**STLC**

STLC stands for Software Testing Life Cycle. STLC is a sequence of different activities performed by the testing team to ensure the quality of the software or the product.

* STLC is an integral part of Software Development Life Cycle (SDLC). But, STLC deals only with the testing phases.
* STLC starts as soon as requirements are defined or SRD (Software Requirement Document) is shared by stakeholders.

# Requirement Analysis

Requirement Analysis is the first step of the Software Testing Life Cycle (STLC). In this phase quality assurance team understands the requirements like what is to be tested. If anything is missing or not understandable then the quality assurance team meets with the stakeholders to better understand the detailed knowledge of requirements.

Test Planning

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

# Test Case Development

* Once the Test Plan is ready, the QA Team initiates the development of test cases. The main objective of this phase is to prepare test cases for an individual unit. These functional and structural test cases cover the functionality, points of verification and validation mentioned in the Test Plan.
* A test case is a document, which includes test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement.
* Test Case acts as the starting point for test execution. After the a set of input values is applied; the application has a definitive outcome and leaves the system at some end point which is also known as execution post condition

Test Environment Setup

* Test Environment consists of elements that support test execution with software, hardware and network configured. Test environment configuration must mimic the production environment in order to uncover any environment/configuration related issues.
* Analyze the environment setup requirements and prepare a list of software and hardware requirements for the setup. Get the official confirmation for setup of the test environment and configure to access the test environment.

# Test Execution

Test execution is the process of executing the code and comparing the expected and actual results. Following factors need to be considered for a test execution process −

* Based on a risk, select a subset of test suite to be executed for this cycle.
* Assign the test cases in each test suite to testers for execution.
* Execute tests, report bugs, and capture test status continuously.
* Resolve blocking issues as they arise.
* Report status, adjust assignments, and reconsider plans and priorities daily.
* Report test cycle findings and status.

Test Cycle Closure

* A check against the test exit criteria is very essential to claim that the testing is now complete. Before putting an end to the test process, the product quality is measured against the test completion criteria.
* The entry criteria of this phase is that the execution of the test case is complete, test results are available and the defects report is ready.

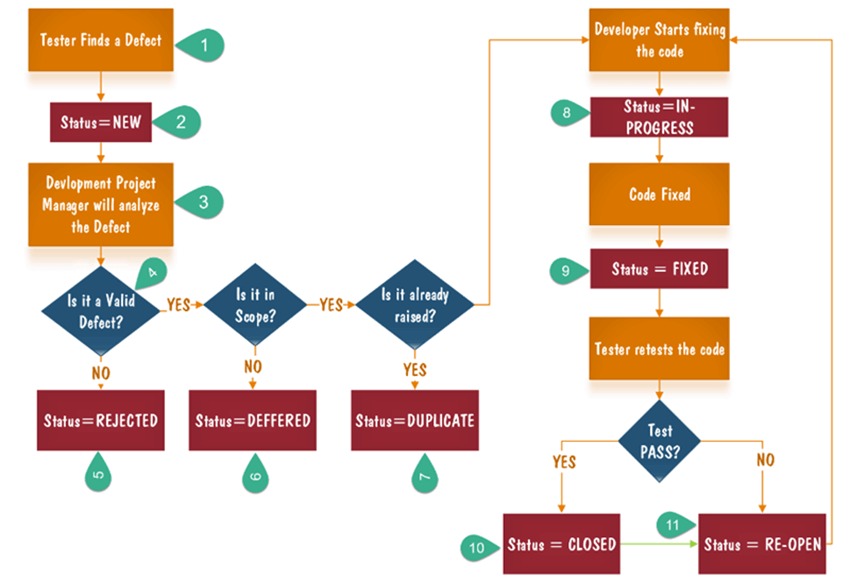
The criteria for test completion includes the following −

* Specified coverage has been achieved.
* No showstoppers or critical defects
* There are very few known medium or low-priority defects. These do not affect the usage of the product.
* The exit criteria of this phase is the provision of test closure reports and preparation of matrices which are later signed off by the client.
* Let us now discuss the activities involved in the closure of Test Cycle.

## Bug in software testing

* The Bug is the informal name of defects, which means that software or application is not working as per the requirement.
* In [software testing](https://www.javatpoint.com/software-testing-tutorial), a software bug can also be issue, error, fault, or failure. The bug occurred when developers made any mistake or error while developing the product.

# Bug Life cycle



* We will talk about the complete life cycle of a bug from the stage it was found, fixed, re-test, and close.
* We have some different status of bugs like new/open, assigned, fix, re-open, and closed.
* As soon as the test engineer finds the bug, status is given as New, which indicates that a bug is just found.
* This new bug needs to be reported to the concerned Developer by changing the status as Assigned so that the responsible person should take care of the bug.
* Then the Developer first go through the bug, which means that the Developers read all the navigation steps to decide whether it is a valid bug or not.
* Based on this, if the bug is valid, the Developer starts reproducing the bug on the application, once the bug is successfully reproduced, the Developer will analyze the code and does the necessary changes, and change the status as Fixed.
* Once the code changes are done, and the bug is fixed, the test engineer re-test the bug, which means that the test engineer performs the same action once again, which is mentioned in the bug report, and changes the status accordingly:
* Close, if the bug fixes properly, and functionally working according to the requirement.
* OR
* Re-open, if the bug still exists or not working properly as per the requirement, then the bug sends it back to the Developer once again.
* This process is going on continuously until all the bugs are fixed and closed.
* When the status of the bug is **Re-open (not fixed)** and affecting another module, then we have to prepare the new bug report.

