Learn Software Testing in 1 Day

By Krishna Rungta

Table Of Content

Section 3: Testing Types

1. Types of Software Testing: 100 Examples of Different Testing Types
2. What is WHITE Box Testing? Techniques, Example, Types & Tools
3. What is BLACK Box Testing? Techniques, Example & Types
4. Unit Testing Tutorial: What is, Types, Tools, EXAMPLE
5. Integration Testing: What is, Types, Top Down & Bottom Up Example
6. What is System Testing? Types & Definition with Example
7. What is Regression Testing? Definition, Test Cases (Example)
8. Sanity Testing Vs Smoke Testing: Introduction & Differences
9. Performance Testing Tutorial: What is, Types, Metrics & Example
10. Load Testing Tutorial: What is? How to? (with Examples)
11. Accessibility Testing Tutorial: What is, Tools & Examples
12. What is STRESS Testing in Software Testing? Tools, Types, Examples
13. What is User Acceptance Testing (UAT)? with Examples
14. Backend Testing Tutorial: What is, Tools & Examples
15. Protocol Testing Tutorial: L2 & L3
16. Web Service Testing: A Beginner's Tutorial
17. API Testing Tutorial: Learn in 10 minutes!

Section 4: Agile Testing

1. What is Agile Testing? Process, Strategy, Test Plan, Life Cycle Example
2. Scrum Testing Methodology Tutorial: What is, Process, Artifacts, Sprint

Section 5: Testing Different Domains

1. Banking Domain Application Testing: Sample Test Cases
2. eCommerce Testing: How to Test an E-Commerce Website
3. Testing Insurance Domain Applications with Sample Test Cases
4. Payment Gateway Testing Tutorial with Example Test Cases
5. Testing Retail Point Of Sale(POS) Systems: Example Test Cases
6. Testing Telecom Domain with Sample OSS/BSS Test cases
7. ETL Testing or Data Warehouse Testing Tutorial
8. Database(Data) Testing Tutorial with Sample TestCases

Section 1- introduction

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 free. It involves execution of a software component or system component to evaluate one or more properties of interest.

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 and Black Box Testing.

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

This tutorial introduces testing software to the audience and justifies it's importance.

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
- Maintenance (Regression and Maintenance)

Testing Category	Types of Testing
Functional Testing	• Unit Testing Integration • Testing Smoke • UAT (User Acceptance Testing) Localization • • Globalization • Interoperability So • on •
Non-Functional Testing	• • Performance • Endurance • Load Volume • Scalability • Usability • So on
Maintenance	• Regression • Maintenance

This is not the complete list as there are more than 150 types of testing types and still adding. Also, note that not all testing types are applicable to all projects but depend on the nature & scope of the project.

7 Software Testing Principles: Learn with Examples

This tutorial introduces the seven basic principles of Software Testing every professional Software tester and QA professional should know.

Background

It is important that you achieve optimum test results while conducting software testing without deviating from the goal. But how you determine that you are following the right strategy for testing? For that, you need to stick to some basic testing principles. Here are the common seven testing principles that are widely practiced in the software industry.

To understand this, consider a scenario where you are moving a file from folder A to Folder B.

Think of all the possible ways you can test this.

Apart from the usual scenarios, you can also test the following conditions

- Trying to move the file when it is Open
- You do not have the security rights to paste the file in Folder B Folder B
- is on a shared drive and storage capacity is full.
- Folder B already has a file with the same name, in fact, the list is endless
- Or suppose you have 15 input fields to test, each having 5 possible values, the number of combinations to be tested would be 5^15

If you were to test the entire possible combinations project

EXECUTION TIME & COSTS would rise exponentially. We need certain principles and strategies to optimize the testing effort

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 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

Myth: "Principles are just for reference. I will not use them in practice ."

This is so very untrue. Test Principles will help you create an effective Test Strategy and draft error catching test cases.

But learning testing principles is just like learning to drive for the first time.

Initially, while you learn to drive, you pay attention to each and everything like gear shifts, speed, clutch handling, etc. But with experience, you just focus on driving the rest comes naturally. Such that you even hold conversations with other passengers in the car.

Same is true for testing principles. Experienced testers have internalized these principles to a level that they apply them even without thinking. Hence the myth that the principles are not used in practice is simply not true.

What is V Model in Software Testing? Learn with SDLC & STLC Example

Before we learn the V model, let's understand -

What is SDLC?

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

Though SDLC uses the term ‘Development’, it does not involve just coding tasks done by developers but also incorporates the tasks

contributed by testers and stakeholders. In

SDLC, test cases are created.

What is STLC?

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

Though STLC uses the term “testing” it does not involve just testers, in some instances, they have to involve developers as well.

In STLC Test cases are executed.

What is the 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 during the SDLC phase. Testing phase in waterfall model starts only after implementation of the system is done.

Testing is done within the SDLC.

