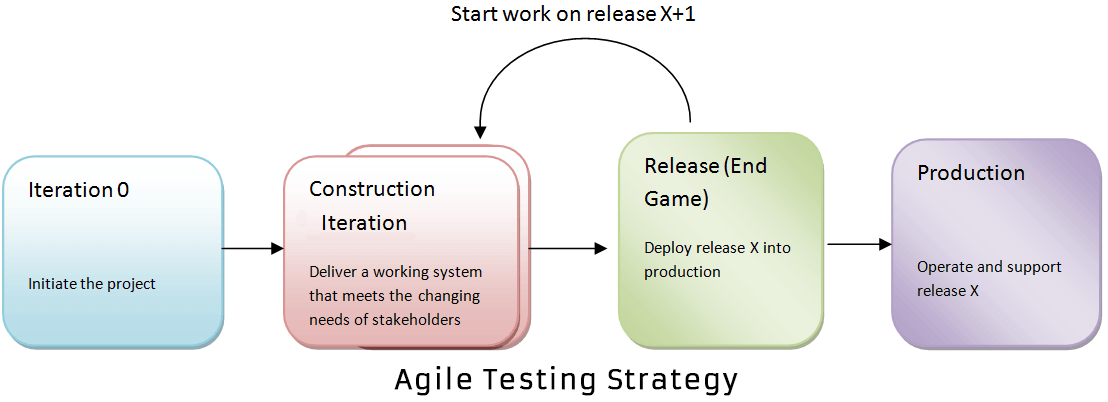
### Test Plan for Agile

Unlike the waterfall model, in an agile model, a test plan is written and updated for every release.  The agile test plan includes types of testing done in that iteration like test data requirements, infrastructure, test environments, and test results. Typical test plans  in agile includes

1. Testing Scope
2. New functionalities which are being tested
3. Level or Types of testing based on the features complexity
4. Load and Performance Testing
5. Infrastructure Consideration
6. Mitigation or Risks Plan
7. Resourcing
8. Deliverables and Milestones

### Agile Testing Strategies

Agile testing life cycle spans through four stages

[](https://www.guru99.com/images/11-2014/agile_testing_1.png)

**(a) Iteration 0**

During the first stage or iteration 0, you perform initial setup tasks. It includes identifying people for testing, installing testing tools, scheduling resources (usability testing lab), etc. The following steps are set to achieve in Iteration 0

a) Establishing a business case for the project

b) Establish the boundary conditions and the project scope

c) Outline the key requirements and use cases that will drive the design trade-offs

d) Outline one or more candidate architectures

e) Identifying the risk

f) Cost estimation and prepare a preliminary project

**(b) Construction Iterations**

The second phase of testing is Construction Iterations, the majority of the testing occurs during this phase. This phase is observed as a set of iterations to build an increment of the solution.  In order to do that, within each iteration, **the team implements** a hybrid of practices from XP, Scrum, Agile modeling, and agile data and so on.

In construction iteration, the agile team follows the prioritized requirement practice: With each iteration, they take the most essential requirements remaining from the work item stack and implement them.

Construction iteration is classified into two, confirmatory testing and investigative testing.  **Confirmatory testing concentrates** on verifying that the system fulfills the intent of the stakeholders as described to the team to date, and is performed by the team.  While the investigative testing detects the problem that confirmatory team has skipped or ignored.  In Investigative testing, tester determines the potential problems in the form of defect stories. Investigative testing deals with common issues like integration testing, load/stress testing, and security testing.

Again for, confirmatory testing there are two aspects **developer testing** and **agile acceptance testing. Both of them** are automated to enable continuous regression testing throughout the lifecycle.  Confirmatory testing is the agile equivalent of testing to the specification.

Agile acceptance testing is a combination of traditional functional testing and traditional acceptance testing as the development team, and stakeholders are doing it together.  While developer testing is a mix of traditional unit testing and traditional service integration testing.  Developer testing verifies both the application code and the database schema.

**(c) Release End Game Or Transition Phase**

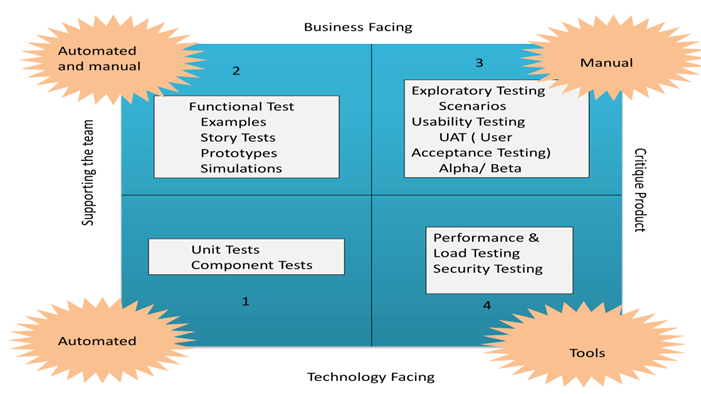
The goal of “Release, End Game” is to deploy your system successfully into production.  The activities include in this phase are training of end users, support people and operational people.  Also, it includes marketing of the product release, back-up & restoration, finalization of system and user documentation.

The final testing stage includes full system testing and acceptance testing.   In accordance to finish your final testing stage without any obstacles, you should have to test the product more rigorously while it is in construction iterations. During the end game, testers will be working on its defect stories.

**(d) Production**

After the release stage, the product will move to the production stage.

### The Agile Testing Quadrants

[](https://www.guru99.com/images/11-2014/agile_testing_2.png)

The agile testing quadrants separate the whole process in four Quadrants and help to understand how agile testing is performed.

a) **Agile Quadrant I** – The internal code quality is the main focus in this quadrant, and it consists of test cases which are technology driven and are implemented to support the team, it includes

1. Unit Tests

2.Component Tests

b) **Agile Quadrant II**– It **contains** test cases that are **business driven and are implemented** to support the team.  This Quadrant focuses on the requirements. The kind of test performed in this phase is

1. Testing of examples of possible scenarios and workflows

2. Testing of User experience such as prototypes

3. Pair testing

c) **Agile Quadrant III** – This quadrant provides feedback to quadrants one and two.  The test cases can be used as the basis to perform automation testing.  In this quadrant, many rounds of iteration reviews are carried out which builds confidence in the product.  The kind of testing done in this quadrant is

1. Usability Testing

2. Exploratory Testing

3. Pair testing with customers

4. Collaborative testing

5. User acceptance testing

d) **Agile Quadrant IV** – **This quadrant concentrates** on the non-functional requirements such as performance, security, stability, etc.  With the help of this quadrant, the application is made to deliver the non-functional qualities and expected value.

1. Non-functional tests such as stress and performance testing

2. Security testing with respect to **authentication**and hacking

3. Infrastructure testing

4. Data migration testing

5. Scalability testing

6. Load testing

### QA challenges with agile software development

a) Chances of error are more in agile, as documentation is given less priority, eventually puts more pressure on QA team

b) New features are introduced quickly, which reduces the available time for test teams to identify whether the latest features are according to the requirement and does it truly address the business suits

c) Testers are often required to play a semi-developer roled

d) Test execution cycles are highly compressed

e) Very less time to prepare test plan

f) For regression testing, they will have minimal timing

g) Change in their role from being a gate-keeper of quality to being a partner in Quality

h) Requirement changes and updates are inherent in an agile method, becoming the biggest challenge for QA

### Risk of Automation in Agile Process

* Automated UI provides a high level of confidence, but they are slow to execute, fragile to maintain and expensive to build. Automation may not significantly improve test productivity unless the testers know how to test
* Unreliable tests are a major concern in automated testing. Fixing failing tests and resolving issues related to brittle tests should be a top priority in order to avoid false positives
* If the automated test are initiated manually rather than through CI (Continuous Integration) then there is a risk that they are not regularly running and therefore may cause failing of tests
* Automated tests are not a replacement for an exploratory manual testing. To obtain the expected quality of the product, a mixture of testing types and levels is required
* Many commercially available automation tools provide simple features like automating the capture and replay of manual test cases. Such tool encourages testing through the UI and leads to an inherently brittle and difficult to maintain tests. Also, storing test cases outside the version control system creates unnecessary complexity
* In order to save time, much times the automation test plan is poorly planned or unplanned which results in the test fail
* A test set up and tear down procedures are usually missed out during test automation, while Performing manual testing, a test set up and tear down procedures sounds seamless
* Productivity metrics such as a number of test cases created or executed per day can be terribly misleading, and could lead to making a large investment in running useless tests
* Members of the agile automation team must be effective consultants: approachable, cooperative, and resourceful, or this system will quickly fail
* Automation may propose and deliver testing solutions that require too much ongoing maintenance relative to the value provided
* Automated testing may lack the expertise to conceive and deliver effective solutions
* Automated testing may be so successful that they run out of important problems to solve, and thus turn to unimportant problems.

## What is Scrum?

Building complex software applications is a difficult task. Scrum methodology comes as a solution for executing such a complicated task. It helps the development team to focus on all aspects of the product like quality, performance, usability and so on.

## What is Scrum Testing?

**SCRUM TESTING** is testing done in Scrum methodology to verify the software application meets requirements. Scrum Testing also involves checking non-functional parameters like security, usability, performance etc. There is no active role of Tester in the Scrum Process. Usually, testing is carried out by a developer with Unit Test. Some Scrum projects do have dedicated test teams depending on the nature & complexity of the project.