## Another status of the bug

* Invalid/rejected
* Duplicate
* Postpone/deferred
* Can't fix
* Not reproducible
* RFE (Request for Enhancement)

### Invalid / rejected

When the Test Engineer wrote an incorrect Bug Report because of misunderstanding the requirements, then the Developer will not accept the bug, and gave the status as **Invalid** and sent it back. (Sometime Developer can also misunderstand the requirements).

### Duplicate

When the same bug has been reported multiple times by the different test engineers are known as a **duplicate** bug.

### Not Reproducible

The Developer accepts the bug, but not able to Reproduce due to some reasons.

### Can't fix

When Developer accepting the bug and also able to reproduce, but can't do the necessary code changes due to some constraints.

### Deferred / postponed

The deferred/postpone is a status in which the bugs are postponed to the future release due to time constraints.

### RFE (Request for Enhancement)

These are the suggestions given by the test engineer towards the enhancement of the application in the form of a bug report. The RFE stands for **Request for Enhancement**.

# Software Testing Principles

Software testing is a procedure of implementing software or the application to identify the defects or bugs. For testing an application or software, we need to follow some principles to make our product defects free, and that also helps the test engineers to test the software with their effort and time. Here, in this section, we are going to learn about the seven essential principles of software testing.

* Testing shows the presence of defects
* Exhaustive Testing is not possible
* Early Testing
* Defect Clustering
* Pesticide Paradox
* Testing is context-dependent
* Absence of errors fallacy

### Testing shows the presence of defects

The test engineer will test the application to make sure that the application is bug or defects free. While doing testing, we can only identify that the application or software has any errors. The primary purpose of doing testing is to identify the numbers of unknown bugs with the help of various methods and testing techniques because the entire test should be traceable to the customer requirement, which means that to find any defects that might cause the product failure to meet the client's needs.

By doing testing on any application, we can decrease the number of bugs, which does not mean that the application is defect-free because sometimes the software seems to be bug-free while performing multiple types of testing on it. But at the time of deployment in the production server, if the end-user encounters those bugs which are not found in the testing process

### Exhaustive Testing is not possible

Sometimes it seems to be very hard to test all the modules and their features with effective and non- effective combinations of the inputs data throughout the actual testing process.

Hence, instead of performing the exhaustive testing as it takes boundless determinations and most of the hard work is unsuccessful. So we can complete this type of variations according to the importance of the modules because the product timelines will not permit us to perform such type of testing scenarios.

### Early Testing

Here early testing means that all the testing activities should start in the early stages of the software development life cycle's **requirement analysis stage** to identify the defects because if we find the bugs at an early stage, it will be fixed in the initial stage itself, which may cost us very less as compared to those which are identified in the future phase of the testing process.

To perform testing, we will require the requirement specification documents; therefore, if the requirements are defined incorrectly, then it can be fixed directly rather than fixing them in another stage, which could be the development phase.

### Defect clustering

The defect clustering defined that throughout the testing process, we can detect the numbers of bugs which are correlated to a small number of modules. We have various reasons for this, such as the modules could be complicated; the coding part may be complex, and so on.

These types of software or the application will follow the **Pareto Principle**, which states that we can identify that approx. Eighty percent of the complication is present in 20 percent of the modules. With the help of this, we can find the uncertain modules, but this method has its difficulties if the same tests are performing regularly, hence the same test will not able to identify the new defects.

### Pesticide paradox

This principle defined that if we are executing the same set of test cases again and again over a particular time, then these kinds of the test will not be able to find the new bugs in the software or the application. To get over these pesticide paradoxes, it is very significant to review all the test cases frequently. And the new and different tests are necessary to be written for the implementation of multiple parts of the application or the software, which helps us to find more bugs.

### Testing is context-dependent

Testing is a context-dependent principle states that we have multiple fields such as e-commerce websites, commercial websites, and so on are available in the market. There is a definite way to test the commercial site as well as the e-commerce websites because every application has its own needs, features, and functionality. To check this type of application, we will take the help of various kinds of testing, different technique, approaches, and multiple methods. Therefore, the testing depends on the context of the application.

### Absence of errors fallacy

Once the application is completely tested and there are no bugs identified before the release, so we can say that the application is 99 percent bug-free. But there is the chance when the application is tested beside the incorrect requirements, identified the flaws, and fixed them on a given period would not help as testing is done on the wrong specification, which does not apply to the client's requirements. The absence of error fallacy means identifying and fixing the bugs would not help if the application is impractical and not able to accomplish the client's requirements and needs.