## **MANUAL TESTING:**

Manual Testing is a type of Software Testing where Testers manually execute test cases without using any automation tools.

Manual Testing is the most primitive of all testing types and helps find bugs in the software system.

Any new application must be manually tested before its testing can be automated. Manual Testing requires more effort, but is necessary to check  automation feasibility.

Manual Testing does not require knowledge of any testing tool.

**AUTOMATION TESTING:**

1. Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.
2. Automation Testing means using an automation tool to execute your test case suite.
3. The automation software can also enter test data into the System Under Test , compare expected and actual results and generate detailed test reports.

**COMPARISON:**

|  |  |
| --- | --- |
| **Manual Testing** | **Automated Testing** |
| Manual testing requires human intervention for test execution. | Automation Testing is use of tools to execute test cases |
| Manual testing will require skilled labour, long time & will imply high costs. | Automation Testing saves time, cost and manpower. Once recorded, it's easier to run an automated test suite |
| Any type of application can be tested manually, certain testing types like ad-hoc and monkey testing are more suited for manual execution. | Automated testing is recommended only for stable systems and is mostly used for Regression Testing |
| Manual testing can be become repetitive and boring. | The boring part of executing same test cases time and again, is handled by automation software in Automation Testing. |

## **What is SDLC?**

SDLC is a process followed for a software project, within a software organization. It consists of a detailed plan describing how to develop, maintain, replace and alter or enhance specific software. The life cycle defines a methodology for improving the quality of software and the overall development process.

The following figure is a graphical representation of the various stages of a typical SDLC.

A typical Software Development Life Cycle consists of the following stages −

### **Stage 1: Planning and Requirement Analysis**

Requirement analysis is the most important and fundamental stage in SDLC. It is performed by the senior members of the team with inputs from the customer, the sales department, market surveys and domain experts in the industry. This information is then used to plan the basic project approach and to conduct product feasibility study in the economical, operational and technical areas.

Planning for the quality assurance requirements and identification of the risks associated with the project is also done in the planning stage. The outcome of the technical feasibility study is to define the various technical approaches that can be followed to implement the project successfully with minimum risks.

### **Stage 2: Defining Requirements**

Once the requirement analysis is done the next step is to clearly define and document the product requirements and get them approved from the customer or the market analysts. This is done through an SRS (Software Requirement Specification) document which consists of all the product requirements to be designed and developed during the project life cycle.

### **Stage 3: Designing the Product Architecture**

SRS is the reference for product architects to come out with the best architecture for the product to be developed. Based on the requirements specified in SRS, usually more than one design approach for the product architecture is proposed and documented in a DDS - Design Document Specification.

This DDS is reviewed by all the important stakeholders and based on various parameters as risk assessment, product robustness, design modularity, budget and time constraints, the best design approach is selected for the product.

A design approach clearly defines all the architectural modules of the product along with its communication and data flow representation with the external and third party modules (if any). The internal design of all the modules of the proposed architecture should be clearly defined with the minutest of the details in DDS.

### **Stage 4: Building or Developing the Product**

In this stage of SDLC the actual development starts and the product is built. The programming code is generated as per DDS during this stage. If the design is performed in a detailed and organized manner, code generation can be accomplished without much hassle.

Developers must follow the coding guidelines defined by their organization and programming tools like compilers, interpreters, debuggers, etc. are used to generate the code. Different high level programming languages such as C, C++, Pascal, Java and PHP are used for coding. The programming language is chosen with respect to the type of software being developed.

### **Stage 5: Testing the Product**

This stage is usually a subset of all the stages as in the modern SDLC models, the testing activities are mostly involved in all the stages of SDLC. However, this stage refers to the testing only stage of the product where product defects are reported, tracked, fixed and retested, until the product reaches the quality standards defined in the SRS.

### **Stage 6: Deployment in the Market and Maintenance**

Once the product is tested and ready to be deployed it is released formally in the appropriate market. Sometimes product deployment happens in stages as per the business strategy of that organization. The product may first be released in a limited segment and tested in the real business environment (UAT-User acceptance testing).

Then based on the feedback, the product may be released as it is or with suggested enhancements in the targeting market segment. After the product is released in the market, its maintenance is done for the existing customer base.

**STLC:**

**Requirement Analysis:**

Requirement Analysis is the very first step in Software Testing Life Cycle (STLC). In this step Quality Assurance (QA) team understands the requirement in terms of what we will testing & figure out the testable requirements. If any conflict, missing or not understood any requirement, then QA team follow up with the various stakeholders like Business Analyst, System Architecture, Client, Technical Manager/Lead etc to better understand the detail knowledge of requirement.

From very first step QA involved in the where STLC which helps to prevent the introducing defects into Software under test. The requirements can be either Functional or Non-Functional like Performance, Security testing.