In this tutorial, you will learn-

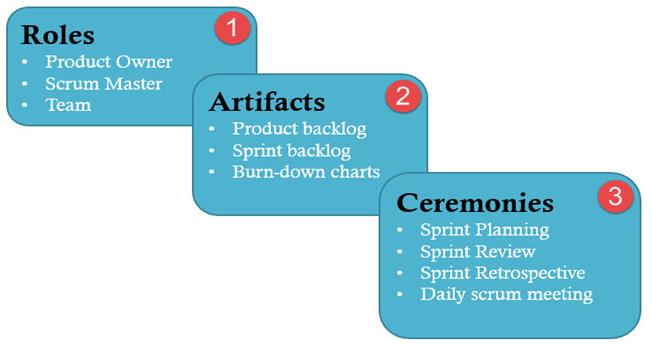
* [What is Scrum?](https://www.guru99.com/scrum-testing-beginner-guide.html#1)
* [Key Features of Scrum Methodology](https://www.guru99.com/scrum-testing-beginner-guide.html#2)
* [Roles in Scrum](https://www.guru99.com/scrum-testing-beginner-guide.html#3)
* [Scrum Artifacts](https://www.guru99.com/scrum-testing-beginner-guide.html#4)
* [Ceremonies (Processes) in Scrum](https://www.guru99.com/scrum-testing-beginner-guide.html#5)
* [Role of Tester in Scrum](https://www.guru99.com/scrum-testing-beginner-guide.html#6)
* [Testing Activities in Scrum](https://www.guru99.com/scrum-testing-beginner-guide.html#7)
* [Test Reporting](https://www.guru99.com/scrum-testing-beginner-guide.html#8)

## Key Features of Scrum Methodology

Following are Key Features of Scrum-

* Scrum has a short fixed schedule of release cycles with adjustable scope known as **sprints** to address rapidly changing development needs. Each release could have multiple sprints. Each Scrum Project could have multiple Release Cycles.
* A repeating sequence of **meetings, events, and milestones**
* A practice of testing and implementing new requirements, known as **stories**, to make sure some work is released ready after each sprint

Scrum is based on the following 3 Pillars-

[](https://www.guru99.com/images/11-2014/112714_1232_ScrumTestin1.jpg)

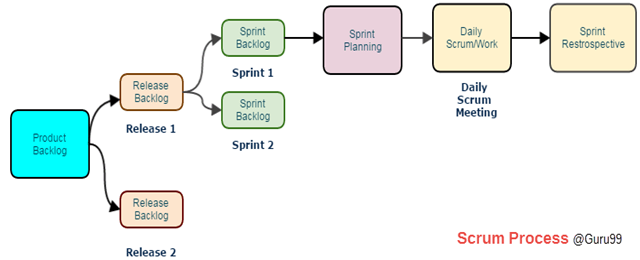
Let's look at the one by one

### 1. Roles in Scrum

There are three chief roles in Scrum Testing – Product Owner, Scrum Master and The Development Team. Let's study them in detail

|  |  |  |
| --- | --- | --- |
| **Product Owner** | **Scrum Master** | **The Team** |
| * He/She defines features of the product. | * He/She manages the team and look after the team's productivity | * The team is usually about 5-9 members |
| * Product Owner decides the release date and corresponding features | * He/She maintains the block list and removes barriers in the development | * It includes developers, designer and sometimes testers, etc. |
| * They prioritize the features according to the market value and profitability of the product | * He/She coordinates with all roles and functions | * The team organizes and schedule their work on their own |
| * He/She is responsible for the profitability of the product | * He/She shields team from external interferences | * Has right to do everything within the boundaries of the project to meet the sprint goal |
| * He/She can accept or reject work item result | * Invites to the daily scrum, sprint review and planning meetings | * Actively participate in daily ceremonies |

### 2. Scrum Artifacts

[](https://www.guru99.com/images/11-2014/112714_1232_ScrumTestin2.png)

A scrum process includes

* **User stories:**They are a short explanation of functionalities of the system under test. Example for Insurance Provider is – "Premium can be paid using the online system."
* **Product Backlog:** It is a collection of user stories captured for a scrum product.**The product owner prepares** and maintains the product backlog. It is prioritized by the product owner, and anyone can add to it with approval from the product owner.
* **Release Backlog:**A release is a time frame in which the number of iterations is completed.**The product owner co-ordinates** with the scrum master to decide which stories should be targeted for a release. Stories in the release backlog are targeted to be completed in a release.
* **Sprints:**It is a set period of time to complete the user stories, decided by the product owner and developer team, usually 2-4 weeks of time.
* **Sprint Backlog:**It's a set of user stories to be completed in a sprint. During sprint backlog, work is never assigned, and the team signs up for work on their own. It is owned and managed by the team while the estimated work remaining is updated daily. It is the list of task that has to be performed in Sprint
* **Block List:**It is a list of blocks and unmade decisions owned by scrum master and updated daily
* **Burndown chart:**Burn-down chart represents overall progress of the work in progress and work completed throughout the process. It represents in a graph format the stories and features not completed

### 3. Ceremonies (Processes) in Scrum

* **Sprint Planning:**A sprint begins with the team importing stories from the release backlog into the sprint backlog; it is hosted by scrum master. The Testers estimate effort to test the various stories in the Sprint Backlog.
* **Daily Scrum:**It is hosted by scrum master, it last about 15 minutes. During Daily Scrum, the members will discuss the work completed the previous day, the planned work for the next day and issues faced during a sprint. During daily stand-up meeting team progress is tracked.
* **Sprint Review/ Retrospective:**It is also hosted by scrum master, it last about 2-4 hours and discuss what the team has accomplished in the last sprint and what lessons were learned.

### Role of Tester in Scrum

[](https://www.guru99.com/images/11-2014/112714_1232_ScrumTestin3.png)

**There is no active role of Tester in the Scrum** Process. Usually, testing is carried out by a developer with Unit Test. While product owner is also frequently involved in the testing process during each sprint. **Some Scrum projects do have dedicated test teams depending on the nature & complexity of the project**.

The next question is, what tester do in a scrum? Following note will answer

## Testing Activities in Scrum

Testers do following activities during the various stages of Scrum-

**Sprint Planning**

* In sprint planning, a tester should pick a user-story from the product backlog that should be tested.
* As a tester, he/she should decide how many hours (Effort Estimation) it should take **to finish** testing for each of selected user stories.
* As a tester, he/she must know what sprint goals are.
* As a tester, contribute to the prioritizing process

**Sprint**

* Support developers in unit testing
* Test user-story when completed. **Test execution is performed** in a lab where both tester and developer work hand in hand. Defect are logged in Defect Management tool which are tracked on a daily basis. Defects can be conferred and analyzed during the scrum meeting. Defects are retested as soon as it is **resolved** and deployed for testing
* As a tester, he/she attends all daily standup meeting to speak up
* As a tester, he/ she can bring any backlog item that cannot be completed in the current sprint and put to the next sprint
* Tester is responsible for developing automation scripts. He schedules automation testing with Continuous Integration (CI) system. Automation receives the importance due to short delivery timelines. Test Automation can be accomplished by utilizing various open source or paid tools available in the market. This proves effective in ensuring that everything that needs to be tested was covered. Sufficient Test coverage can be achieved with a close communication with the team.
* Review CI automation results and send Reports to the stakeholders
* Executing non-functional testing for approved user stories
* Coordinate with customer and product owner to define acceptance criteria for Acceptance Tests
* At the end of the sprint, the tester also does acceptance testing(UAT) in some case and confirms testing completeness for the current sprint

**Sprint Retrospective**

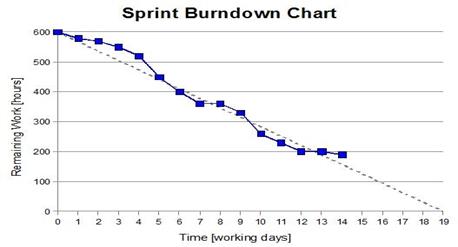
* As a tester, he will figure out what went wrong and what went right in the current sprint
* As a tester, he identifies lesson learned and best practices

## Test Reporting

Scrum Test metrics reporting provides transparency and visibility to stakeholders about the project. The metrics that are reported allow a team to analyze their progress and plan their future strategy to improve the product. There are two metrics that are frequently used to report.

**Burn down chart:** Each day, Scrum Master records the estimated remaining work for the sprint. This is nothing but the Burn Down Chart. It is updated daily.

A burndown chart gives a quick overview of the project progress, this chart contains information like the total amount of work in the project that must be completed, amount of work completed during each sprint and so on.

[](https://www.guru99.com/images/11-2014/112714_1232_ScrumTestin4.jpg)

**Velocity history graph:** The velocity history graph predicts the velocity of the team reached in each sprint. It is a bar graph and represents how teams output has changed over time.

The additional metrics that may be useful are schedule burn, budget burn, theme percent complete, stories completed - stories remaining and so on.

Do you have any tips or experiences to share for Scrum Testing? Do leave a comment below-

## What is a Scaled Agile Framework (SAFe)?

Scaled Agile Framework SAFe, is a freely available online knowledge base that allows you to apply lean-agile practices at the enterprise level. It provides a simple, lightweight experience for the software development team. The whole framework is divided into three segments **Team, Program and Portfolio**. We will see this in detail later on. SAFe allows team for,

* Implementing Lean-Agile software and systems in enterprise level
* It's based on Lean and Agile principles.
* It gives detailed guidance for work at the enterprise Portfolio, Value Stream, Program, and Team.
* It's designed to meet the needs of all stakeholders within an organization.

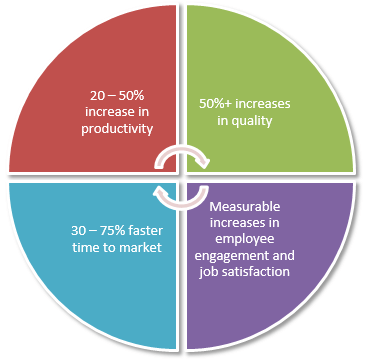
SAFe was first developed in the field and was elaborated in **Dean Leffingwell'**s books and blog. Version 1.0 is the first official release in 2011. The latest version is 4.6, was released in October 2018. It provides guidance to work at enterprise Portfolio, Value Stream, Program, and Team levels.