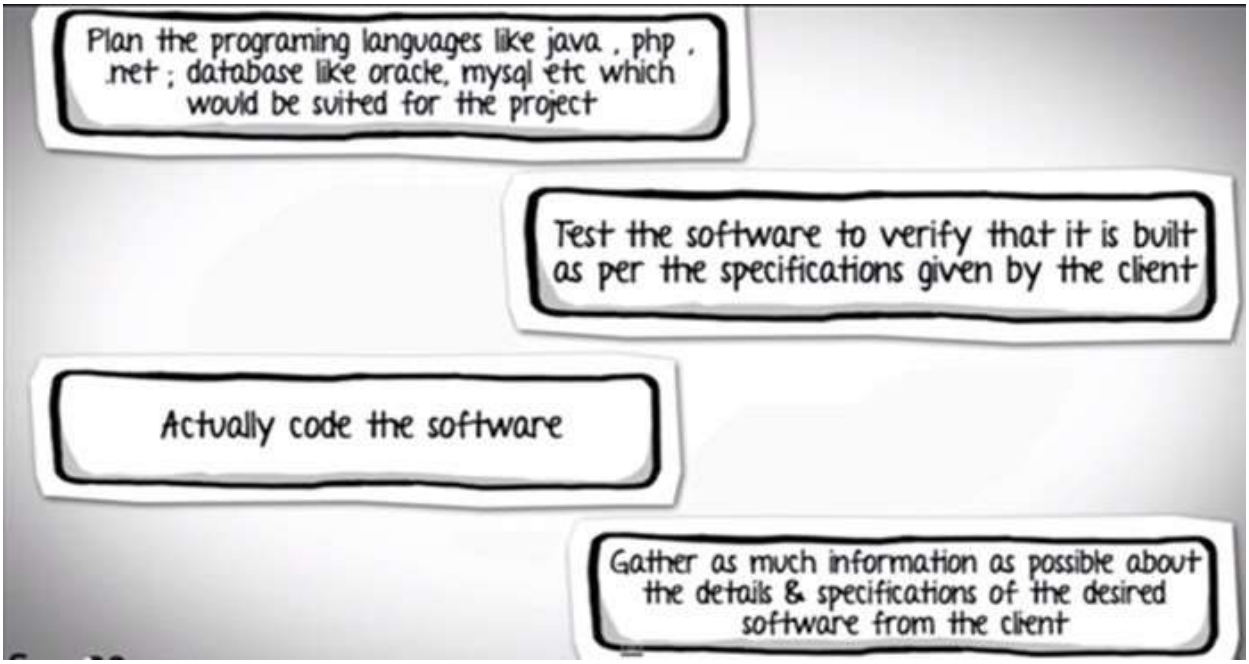
What is V- Model?

V- model is an extension of the waterfall model. It is pronounced as the "vee" model. Unlike the waterfall model, In V-model, there is a corresponding testing phase for each software development phase. Testing in V-model is done in parallel to SDLC stage.

Testing is done as a subproject of SDLC.

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.



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, PHP, .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

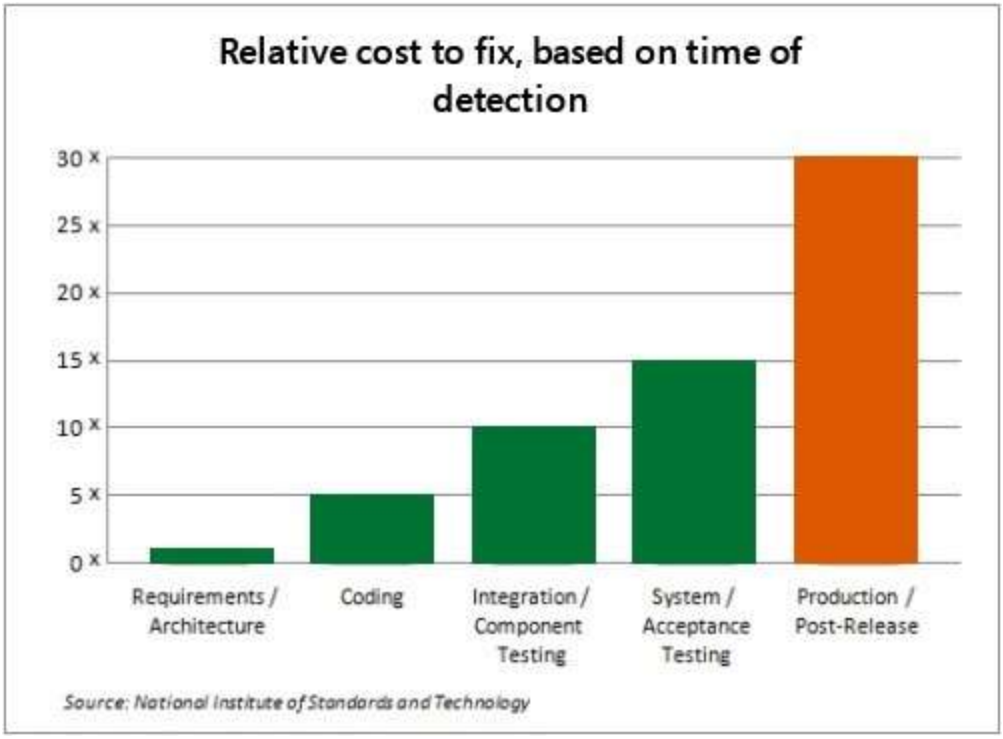
All these levels constitute the **waterfall method** of the software development lifecycle.