**Test Planning:**

Test Planning is most important phase of Software testing life cycle where all testing strategy is defined. This phase also called as Test Strategy phase. In this phase typically Test Manager (or Test Lead based on company to company) involved to determine the effort and cost estimates for entire project. This phase will be kicked off once the requirement gathering phase is completed & based on the requirement analysis, start preparing the Test Plan. The Result of Test Planning phase will be Test Plan or Test strategy & Testing Effort estimation documents. Once test planning phase is completed the QA team can start with test cases development activity.

**Test Case Development:**

The test case development activity is started once the test planning activity is finished. This is the phase of STLC where testing team write down the detailed test cases. Along with test cases testing team also prepare the test data if any required for testing. Once the test cases are ready then these test cases are reviewed by peer members or QA lead.

Also the Requirement Traceability Matrix (RTM) is prepared. The Requirement Traceability Matrix is an industry-accepted format for tracking requirements where each test case is mapped with the requirement. Using this RTM we can track backward & forward traceability.

**Test Environment Setup:**

Setting up the test environment is vital part of the STLC. Basically test environment decides on which conditions software is tested. This is independent activity and can be started parallel with Test Case Development. In process of setting up testing environment test team is not involved in it. Based on company to company may be developer or customer creates the testing environment. Mean while testing team should prepare the smoke test cases to check the readiness of the test environment setup.

**Test Execution:**

Once the preparation of Test Case Development and Test Environment setup is completed then test execution phase can be kicked off. In this phase testing team start executing test cases based on prepared test planning & prepared test cases in the prior step.

Once the test case is passed then same can be marked as Passed. If any test case is failed then corresponding defect can be reported to developer team via bug tracking system & bug can be linked for corresponding test case for further analysis. Ideally every failed test case should be associated with at least single bug. Using this linking we can get the failed test case with bug associated with it. Once the bug fixed by development team then same test case can be executed based on your test planning.

If any of the test cases are blocked due to any defect then such test cases can be marked as Blocked, so we can get the report based on how many test cases passed, failed, blocked or not run etc. Once the defects are fixed, same Failed or Blocked test cases can be executed again to retest the functionality.

**Test Cycle Closure:**

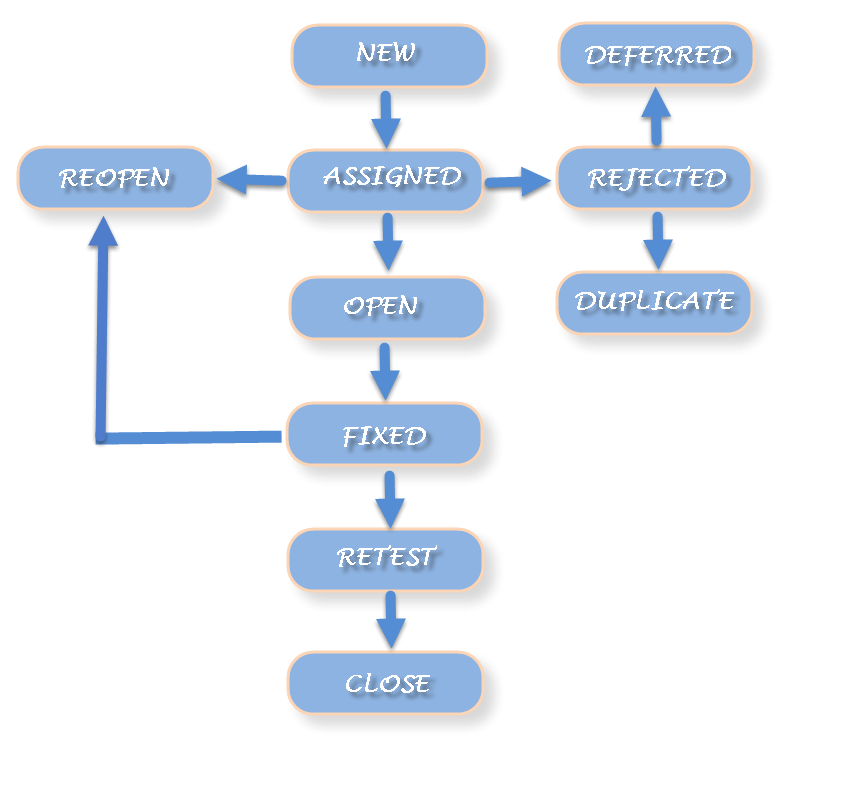
Call out the testing team member meeting & evaluate cycle completion criteria based on Test coverage, Quality, Cost, Time, Critical Business Objectives, and Software. Discuss what all went good, which area needs to be improve & taking the lessons from current STLC as input to upcoming test cycles, which will help to improve bottleneck in the STLC process. Test case & bug report will analyze to find out the defect distribution by type and severity. Once complete the test cycle then test closure report & Test metrics will be prepared. Test result analysis to find out the defect distribution by type and severity.