In this tutorial, you will learn-

* [What is Scaled Agile Framework (SAFe)](https://www.guru99.com/scaled-agile-framework.html#1)
* [Why to use Agile Framework](https://www.guru99.com/scaled-agile-framework.html#2)
* [When to Use Scaled Agile Framework](https://www.guru99.com/scaled-agile-framework.html#3)
* [How different than other Agile practices](https://www.guru99.com/scaled-agile-framework.html#4)
* [Foundations of Scaled Agile Framework](https://www.guru99.com/scaled-agile-framework.html#5)
* [Agile Manifesto](https://www.guru99.com/scaled-agile-framework.html#6)
* [Different Levels in SAFE](https://www.guru99.com/scaled-agile-framework.html#7)
  + [Team Level](https://www.guru99.com/scaled-agile-framework.html#8)
  + [Program Level](https://www.guru99.com/scaled-agile-framework.html#9)
  + [Portfolio Level](https://www.guru99.com/scaled-agile-framework.html#10)
  + [Value Stream Level](https://www.guru99.com/scaled-agile-framework.html#11)

## Why to use Agile Framework

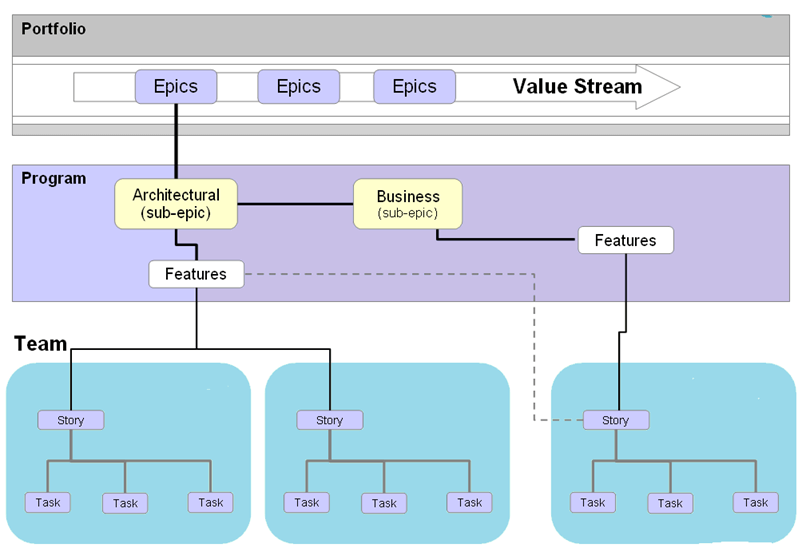
It is simpler and lighter in weight, yet it expands to handle the needs of large value streams and complex system development. By implementing an Agile Framework, you will have following benefits,

[](https://www.guru99.com/images/8-2016/090116_0956_WhatisScale2.png)

* **Productivity increased** by **20 - 50%**
* **Quality** increased more than **50%**
* **Time to Market** is faster than **30 -75%**
* Increased **employee engagement** and **job satisfaction.**

The detailed framework diagram is available on the [website](http://www.scaledagileframework.com/). It shows all of the key roles, Activities, deliverables, and flows. It also serves as a navigational aid to the rest of the site.

The below image explains how agile process works. Epics are a large body of work, which is further broken down into a number of smaller stories or sub-epics. These sub-epics are allocated to the team as a story. Each team then work on these stories or software features accordingly.

[](https://www.guru99.com/images/8-2016/090116_0956_WhatisScale3.png)

## When to Use Scaled Agile Framework

[](https://www.guru99.com/images/8-2016/090116_0956_WhatisScale4.jpg)

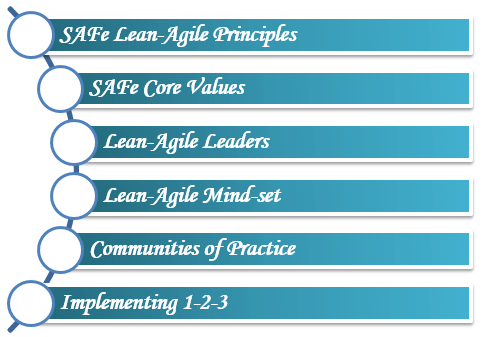
* When a team is interested to implement an agile approach consistently across larger, multi-team programs and portfolios.
* When multiple teams are running their own way of Agile implementation but regularly facing obstacles, delays, and failures.
* When teams want to work independently.
* When you want to scale Agile across the organization but not sure what new roles may be needed or what existing roles (i.e., management) need to change and how.
* When you have attempted to scale the Agile across your organization but struggling in alignment to achieve uniform or consistent strategy across business departments from portfolio to program and team levels.
* When an organization needs to improve its product development lead time and want to know how other companies have succeeded in scaling Agile with SAFe.

## How different than other Agile practices

Let's see how Scaled Agile framework is different from other agile practices,

* It's publicly available and free to use.
* Available in a highly approachable and usable form.
* It's lightweight, practically proven results and specific to level.
* It constantly/regularly modifies/maintains most commonly used agile practices.
* Offers useful extensions to common agile practices.
* Grounds agile practices to an enterprise context.
* Offers complete picture of software development.
* Visibility or transparency is more on all the levels.
* Continues or regular feedback on quality and improvement.

## Foundations of Scaled Agile Framework

[](https://www.guru99.com/images/8-2016/090116_0956_WhatisScale5.png)

Scaled Agile Framework(SAFe): It stands on the foundations of its

1. Lean-Agile Principles
2. Core Values,
3. Lean-Agile Leadership
4. Lean-Agile Mind-set,
5. Communities of Practice(Group of people who are constantly working on SAFe practices)
6. Implementing 1-2-3
7. **SAFe Lean-Agile Principles**

These basic principles and values for SAFe must be understood, exhibited and continued in order to get the desired results.

* Take an economic view
* Apply systems thinking
* Assume variability; preserve options
* Build incrementally with fast, integrated learning cycles
* Base milestones on an objective evaluation of working systems
* Visualize and limit WIP, reduce batch sizes and manage queue lengths
* Apply cadence, synchronize with cross-domain planning
* Unlock the intrinsic motivation of knowledge workers
* Decentralize decision-making

1. **SAFe Agile Core Values**

The SAFe agile is based on these four values.

**Alignment:**

* SAFe supports alignment.
* Alignment starts at,
  + Strategic Themes in Portfolio Backlog and
  + Moves down to Vision and Roadmap of Program Backlogs and then
  + Moves to the Team Backlogs.

**Built-in Quality:**

* It ensures that every incremental delivery reflects the quality standards.
* Quality is not "added later" is built in.
* Built-in quality is a prerequisite of Lean and its mandatory

**Transparency:**

* Transparency is the enabler for trust.
* SAFe helps the enterprise to achieve transparency at all levels- Executives, Portfolio Managers, and other stakeholders.
* Everyone can see into the portfolio backlog/Kanban, program backlogs/Kanban, and Team Backlog/Kanban.
* Each level has a clear understanding of the PI goals.
* Train Programs have visibility into the team's backlogs, as well other program backlogs
* Teams and programs have visibility into business and architecture Epics. They can see what might be headed their way.

**Program Execution:**

* SAFe places great focus on working systems and resultant business outcomes.
* SAFe is not useful if teams can't execute and continuously deliver value.

1. **Lean Agile Leaders:**

The Lean-Agile Leaders are lifelong learners and teachers. It helps teams to build better systems through understanding and exhibiting the Lean-Agile SAFe Principles.

As an enabler for the teams, the ultimate responsibility is adoption, success and ongoing improvement of Lean-Agile developments. For the change and continuous improvement, leaders must be trained.

Leaders need to adopt a new style of leadership. One that truly empowers and engages individuals and teams to reach their highest potential.

**Principles of these Lean-Agile Leaders**

* Lead the Change
* Know the Way; Emphasize Lifelong Learning
* Develop People
* Inspire and Align with Mission; Minimize Constraints
* Decentralize Decision-Making
* Unlock the Intrinsic Motivation of Knowledge Workers

1. **Lean Agile Mind-Set:**

Lean-Agile mindset is represented in two things:

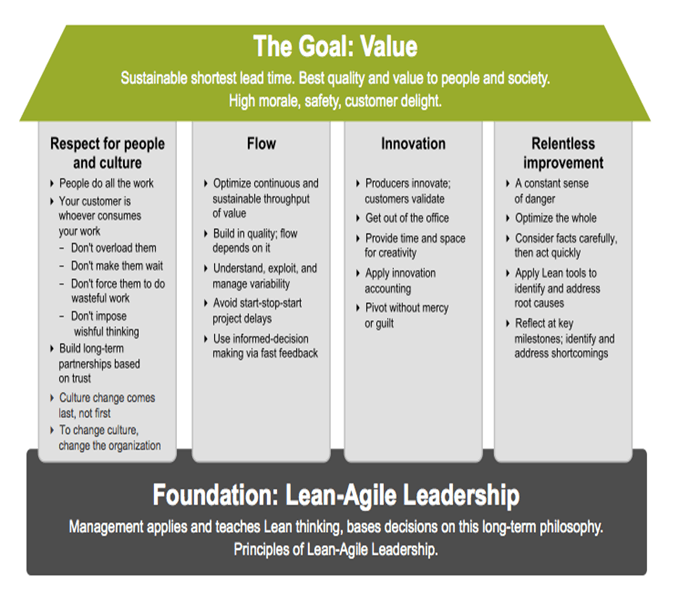
1. **The SAFe House of Lean**
2. **Agile Manifesto**

**The SAFe House of Lean**:

SAFe is derived from Lean manufacturing principles and practices. Based on these factors SAFe presents the "SAFe House of Lean". It is inspired by "house" of lean Toyota.

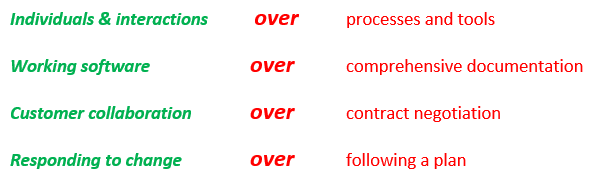
The Goal of lean is unbeatable: To deliver maximum customer value in the shortest lead time with the highest possible quality to customer

Below figure explains the Goal, Pillars, and Foundation of "SAFe House of Lean."

[](https://www.guru99.com/images/8-2016/090116_0956_WhatisScale6.png)

## Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

[](https://www.guru99.com/images/8-2016/2016-09-10_11-19-04.png)

That' why , while there is a value in the items on the right, we value the items on the left more.

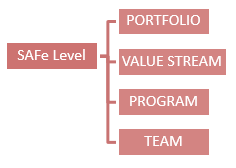
**Agile Manifesto**

1. The highest priority is to satisfy the customer through continuous and early delivery of valuable software.
2. Embrace the changing requirements, even late in development. Agile processes harness change for the customer's benefit.
3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.
4. Developers and business people must work together daily throughout the project.
5. Build projects around motivated individuals. Give them support and the environment they need, and trust them to get the job done.
6. The most efficient method for communication with a development team is a face-to-face conversation.
7. Working software is the primary measure of progress.
8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.
9. Continuous attention to technical excellence and good design enhances agility.
10. Simplicity--the art of maximizing the amount of work not done--is essential.
11. The best architectures, requirements, and designs emerge from self-organizing teams.
12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

## Different Levels in SAFE

There are two different types of SAFe implementation:

1. **SAFe 4.0 implementation**
2. **SAFe 3.0 implementation**

[](https://www.guru99.com/images/8-2016/090116_0956_WhatisScale8.png)

* In SAFe 4.0 implementation we have 4-Levels: **Portfolio, Value Stream, Program, and Team.**
* In SAFe 3.0 implementation we have 3-Levels: **Portfolio, Program, and Team**
* 3-Level SAFe is for smaller implementations with 100 or fewer people. Programs that do not require significant collaboration.
* 4-Level SAFe is for solutions that typically require many hundreds of practitioners to develop deploy and maintain software.