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.

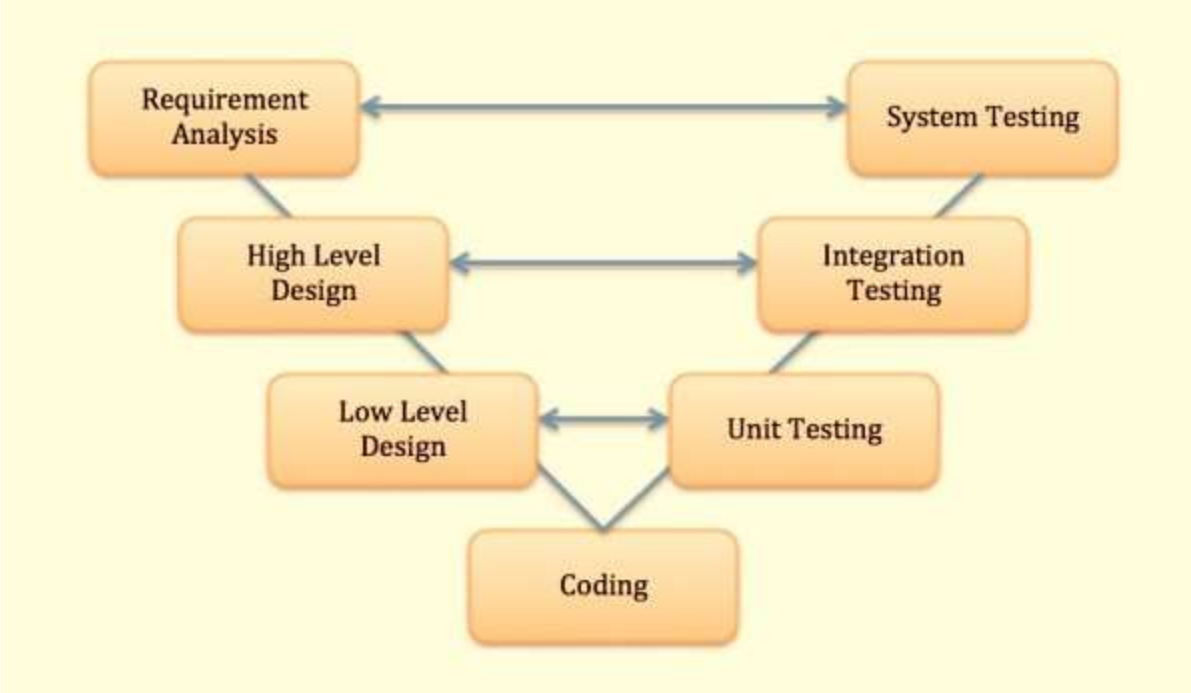
Assessments of thousands of projects have shown that **defects introduced during requirements & design make up close to half of the total number of defects.**



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**



- 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.

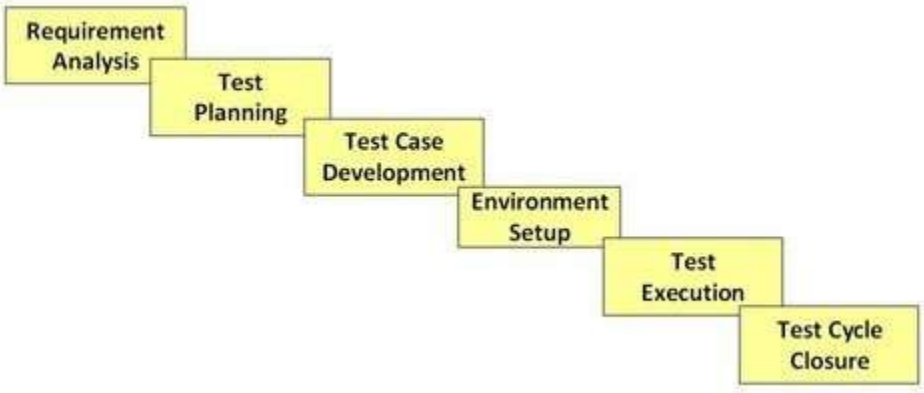
STLC - Software Testing Life Cycle Phases & Entry, Exit Criteria

What is Software Testing Life Cycle (STLC)?

Software Testing Life Cycle (STLC) is defined as a sequence of activities conducted to perform Software Testing.

Contrary to popular belief, Software Testing is not a just a single activity. It consists of a series of activities carried out methodologically to help certify your software product.

Different Phases of the STLC Model



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).
- 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 /strategy document.
- Effort estimation document.

Test Case Development

This phase involves the creation, verification and rework of test cases & test scripts. Test data, 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 Execution

During 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 fixes Track
- the defects to closure

Deliverables

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

Test 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