### Team Level

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Roles/Teams** |  | **Events** |  | **Artifacts** |
| \* Agile Team |  | \* Sprint Planning |  | \* Team Backlog |
| \* Product Owner |  | \* Backlog Grooming |  | \* Non-Functional Requirements |
| \* Scrum Master |  | \* Daily Stand-Up |  | \* Team PI Objectives |
|  |  | \* Execution |  | \* Iterations |
|  |  | \* Sprint Demo |  | \* Stories(Working Software) |
|  |  | \* Sprint Retrospective |  | \* Sprint Goals |
|  |  | \* IP Sprints |  | \* Built-In Quality |
|  |  |  |  | \* Spikes |
|  |  |  |  | \* Team Kanban |

* All SAFe teams are part of one or other Agile Release Train (ART).
* SAFe teams are empowered, self-organizing, self-managing, cross-functional teams
* Each team is equally responsible for defining, building and testing stories from their Team Backlog in a fixed-length Iterations
* Teams plan and execute two-week time-boxed iterations in accordance with agreed-to Iteration Goals.
* Teams will use ScrumXP/Team Kanban routine to deliver high-quality systems to produce a System Demo on every two weeks.
* All different teams in the ART (Agile Release Trains) will create an integrated and tested system. Stakeholders will evaluate and respond with fast feedback
* They apply Built-in Quality practices.
* Each ScrumXP team will have 5-9 team members, which includes all the roles necessary to build a quality incremental value in each Iteration.
* ScrumXP roles includes:
  + Team(Dev+QA)
  + Scrum Master
  + Product Owner. Etc..
* SAFe divides the development timeline into a set of iterations within a PI (Program Increment).
* PI duration is between 8 -12 weeks.
* The team will use stories to deliver the value. The Product Owner will have content authority over their creation and acceptance of the stories.
* Stories contain Customer's requirements.
* Team Backlog includes user and enabler stories, which are identified during PI planning. When the Product Management presents the Roadmap, Vision, and Program Backlog.
* Identifying, elaborating, prioritizing, scheduling, implementing, testing, and accepting the stories are the primary requirements of management work in team level.
* Each iteration provides:
  + A valuable increment of new functionality
  + Accomplish via constantly repeating pattern
  + Plan the iteration
  + Commit to some functionality
  + Execute the iteration by building and testing Stories
  + Demo the new functionality
  + Retrospective
  + Repeat for the next iteration
* Teams also support the System Demo at the end of each Iteration. which is the critical integration point for the ART.
* Larger Value Streams will have multiple ARTs.
* The Innovation and Planning (IP) Iterations leverage the teams with an opportunity for innovation and exploration.

### Program Level

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Roles/Teams** |  | **Events** |  | **Artifacts** |
| \* DevOps |  | \* PI(Program Increment) Planning |  | \* Vision |
| \* System Team |  | \* System Demos |  | \* Roadmap |
| \* Release Management |  | \* Inspect and Adopt Workshop |  | \* Metrics |
| \* Product Management |  | \* Architectural Runway |  | \* Milestones |
| \* UEX Architect |  | \* Release Any Time |  | \* Releases |
| \* Release Train Engineer(RTE) |  | \* Agile Release Train |  | \* Program Epics |
| \* System Architect/Engineer |  | \* Release |  | \* Program Kanban |
| \* Business Owners |  |  |  | \* Program Backlog |
| \* Lean-Agile Leaders |  |  |  | \* Non-Functional Requirements |
| \* Communities of Practice |  |  |  | \* Weighted Shortest Job First (WSJF) |
| \* Shared Services |  |  |  | \* Program PI Objectives |
| \* Customer |  |  |  | \* Feature |
|  |  |  |  | \* Enabler |
|  |  |  |  | \* Solution |
|  |  |  |  | \* Value Stream Coordination |

* In Program level, Value of SAFe is delivered by long-lived Agile Release Trains (ART). Iteration is for team and train is for the program.
* Agile Release Trains (ART) is the primary vehicle for value delivery at the program level. It delivers a value stream to the organization.
* The Program Increments (PIs) duration is of 8 to 12 weeks.
* ART is of 5 - 12 Agile Teams (~50 – 125+ people) which includes all the roles and infrastructure needed to deliver fully tested, working, system-level software.
* Each PI is a multiple-Iteration time box. During which a significant, valuable increment of the system is developed and delivered.
* In each PI a "demo" and "Inspect and adapt" sessions will happen, and Planning begins for the next PSI.
* At the Program level, SAFe emphasis on the principle of alignment. This is because multiple agile team efforts are integrated to create customer value.
* SAFe artifact hierarchy is **Epics->features->user stories**.
* At a Program level, Product Manager/Program Manager has content authority. He defines and prioritizes the program backlog.
* Program backlog is a prioritized list of features.
* At the program level, features can be originated, or they can derive from epics defined at the portfolio level.
* Features decompose to user stories and flow into team-level backlogs.
* Product Manager or the Release Train Engineer role could be handled by the Program Manager/Senior Project Manager
* System Architect role at the program level is to collaborate day to day work with the teams. It ensures that non-functional requirements are met. Also, they work with the enterprise architect at the portfolio level to make sure there is sufficient architectural runway to support upcoming user and business needs.
* Interface design, user experience guidelines and design elements for the teams are provided by UX Designers.
* Chief-Scrum Master role is played by 'Release Train Engineer'.
* Various team (from marketing, development, quality, operations, and deployment) forms 'Release Management Team'. They will approve routine releases of quality solutions to customers.
* Deployment of software into customer environments and successful delivery is taken care of by DevOps team.

### Portfolio Level

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Roles/Teams** |  | **Events** |  | **Artifacts** |
| \* Enterprise Architect |  | \* Strategic Investment Planning |  | \* Strategic Themes |
| \* Program Portfolio Mgmt |  | \* Kanban Portfolio(Epic) Planning |  | \* Enterprise |
| \* Epic Owners |  |  |  | \* Portfolio Backlog |
|  |  |  |  | \* Portfolio Kanban |
|  |  |  |  | \* Non-Functional Requirements |
|  |  |  |  | \* Epic and Enabler |
|  |  |  |  | \* Value Stream |
|  |  |  |  | \* Budgets(CapEx and OpEx) |

* Highest level of interest/ concern /involvement/ in SAFe is **SAFe Portfolio**
* The portfolio provides the basic blocks for organizing the Lean-Agile Enterprise flow of value via one or more Value Streams.
* The portfolio helps to develop systems and solutions which are described in strategic themes (links a SAFe portfolio to the changing business strategy of an enterprise).
* To meet strategic objectives, portfolio level encapsulates these elements. It provides basic budgeting and other governance mechanisms. This way it assures that the investment in the value streams provides the returns necessary for the enterprise.
* A portfolio is connected to business bi-directionally:
  + In order to guide the Portfolio to the larger changing business objectives, it provides strategic themes.
  + Another direction indicates the constant flow of portfolio values.
* Program Portfolio Management acts as stakeholders, and they are accountable to deliver the business results.
* SAFe Portfolio Level contains people, processes and necessary build systems and solutions that an enterprise needs to meet its strategic objectives.
* Value Streams are the primary objectives in Portfolio, with which funding for the people and other resources required to build the Solutions.
* Important key concepts used here are:
  + Connection to the Enterprise,
  + Program Portfolio Management,
  + Managing the Flow of Portfolio Epics.

### Value Stream Level

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Roles/Teams** |  | **Events** |  | **Artifacts** |
| \* DevOps |  | \* Pre and Post PI(Program Increment) Planning |  | \* Vision |
| \* System Team |  | \* Solution Demos |  | \* Roadmap |
| \* Release Management |  | \* Inspect and Adopt Workshop |  | \* Metrics |
| \* Solution Management |  | \* Agile Release Train |  | \* Milestones |
| \* UEX Architect |  |  |  | \* Releases |
| \* Value Stream Engineer(RTE) |  |  |  | \*Value Stream Epics |
| \* Solution Architect/Engineer |  |  |  | \* Value Stream Kanban |
| \* Shared Services |  |  |  | \* Value Stream Backlog |
| \* Customer |  |  |  | \* Non-Functional Requirements |
| \* Supplier |  |  |  | \* Weighted Shortest Job First (WSJF) |
|  |  |  |  | \* Value Stream PI Objectives |
|  |  |  |  | \* Capability |
|  |  |  |  | \* Enabler |
|  |  |  |  | \* Solution Context |
|  |  |  |  | \* Value Stream Coordination |
|  |  |  |  | \* Economic Framework |
|  |  |  |  | \* Solution Intent |
|  |  |  |  | \* MBSE |
|  |  |  |  | \* Set Based |
|  |  |  |  | \* Agile Architecture |

* The Value Stream Level is optional in SAFe.
* Value Stream Level is new in SAFe 4.0.
* The Value Stream Level is intended/designed for Enterprises /builders/organization who are:

1. Large in size
2. Independent
3. Have complex solutions
4. Their solutions typically require multiple ARTs
5. They have Suppliers contribution.
6. They face the largest systems challenges
7. For cyber-physical systems
8. For software, hardware, electrical and electronics, optics, mechanics, fluidics and more.

* Building this kind of systems often takes hundreds, even thousands of practitioners, external and internal suppliers.
* If the systems are mission crucial. Failure of the Solution, or even a subsystem, has unacceptable economic and social consequences.
* If the Enterprises can be built with a few hundred practitioners, it may not need the constructs of this level. In that case, they can use from the '**collapsed view'** which is 3-level SAFe.
* Building value stream solutions in a Lean-Agile pattern requires additional artifacts, coordination, and constructs. So this level contains an Economic Framework to provide financial boundaries for Value Stream
* It supports cadence and synchronization for multiple ARTs and Suppliers. It includes Pre-and Post-PI Planning meetings and Solution Demo.
* It gives additional roles which are: Value Stream Engineer, Solution Architect/Engineering, and Solution Management.

**Summary:**

* SAFe is an industry-proven, value-focused method for scaling Agile at the Enterprise level.
* It answers the questions like "How do we plan?", "How do we budget?", and "How do we become cross-functional in architecture and DevOps?"
* SAFe helps large organization teams to meet an organization's strategic goals, not just individual project goals.
* The framework offers the ability to maintain and create a centralized strategy to deliver value.
* The SAFe model has three/four levels that centralize the strategic themes of an organization.
* Centralized strategy, combined with the de-centralized agile development execution.

**References:**

The official and current websites for reference are:

[http://www.scaledagileframework.com](http://www.scaledagileframework.com/)

This Article is contributed by **Jyothi Rangaraj**

## ****Testing Different Domains****

# Web Application Testing: 8 Step Guide to Website Testing

Before we pen down more details on the type of web testing, lets quickly define Web Testing.

## What is Web Testing?

**WEB TESTING**, or website testing is checking your web application or website for potential bugs before its made live and is accessible to general public. Web Testing checks for functionality, usability, security, compatibility, performance of the web application or website.

During this stage issues such as that of web application security, the functioning of the site, its access to handicapped as well as regular users and its ability to handle traffic is checked.

## How to test Web Application

In Software Engineering, the following testing types/technique may be performed depending on your web testing requirements.

## 1. Functionality Testing:

This is used to check if your product is as per the specifications you intended for it as well as the functional requirements you charted out for it in your developmental documentation. Web based Testing Activities includes:

**Test**all **links**in your webpages are working correctly and make sure there are no broken links. Links to be checked will include -

* Outgoing links
* Internal links
* Anchor Links
* MailTo Links

**Test Forms**are working as expected. This will include-

* Scripting checks on the form are working as expected. For example- if a user does not fill a mandatory field in a form an error message is shown.
* Check default values are being populated
* Once submitted, the data in the forms is submitted to a live database or is linked to a working email address
* Forms are optimally formatted for better readability

**Test Cookies** are working as expected. Cookies are small files used by websites to primarily remember active user sessions so you do not need to log in every time you visit a website. Cookie Testing will include

* Testing cookies (sessions) are deleted either when cache is cleared or when they reach their expiry.
* Delete cookies (sessions) and test that login credentials are asked for when you next visit the site.

**Test HTML and CSS** to ensure that search engines can crawl your site easily. This will include

* Checking for Syntax Errors
* Readable Color Schemas
* Standard Compliance. Ensure standards such W3C, OASIS, IETF, ISO, ECMA, or WS-I are followed.

**Test business workflow**- This will include

* Testing your end - to - end workflow/ business scenarios which takes the user through a series of webpages to complete.
* Test negative scenarios as well, such that when a user executes an unexpected step, appropriate error message or help is shown in your web application.

**Tools that can be used**: [**QTP**](https://www.guru99.com/quick-test-professional-qtp-tutorial.html) , IBM Rational , [**Selenium**](https://www.guru99.com/selenium-tutorial.html)

## ****2. Usability testing****:

[Usability Testing](https://www.guru99.com/usability-testing-tutorial.html) has now become a vital part of any web based project. It can be **carried out by testers** like you **or a small focus group** similar to the target audience of the web application.

**Test**the site **Navigation**:

* Menus, buttons or Links to different pages on your site should be easily visible and consistent on all webpages

**Test**the **Content**:

* Content should be legible with no spelling or grammatical errors.
* Images if present should contain an "alt" text

**Tools that can be used**: Chalkmark, Clicktale, Clixpy and Feedback Army

## ****3.Interface Testing****:

Three areas to be tested here are - Application, Web and Database Server

* **Application:** Test requests are sent correctly to the Database and output at the client side is displayed correctly. Errors if any must be caught by the application and must be only shown to the administrator and not the end user.
* **Web Server**: Test Web server is handling all application requests without any service denial.
* **Database Server:**Make sure queries sent to the database give expected results.

**Test system response** when **connection between the three layers**(Application, Web and Database) **cannot be established** and appropriate message is shown to the end user.

**Tools that can be used**: AlertFox, Ranorex

## 4. Database Testing:

Database is one critical component of your web application and stress must be laid to test it thoroughly. Testing activities will include-

* Test if any errors are shown while executing queries
* Data Integrity is maintained while creating, updating or deleting data in database.
* Check response time of queries and fine tune them if necessary.
* Test data retrieved from your database is shown accurately in your web application

**Tools that can be used**: [**QTP**](https://www.guru99.com/quick-test-professional-qtp-tutorial.html), [**Selenium**](https://www.guru99.com/selenium-tutorial.html)

## 5. Compatibility testing.

Compatibility tests ensures that your web application displays correctly across different devices. This would include-

**Browser Compatibility Test**: Same website in different browsers will display differently. You need to test if your web application is being displayed correctly across browsers, JavaScript, AJAX and authentication is working fine. You may also check for[Mobile](https://www.guru99.com/mobile-testing.html)Browser Compatibility.

The rendering of web elements like buttons, text fields etc. changes with change in **Operating System**. Make sure your website works fine for various combination of Operating systems such as Windows, Linux, Mac and Browsers such as Firefox, Internet Explorer, Safari etc.

**Tools that can be used**: NetMechanic

## 6. Performance Testing:

This will ensure your site works under all loads. Software Testing activities will include but not limited to -

* Website application response times at different connection speeds
* Load test your web application to determine its behavior under normal and peak loads
* Stress test your web site to determine its break point when pushed to beyond normal loads at peak time.
* Test if a crash occurs due to peak load, how does the site recover from such an event
* Make sure optimization techniques like gzip compression, browser and server side cache enabled to reduce load times

**Tools that can be used**: **[Loadrunner](https://www.guru99.com/loadrunner-v12-tutorials.html" \o "Loadrunner)**, [**JMeter**](https://www.guru99.com/jmeter-tutorials.html)

## 7. Security testing:

[Security Testing](https://www.guru99.com/what-is-security-testing.html) is vital for e-commerce website that store sensitive customer information like credit cards. Testing Activities will include-

* Test unauthorized access to secure pages should not be permitted
* Restricted files should not be downloadable without appropriate access
* Check sessions are automatically killed after prolonged user inactivity
* On use of SSL certificates, website should re-direct to encrypted SSL pages.

**Tools that can be used**: Babel Enterprise, BFBTester and CROSS

## 8. Crowd Testing:

You will select a large number of people (crowd) to execute tests which otherwise would have been executed a select group of people in the company. Crowdsourced testing is an interesting and upcoming concept and helps unravel many a unnoticed defects.

**Tools that can be used**: People like you and me !!!. And yes , loads of them!

This concludes the tutorial. It includes almost all testing types applicable to your web application.

As a Web-tester its important to note that web testing is quite an arduous process and you are bound to come across many obstacles. One of the major problems you will face is of course **deadline pressure**. Everything is always needed yesterday! The number of times the **code will need changing**is also taxing. Make sure you **plan your work** and know clearly what is expected of you. Its best **define all the tasks** involved in your web testing and then **create a work chart for accurate estimates and planning**.

Top of Form

|  |  |  |
| --- | --- | --- |
| |  | | --- | |  | |  |

Bottom of Form

# Web Application Testing Checklist: Example Test Cases for Website

While testing the web applications, one should consider the below mentioned template. The below mentioned checklist is almost applicable for all types of web applications depending on the business requirements.

The web application testing checklist consists of-

* [Usability Testing](https://www.guru99.com/complete-web-application-testing-checklist.html#1)
* [Functional Testing](https://www.guru99.com/complete-web-application-testing-checklist.html#2)
* [Compatibility Testing](https://www.guru99.com/complete-web-application-testing-checklist.html#3)
* [Database Testing](https://www.guru99.com/complete-web-application-testing-checklist.html#4)
* [Security Testing](https://www.guru99.com/complete-web-application-testing-checklist.html#5)
* [Performance Testing](https://www.guru99.com/complete-web-application-testing-checklist.html#6)

Now let's look each checklist in detail:

## Usability Testing

**What is Usability Testing?**

* Usability testing is nothing but the User-friendliness check.
* In Usability testing, the application flow is tested so that a new user can understand the application easily.
* Basically, system navigation is checked in Usability testing.

**What is the purpose or Goal of Usability testing?**

A Usability test establishes the ease of use and effectiveness of a product using a standard Usability test practices.

**Example Usability Test Cases**

* Web page content should be correct without any spelling or grammatical errors
* All fonts should be same as per the requirements.
* All the text should be properly aligned.
* All the error messages should be correct without any spelling or grammatical errors and the error message should match with the field label.
* Tool tip text should be there for every field.
* All the fields should be properly aligned.
* Enough space should be provided between field labels, columns, rows, and error messages.
* All the buttons should be in a standard format and size.
* Home link should be there on every single page.
* Disabled fields should be grayed out.
* Check for broken links and images.
* Confirmation message should be displayed for any kind of update and delete operation.
* Check the site on different resolutions (640 x 480, 600x800 etc.?)
* Check the end user can run the system without frustration.
* Check the tab should work properly.
* Scroll bar should appear only if required.
* If there is an error message on submit, the information filled by the user should be there.
* Title should display on each web page
* All fields (Textbox, dropdown, radio button, etc) and buttons should be accessible by keyboard shortcuts and the user should be able to perform all operations by using keyboard.
* Check if the dropdown data is not truncated due to the field size. Also, check whether the data is hardcoded or managed via administrator.

## Functional Testing:

**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.

**What is the purpose or Goal of Functional testing?**

* The goal of[Functional Testing](https://www.guru99.com/functional-testing.html) is to verify whether your product meets the intended functional specifications mentioned in your development documentation.

**Example Functional Test Scenarios:**

* Test all the mandatory fields should be validated.
* Test the asterisk sign should display for all the mandatory fields.
* Test the system should not display the error message for optional fields.
* Test that leap years are validated correctly & do not cause errors/miscalculations.
* Test the numeric fields should not accept the alphabets and proper error message should display.
* Test for negative numbers if allowed for numeric fields.
* Test division by zero should be handled properly for calculations.
* Test the max length of every field to ensure the data is not truncated.
* Test the pop up message ("This field is limited to 500 characters") should display if the data reaches the maximum size of the field.
* Test that a confirmation message should display for update and delete operations.
* Test the amount values should display in currency format.
* Test all input fields for special characters.
* Test the timeout functionality.
* Test the Sorting functionality.
* Test the functionality of the buttons available
* Test the Privacy Policy & FAQ is clearly defined and should be available for users.
* Test if any functionality fails the user gets redirected to the custom error page.
* Test all the uploaded documents are opened properly.
* Test the user should be able to download the uploaded files.
* Test the email functionality of the system.
* Test the[Java](https://www.guru99.com/java-tutorial.html)script is properly working in different browsers (IE, Firefox, Chrome, safari and Opera).
* Test to see what happens if a user deletes cookies while in the site.
* Test to see what happens if a user deletes cookies after visiting a site.
* Test all the data inside combo/list box is arranged in chronological order.

## Compatibility Testing:

**What is Compatibility testing?**

* Compatibility testing is used to determine if your software is compatible with other elements of a system with which it should operate, e.g. Browsers, Operating Systems, or hardware.

**What is the purpose or Goal of Compatibility testing?**

* The purpose of Compatibility testing is to evaluate how well software performs in a particular browser, Operating Systems, hardware or software.

**Sample Compatibility Test Scenarios:**

* Test the website in different browsers (IE, Firefox, Chrome, Safari and Opera) and ensure the website is displaying properly.
* Test the HTML version being used is compatible with appropriate browser versions.
* Test the images display correctly in different browsers.
* Test the fonts are usable in different browsers.
* Test the java script code is usable in different browsers.
* Test the Animated GIF's across different browsers.

## Database Testing:

**What is Database Testing?**

* In Database testing backend records are tested which have been inserted through the web or desktop applications. The data which is displaying in the web application should match with the data stored in the Database.

**To perform the Database testing, the tester should be aware of the below mentioned points**:

* The tester should understand the functional requirements, business logic, application flow and database design thoroughly.
* The tester should figure out the tables, triggers, store procedures, views and cursors used for the application.
* The tester should understand the logic of the triggers, store procedures, views and cursors created.
* The tester should figure out the tables which get affected when insert update and delete (DML) operations are performed through the web or desktop applications.

**With the help of the above mentioned points, the tester can easily write the test scenarios for Database testing.**

**Example Test Cases for Database Testing:**

* Verify the database name: The database name should match with the specifications.
* Verify the Tables, columns, column types and defaults: All things should match with the specifications.
* Verify whether the column allows a null or not.
* Verify the Primary and foreign key of each table.
* Verify the Stored Procedure:
* Test whether the Stored procedure is installed or not.
* Verify the Stored procedure name
* Verify the parameter names, types and number of parameters.
* Test the parameters if they are required or not.
* Test the stored procedure by deleting some parameters
* Test when the output is zero, the zero records should be affected.
* Test the stored procedure by writing simple[SQL](https://www.guru99.com/sql.html)queries.
* Test whether the stored procedure returns the values
* Test the stored procedure with sample input data.
* Verify the behavior of each flag in the table.
* Verify the data gets properly saved into the database after each page submission.
* Verify the data if the DML (Update, delete and insert) operations are performed.
* Check the length of every field: The field length in the back end and front end must be same.
* Verify the database names of QA, UAT and production. The names should be unique.
* Verify the encrypted data in the database.
* Verify the database size. Also test the response time of each query executed.
* Verify the data displayed on the front end and make sure it is same in the back end.
* Verify the data validity by inserting the invalid data in the database.
* Verify the Triggers.

## What is Security Testing?

Security Testing involves the test to identify any flaws and gaps from a security point of view.

**Sample Test Scenarios for Security Testing:**

* Verify the web page which contains important data like password, credit card numbers, secret answers for security question etc should be submitted via HTTPS (SSL).
* Verify the important information like password, credit card numbers etc should display in encrypted format.
* Verify password rules are implemented on all authentication pages like Registration, forgot password, change password.
* Verify if the password is changed the user should not be able to login with the old password.
* Verify the error messages should not display any important information.
* Verify if the user is logged out from the system or user session was expired, the user should not be able to navigate the site.
* Verify to access the secured and non-secured web pages directly without login.
* Verify the “View Source code” option is disabled and should not be visible to the user.
* Verify the user account gets locked out if the user is entering the wrong password several times.
* Verify the cookies should not store passwords.
* Verify if, any functionality is not working, the system should not display any application, server, or database information. Instead, it should display the custom error page.
* Verify the SQL injection attacks.
* Verify the user roles and their rights. For Example, the requestor should not be able to access the admin page.
* Verify the important operations are written in log files, and that information should be traceable.
* Verify the session values are in an encrypted format in the address bar.
* Verify the cookie information is stored in encrypted format.
* Verify the application for Brute Force Attacks

## What is Performance Testing?

Performance Testing is conducted to evaluate the compliance of a system or component with specified performance requirements.

**General Test scenarios:**

* To determine the performance, stability and scalability of an application under different load conditions.
* To determine if the current architecture can support the application at peak user levels.
* To determine which configuration sizing provides the best performance level.
* To identify application and infrastructure bottlenecks.
* To determine if the new version of the software adversely had an impact on response time.
* To evaluate product and/or hardware to determine if it can handle projected load volumes.

**How to do Performance testing? By Manual Testing or by Automation**

Practically it is not possible to do the Performance Testing manually because of some drawbacks like:

* More number of resources will be required.
* Simultaneous actions are not possible.
* Proper system monitoring is not available.
* Not easy to perform the repetitive task.

Hence to overcome the above problems we should use Performance Testing tool. Below is the list of some popular testing tools.

* Apache JMeter
* Load Runner
* Borland Silk Performer.
* Rational Performance Tester
* WAPT
* NEO LOAD

# Testing Telecom Domain with Sample OSS/BSS Test cases

### What is Telecom Testing?

Telecom Testing is defined as the testing of Telecommunication software. Since the shift of the telecom sector to digital and computer networks, telecommunication industry uses software indispensable. Telecom sector depends on the various types of software components to deliver many services like routing and switching, VoIP broadband access, etc. Hence, telecom software testing is inevitable.

In this tutorial, you will learn-

* [What is Domain in Testing?](https://www.guru99.com/testing-telecom-application-with-sample-testcases.html#1)
* [Why Domain Knowledge Matters?](https://www.guru99.com/testing-telecom-application-with-sample-testcases.html#2)
* [Business Processes in Telecom Industry](https://www.guru99.com/testing-telecom-application-with-sample-testcases.html#3)
* [Typical Telecom Business Process](https://www.guru99.com/testing-telecom-application-with-sample-testcases.html#4)
* [Types of Protocols used in Telecom Industry](https://www.guru99.com/testing-telecom-application-with-sample-testcases.html#5)
* [Testing LifeCycle in the Telecom Industry](https://www.guru99.com/testing-telecom-application-with-sample-testcases.html#6)
* [Types of Testing Performed on Telecom Software](https://www.guru99.com/testing-telecom-application-with-sample-testcases.html#7)
* [Sample TestCases for Telecom Testing](https://www.guru99.com/testing-telecom-application-with-sample-testcases.html#8)

[**Join our Live Telecom Testing Project for Free**](https://www.guru99.com/live-telecom-project.html)

## What is Domain in Testing?

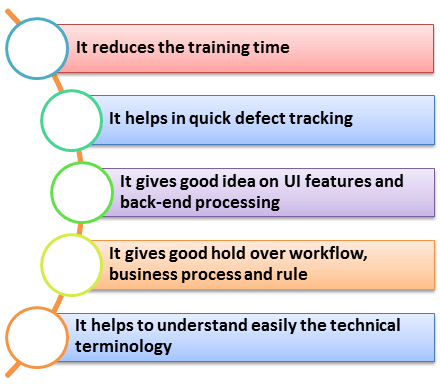
A domain is nothing but the industry for which the software testing project is created. When we talk about software project or development, this term is often referred. For example, the Insurance domain, Banking domain, Retail Domain, Telecom Domain, etc.

[](https://www.guru99.com/images/6-2015/052615_1201_TestingTele1.png)

Usually while developing any specific domain project, domain expert help is sought out. Domain expert is master of the subject and he may know the inside-out of the product or application.

## Why Testing Domain Knowledge Matters?

Domain knowledge is quintessential for testing any software product, and it has its own benefits like

[](https://www.guru99.com/images/6-2015/052615_1201_TestingTele2.png)

## Business Processes in the Telecom Industry

For telecom testing end-to-end service verification is important. To ensure efficient testing a good understanding of the different Business process is a must.

You need to understand each stage of service deliverability before drafting the test cases.

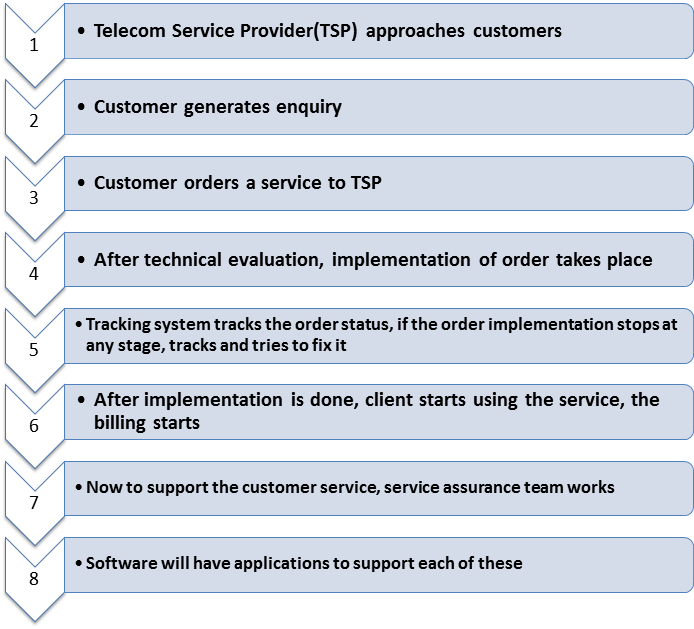
Telecom services are either based on a business support system that includes IVR's, Call Centers, generating invoices, etc. or an operation support system that includes routers, switches, cell towers, etc.

The following table shows what activities are performed at different levels

|  |  |
| --- | --- |
| **Telecom Department** | **Telecom Activities** |
| Pre-sales | * It handles all the sales information like discounts, services, promos, etc. |
| Ordering | * Applying for a new connection or disconnecting a connection |
| Provisioning | * This division deals with the physical connection between customers and TSP ( Telecom Service Provider) |
| Billing | * Under this division, all billing work is done |
| Service Assurance | * In case of any failure, this division corrects the problem |
| Inventory Systems | * It is the repository of all information |
| Tracking | * This division tracks the ordering system and the status of an order |

## Typical Telecom Business Process

Following is a typical business process in the Telecom Industry.

[](https://www.guru99.com/images/6-2015/052615_1201_TestingTele3.png)

## Types of Protocols used in Telecom Industry

Here the popular protocols used in the Telecom industry

* **VoIP technologies**: VoIP, IMS, MPLS, ISDN, PSTN
* **Signaling and Protocols**: SIP, ISDN, Codecs, H.323
* **Wireless technologies:**GPRS, CDMA, GSM, UMTS
* **Network Management:**SNMP
* **Layer 2 Protocols:**ARP, STP, L2TP, PPP
* **Layer 3 protocols/routing:**ICMP, BGP, ISIS, MPLS
* **Infrastructure/Security:**ATM, TCP/IP, LAN/VLAN, SSH

**You can learn more about Protocol Testing**[**here**](https://www.guru99.com/protocol-testing.html)

## Testing LifeCycle in the Telecom Industry

The Test Lifecycle in the telecom industry is similar to that of any other industry but with a stress on details. Here is how the test lifecycle looks like along with the test artifacts.

|  |  |
| --- | --- |
| **Telecom Testing Stage** | **Test artifacts** |
| * Business View | * Requirement based test artifacts * Feasibility based artifacts * Standard and policy identification based test artifacts * Operation and maintenance considerations related test artifacts |
| * System/ Architecture | * System test artifacts (Security, Installation) * Test artifacts for virtual prototype * Special [System Testing](https://www.guru99.com/system-testing.html) artifacts ( interoperability, disaster recovery) |
| * Implementation | * Unit test artifacts * Integration test artifacts * Quality and performance artifacts * Regression, load testing, sanity, etc. |
| * Deployment | * Acceptance test artifacts * Integration test artifacts * Quality and performance artifacts * Functional test artifacts * Alpha/Beta test artifacts |

## Types of Testing Performed on Telecom Software

* Interconnection Testing
* Conformance Testing
* IVR Testing
* [Performance Testing](https://www.guru99.com/performance-testing.html)
* [Security Testing](https://www.guru99.com/what-is-security-testing.html)
* Interoperability Testing
* Protocol Testing
* Functional Testing
* [Automation Testing](https://www.guru99.com/automation-testing.html)

## Sample TestCases for Telecom Testing

In Telecom Testing, one must consider the testing following

|  |  |
| --- | --- |
| **Various Telecom Testing** | **Testing activities in Telecom** |
| **Billing System** | * Verify, the telephone number of the customer is registered under telecom operator * Verify whether the number is still working * Verify the number entered is valid, and it is 10 digit number * Verify the number is not blocked due to some reasons * Verify if the number has any outstanding bills, if exist, display it on screen * Verify the number has all previous accounts or bills cleared * Verify the system enables statement generation as per customer requirement * Verify the system has recorded a number of calls accurately * Verify the plan chosen by the customer displays on the billing system * Verify the total amount billed is accurate and mapped to the service offered |
| **Application Testing** | * Protocols, signaling, field testing for IOT * Usage and[Functional Testing](https://www.guru99.com/functional-testing.html) for core[Mobile](https://www.guru99.com/mobile-testing.html)handset applications like a call, SMS, transfer/hold, etc. * Testing of various applications like finance, sports and location-based services, etc. OSS-BSS testing |
| **OSS-BSS Testing** | * Billing, customer case, interconnect billing, order and fraud management, revenue assurance * Network management, mediation, provisioning, etc. * EAI, CRM & ERP, data warehousing, etc. |
| **Conformance Testing** | * Electrical interface compatibility * Conformance of protocol * Conformance of transport layers |
| **IVR Testing** | * Interactive test scenarios * Detection of voice energy * Broadband audio tones * Extensive conditional branching sequences * DTMF Entries |

**Summary**

The telecom service is a very broad field consists of a various component including cables, networks, signals, protocols, etc. and their testing requires a broad range of testing techniques, so the choice of testing techniques and strategy highly depends on what component of telecom is tested.

The test requirement, scope, test scenarios, testing techniques, testing tools, etc. varies with the type of testing involves, it can be protocol testing for VoIP or wireless device testing for CDMA. The tutorial gives a basic but complete overview of how telecom testing can be performed and discuss various prospects that are crucial for telecom testing.

# eCommerce Testing: How to Test an E-Commerce Website

### What is Ecommerce Testing?

eCommerce testing is defined as testing of an eCommerce (online shopping) application. It helps in the prevention of errors and adds value to the product by ensuring conformity to client requirements.

The objective of testing is to ensure

* Software reliability
* Software quality
* System Assurance
* Optimum performance and capacity utilization

Setting up an E-commerce system is a complex process and subject to many market-specific variables. To maintain the integrity of the E Commerce system, testing becomes compulsory

[**Join our Live Ecommerce Project for Free**](https://www.guru99.com/live-ecommerce-project.html)

In this tutorial, you will learn,

* [Types of Testing for E-commerce System](https://www.guru99.com/testing-e-commerce-applications.html#1)
* [Performance testing- a top priority in E-commerce](https://www.guru99.com/testing-e-commerce-applications.html#2)
* [Useful Tools for Mapping E-commerce Site](https://www.guru99.com/testing-e-commerce-applications.html#3)
* [Challenges of E-commerce Testing](https://www.guru99.com/testing-e-commerce-applications.html#4)

[](https://www.guru99.com/images/1/052515_0958_TestingEcom1.png)

E-Commerce Domain Knowledge is important for testing.

## Types of Testing for E-commerce System

A common type of testing included into e commerce system is

|  |  |  |
| --- | --- | --- |
| **Sr.#** | **Type of Testing** | **Testing Process** |
| 1 | Browser compatibility | * Lack of support for early browsers * Browser specific extensions * Browser testing should cover the main platforms (Linux, Windows, Mac etc.) |
| 2 | Page display | * Incorrect display of pages * Runtime error messages * Poor page download time * Dead hyperlink, plugin dependency, font sizing, etc. |
| 3 | Session Management | * Session Expiration * Session storage |
| 4 | Usability | * Non-intuitive design * Poor site navigation * Catalog navigation * Lack of help-support |
| 5 | Content Analysis | * Misleading, offensive and litigious content * Royalty free images and copyright infringement * Personalization functionality * Availability 24/7 |
| 6 | Availability | * Denial of service attacks * Unacceptable levels of unavailability |
| 7 | Back-up and Recovery | * Failure or fall over recovery * Backup failure * Fault tolerance |
| 8 | Transactions | * Transaction Integrity * Throughput * Auditing |
| 9 | Shopping order processing and purchasing | * Shopping cart functionality * Order processing * Payment processing * Order tracking |
| 10 | Internationalization | * Language support * Language display * Cultural sensitivity * Regional Accounting |
| 11 | Operational business procedures | * How well e-procedure copes * Observe for bottlenecks |
| 12 | System Integration | * Data Interface format * Interface frequency and activation * Updates * Interface volume capacity * Integrated performance |
| 13 | Performance | * Performance bottlenecks * Load handling * Scalability analysis |
| 14 | Login and Security | * Login capability * Penetration and access control * Insecure information transmission * Web attacks * Computer viruses * Digital signatures |

## Performance testing- a top priority in E-commerce

Just delay about 250 milliseconds of a page load time, is what keeps your customer going to your competitor. Retail giant Walmart overhaul their site speed and noticed an increase of 2% in visitor's conversion rate and revenue by 1%.

Performance of your site depends on these factors

* **Throughput**
  + Request per second
  + Transactions per minute
  + Executions per click
* **Response Time**
  + Duration of a task
  + Seconds per click
  + Page Load
  + DNS Lookup
  + Length of time between click and seeing page

## Useful Tools for Mapping E-commerce Site

* [**UsabilityHub**](https://usabilityhub.com/): UsabilityHub's user testing platform and research panel help you improve the UX of your apps and websites. Get feedback from real people.
* [**HotJar**](https://www.hotjar.com/): It shows the most clicked and unclicked zones of sites by visitors
* [**FiveSecondTest**](http://fivesecondtest.com/): This tool ensures that your message is communicated as effectively as possible, in just five seconds it tells what a person recalls about your website design
* [**Feng-GUI**](https://www.feng-gui.com/): It simulates the human vision during the first five seconds and predicts what a real human would most likely look at
* [**Optimizely**](https://www.optimizely.com/): It enables you to test track, clicks, conversions or anything else that matters to e-commerce business

## Challenges of E-commerce Testing

* Compliance with security guidelines to safeguard customer data and identity
* Compliance with accessibility standards to support multi-lingual markets and business regions
* End to end testing and test management for large e-commerce transformation programs
* Scalability and reliability of applications

# Banking Domain Application Testing: Sample Test Cases

The BFSI (Banking, Financial services and Insurance) sector is the biggest consumer of IT services. Banking Applications directly deal with confidential financial data. It is mandatory that all the activities performed by banking software run smoothly and without any error. Banking software perform various functions like transferring and depositing fund, balance inquiry, transaction history, withdrawal and so on. Testing banking application assures that these activities are not only executed well but also remain protected from hackers.

In this tutorial, we will learn

* [What is Domain in Testing?](https://www.guru99.com/banking-application-testing.html#1)
* [Why Domain Knowledge Matters?](https://www.guru99.com/banking-application-testing.html#2)
* [Introduction to Banking Domain](https://www.guru99.com/banking-application-testing.html#3)
* [Characteristics of a banking application](https://www.guru99.com/banking-application-testing.html#4)
* [Stages of testing banking applications](https://www.guru99.com/banking-application-testing.html#5)
* [Sample Test Case for Net Banking Login Application](https://www.guru99.com/banking-application-testing.html#6)
* [Challenges in testing banking domain & their Mitigation](https://www.guru99.com/banking-application-testing.html#7)

[**Join our Live Banking Testing Project for Free**](https://www.guru99.com/live-testing-project.html)

## What is Domain in Testing?

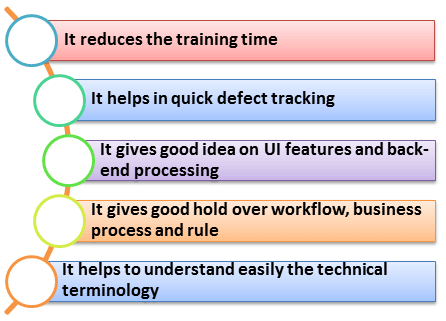
Domain is nothing but the industry for which the software testing project is created. When we talk about software project or development, this term is often referred. For example, Insurance domain, Banking domain, Retail Domain, Telecom Domain, etc.

[](https://www.guru99.com/images/6-2015/052515_0721_BankingAppl1.png)

Usually, while developing any specific domain project, domain expert help is sought out. Domain expert are master of the subject, and he may know the inside-out of the product or application.

## Why Domain Knowledge Matters?

Domain knowledge is quintessential for testing any software product, and it has its own benefits like

[](https://www.guru99.com/images/6-2015/052515_0721_BankingAppl2.png)

## Banking Domain Knowledge - Introduction

Banking domain concepts are huge, and basically it is sub-characterized into two sectors

1. **Traditional banking sector**
2. **Service based banking sector**

Below is the table of the services these two sub-sectors of banking encompass

|  |  |
| --- | --- |
| **Traditional banking sector** | * Core banking * Corporate banking * Retail banking |
| **Service based banking sector** | * Core * Corporate * Retail * Loan * Trade finance * Private banking * Consumer finance * Islamic banking * Customer delivery channels/Front end delivery |

Based on the scope of your project you may need to test one or all of the above service offerings. Before you begin testing, ensure you have enough background on the service being tested.

## Characteristics of a Banking Application

Before you begin testing, it's important to note the standard features expected of any banking application. So that, you can gear your test efforts to achieve these characteristics.

A standard banking application should meet all these characteristics as mentioned below.

* It should support thousands of concurrent user sessions
* A banking application should integrate with other numerous applications like trading accounts, Bill pay utility, credit cards, etc.
* It should process fast and secure transactions
* It should include massive storage system.
* To troubleshoot customer issues, it should have high auditing capability
* It should handle complex business workflows
* Need to support users on multiple platforms (Mac, Linux, Unix, Windows)
* It should support users from multiple locations
* It should support multi-lingual users
* It should support users on various payment systems (VISA, AMEX, MasterCard)
* It should support multiple service sectors (Loans, Retail banking etc.)
* Foolproof disaster management mechanism

## Test Phases in Testing Banking Applications

For testing banking applications, different stages of testing include

* **Requirement Analysis:** It is done by business analyst; requirements for a particular banking application are gathered and documented
* **Requirement Review:**Quality analysts, business analysts, and development leads are involved in this task. The requirement gathering document is reviewed at this stage, and cross-checked to ensure that it does not affect the workflow
* **Business Requirements Documentation:**Business requirements documents are prepared by quality analysts in which all reviewed business requirements are covered
* **Database Testing:**It is the most important part of bank application testing. This testing is done to ensure data integrity, data loading, data migration, stored procedures, and functions validation, rules testing, etc.
* **Integration Testing:**Under [Integration Testing](https://www.guru99.com/integration-testing.html) all components that are developed are integrated and validated
* **Functional Testing:**The usual software testing activities like[Test Case](https://www.guru99.com/test-case.html)preparation, test case review and test case execution is done during this phase
* **Security Testing:**It ensures that the software does not have any security flaws. During test preparation, QA team needs to include both negative as well as positive test scenarios so as to break into the system and report it before any unauthorized individual access it. While to prevent from hacking, the bank should also implement a multi-layer of access validation like a one-time password. For [Security Testing](https://www.guru99.com/what-is-security-testing.html), automation tools like IBM AppScan and HPWebInspect are used while for [Manual Testing](https://www.guru99.com/manual-testing.html) tools like Proxy Sniffer, Paros proxy, HTTP watch, etc. are used
* **Usability Testing:**It ensures that differently able people should be able to use the system as normal user. For example, ATM with hearing and Braille facility for disabled
* **User Acceptance Testing:**It is the final stage of testing done by the end users to ensure the compliance of the application with the real world scenario.

## Sample Test Case for Net Banking Login Application

Security is prime for any banking application. Therefore, during test preparation, QA team should include both negative and positive test scenarios in order to sneak into the system and report for any vulnerabilities before any unauthorized individual get access to it. It not only involves writing negative test cases but may also include destructive testing.

Following are generic test cases to check any banking application

|  |  |
| --- | --- |
| **Sample test cases** |  |
| For Admin | * Verify Admin login with valid and Invalid data * Verify admin login without data * Verify all admin home links * Verify admin change password with valid and invalid data * Verify admin change password without data * Verify admin change password with existing data * Verify admin logout |
| For new Branch | * Create a new branch with valid and invalid data * Create a new branch without data * Create a new branch with existing branch data * Verify reset and cancel option * Update branch with valid and invalid data * Update branch without data * Update branch with existing branch data * Verify cancel option * Verify branch deletion with and without dependencies * Verify branch search option |
| For New Role | * Create a new role with valid and invalid data * Create a new role without data * Verify new role with existing data * verify role description and role types * Verify cancel and reset option * Verify role deletion with and without dependency * verify links in role details page |
| For customer & Visitors | * Verify all visitor or customer links * Verify customers login with valid and invalid data * Verify customers login without data * Verify banker's login without data * Verify banker's login with valid or invalid data |
| For New users | * Create a new user with valid and invalid data * Create a new user without data * Create a new user with existing branch data * Verify cancel and reset option * Update user with valid and invalid data * Update user with existing data * Verify cancel option * Verify deletion of the user |

## Challenges in testing Banking domain & their Mitigation

Challenges tester might face during testing banking domain are

|  |  |
| --- | --- |
| **Challenge** | **Mitigation** |
| * Getting access to production data and replicating it as test data, for testing is challenging | * Ensure that test data meets regulatory compliances requirements and guidelines * Maintain the data confidentiality by following techniques like data masking, synthetic test data, testing system integration, etc. |
| * The biggest challenge in testing banking system is during the migration of the system from the old system to the new system like testing of all the routines, procedures and plans. Also how the data will be fetched, uploaded and transferred to the new system after migration | * Ensure Data Migration Testing is complete * Ensure Regression Test cases are executed on old and new systems, and the results match. |
| * There may be the cases where requirements are not documented well and may lead to functional gaps in test plan * Many non-functional requirements are not fully documented, and testers do not know whether to test it or not | * The test should participate in the project right from Requirement Analysis phases and should actively review the Business Requirements |
| * The most important point is to check whether the said system follows the desired policies and procedures | * Compliance or Regulatory Policies testing must be done |
| * The scope and the timelines increases as banking application are integrated with other application like internet or[Mobile](https://www.guru99.com/mobile-testing.html)banking | * Ensure Time budget for Integration Testing is accounted if your banking application has many external interfaces |

## Summary

Banking domain is the most vulnerable area for cyber-theft, and safeguarding the software requires precise testing. This tutorial gives a clear idea of what it takes for banking domain testing and how important it is. One must understand that -

* Majority of banking software are developed on **Mainframe**and**Unix**
* Testing helps to lessen possible glitches encounter during software development
* Proper testing and compliance to industry standards, save companies from penalties
* Good practices help develop good results, reputation and more business for companies
* Both manual and automated testing have respective merits and usability

## T

## est Cycle Closure

Testing team will meet, discuss and analyze testing artifacts to identify strategies that have to be implemented in the future, taking lessons from the current test cycle. The idea is to remove the process bottlenecks for future test cycles and share best practices for any similar projects in the future.

**Activities**

* Evaluate cycle completion criteria based on Time, Test coverage, Cost,Software, Critical Business Objectives, Quality
* Prepare test metrics based on the above parameters.
* Document the learning out of the project
* Prepare Test closure report
* Qualitative and quantitative reporting of quality of the work product to the customer.
* Test result analysis to find out the defect distribution by type and severity.

**Deliverables**

* Test Closure report
* Test metrics

## STLC Phases along with Entry and Exit Criteria

| **STLC Stage** | **Entry Criteria** | **Activity** | **Exit Criteria** | **Deliverables** |
| --- | --- | --- | --- | --- |
| Requirement Analysis | * Requirements Document available (both functional and non functional) * Acceptance criteria defined. * Application architectural document available. | * Analyse business functionality to know the business modules and module specific functionalities. * Identify all transactions in the modules. * Identify all the user profiles. * Gather user interface/ authentication, geographic spread requirements. * Identify types of tests to be performed. * Gather details about testing priorities and focus. * Prepare Requirement[Traceability Matrix](https://www.guru99.com/traceability-matrix.html)(RTM). * Identify test environment details where testing is supposed to be carried out. * Automation feasibility analysis (if required). | * Signed off RTM * Test automation feasibility report signed off by the client | * RTM * Automation feasibility report (if applicable) |
| Test Planning | * Requirements Documents * Requirement Traceability matrix. * Test automation feasibility document. | * Analyze various testing approaches available * Finalize on the best-suited approach * Preparation of test plan/strategy document for various types of testing * Test tool selection * Test effort estimation * Resource planning and determining roles and responsibilities. | * Approved test plan/strategy document. * Effort estimation document signed off. | * Test plan/strategy document. * Effort estimation document. |
| Test case development | * Requirements Documents * RTM and test plan * Automation analysis report | * Create test cases, test design, automation scripts (where applicable) * Review and baseline test cases and scripts * Create test data | * Reviewed and signed test Cases/scripts * Reviewed and signed test data | * Test cases/scripts * Test data |
| Test Environment setup | * System Design and architecture documents are available * Environment set-up plan is available | * Understand the required architecture, environment set-up * Prepare hardware and software development requirement list * Finalize connectivity requirements * Prepare environment setup checklist * Setup test Environment and test data * Perform smoke test on the build * Accept/reject the build depending on smoke test result | * Environment setup is working as per the plan and checklist * Test data setup is complete * Smoke test is successful | * Environment ready with test data set up * Smoke Test Results. |
| Test Execution | * Baselined RTM,[Test Plan](https://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html), Test case/scripts are available * Test environment is ready * Test data set up is done * Unit/Integration test report for the build to be tested is available | * Execute tests as per plan * Document test results, and log defects for failed cases * Update test plans/test cases, if necessary * Map defects to test cases in RTM * Retest the defect fixes * [Regression Testing](https://www.guru99.com/regression-testing.html) of application * Track the defects to closure | * All tests planned are executed * Defects logged and tracked to closure | * Completed RTM with execution status * Test cases updated with results * Defect reports |
| Test Cycle closure | * Testing has been completed * Test results are available * Defect logs are available | * Evaluate cycle completion criteria based on - Time,[Test coverage](https://www.guru99.com/test-coverage-in-software-testing.html), Cost, Software Quality, Critical Business Objectives * Prepare test metrics based on the above parameters. * Document the learning out of the project * Prepare Test closure report * Qualitative and quantitative reporting of quality of the work product to the customer. * Test result analysis to find out the defect distribution by type and severity | Test Closure report signed off by client | * Test Closure report |