**COMPARISON OF STLC AND SDLC:**

* STLC is part of SDLC. It can be said that STLC is a subset of the SDLC set.
* STLC is limited to the testing phase where quality of software or product ensures. SDLC has vast and vital role in complete development of a software or product.
* However, STLC is a very important phase of SDLC and the final product or the software cannot be released without passing through the STLC process.
* STLC is also a part of the post-release/ update cycle, the maintenance phase of SDLC where known defects get fixed or a new functionality is added to the software.

**BUG LIFE CYCLE:**

Defect life cycle, also known as Bug Life cycle is the journey of a defect cycle, which a defect goes through during its lifetime. It varies from organization to organization and also from project to project as it is governed by the software testing process and also depends upon the tools used.



### Defect Life Cycle includes following stages:

**New:** When a defect is logged and posted for the first time. Its state is given as new.

Assigned: Once the bug is posted by the tester, the lead of the tester approves the bug and assigns the bug to developer team. There can be two scenario, first that the defect can directly assign to the developer, who owns the functionality of the defect. Second, it can also be assigned to the Dev Lead and once it is approved with the Dev Lead, he or she can further move the defect to the developer.

**Open:** Its state when the developer starts analyzing and working on the defect fix.

**Fixed:** When developer makes necessary code changes and verifies the changes then he/she can make bug status as ‘Fixed’. This is also an indication to the Dev Lead that the defects on Fixed status are the defect which will be available to tester to test in the coming build.

**Retest:** At this stage the tester do the retesting of the changed code which developer has given to him to check whether the defect got fixed or not.

Once the latest build is pushed to the environment, Dev lead move all the Fixed defects to Retest. It is an indication to the testing team that the defects are ready to test.

**Reopened:**  If the bug still exists even after the bug is fixed by the developer, the tester changes the status to “reopened”. The bug goes through the life cycle once again.

**Deferred:** The bug, changed to deferred state means the bug is expected to be fixed in next releases. The reasons for changing the bug to this state have many factors. Some of them are [priority](http://istqbexamcertification.com/what-is-the-difference-between-severity-and-priority/) of the bug may be low, lack of time for the release or the bug may not have major effect on the software.

**Rejected:** If the developer feels that the bug is not genuine, developer rejects the bug. Then the state of the bug is changed to “rejected”.

**Duplicate :** If the bug is repeated twice or the two bugs mention the same concept of the bug, then the recent/latest bug status is changed to “duplicate“.

**Closed:**  Once the bug is fixed, it is tested by the tester. If the tester feels that the bug no longer exists in the software, tester changes the status of the bug to “closed”. This state means that the bug is fixed, tested and approved.

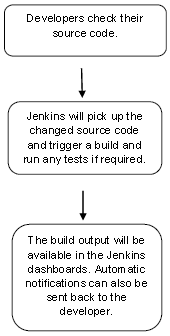
**Not a bug/Enhancement:**The state given as “Not a bug/Enhancement” if there is no change in the functionality of the application. For an example: If customer asks for some change in the look and field of the application like change of color of some text then it is not a bug but just some change in the looks of the  application.

**GIT:**

**Git** is a version control system for tracking changes in computer files and coordinating work on those files among multiple people. It is primarily used for source code management in software development, but it can be used to keep track of changes in any set of files.

**JENKINS:**

Jenkins is a software that allows continuous integration. Jenkins will be installed on a server where the central build will take place. The following flowchart demonstrates a very simple workflow of how Jenkins works.



**CONTINUOUS INTEGRATION:**

**C**ontinuous integration (CI) is a software engineering practice in which isolated changes are immediately tested and reported on when they are added to a larger code base. The goal of CI is to provide rapid feedback so that if a defect is introduced into the code base, it can be identified and corrected as soon as possible. Continuous integration software tools can be used to automate the testing and build a document trail.

**TESTING TYPES:**

**UNIT TEST:**

Unit testing is a software testing method by which individual units of source code, such as functions, methods, and class are tested to determine whether they are fit for use. Intuitively, one can view a unit as the smallest testable part of an application. Unit tests are short code fragments created by programmers during the development process.

**SANITY TEST:**

Sanity testing, a software testing technique performed by the test team for some basic tests. The aim of basic test is to be conducted whenever a new build is received for testing.Sanity test is usually unscripted, helps to identify the dependent missing functionalities. It is used to determine if the section of the application is still working after a minor change.Sanity testing can be narrow and deep. Sanity test is a narrow regression test that focuses on one or a few areas of functionality.

## **Functional Testing:**

Functional Testing is a testing technique that is used to test the features/functionality of the system or Software, should cover all the scenarios including failure paths and boundary cases.

**Integration Testing:**

**integration testing** (sometimes called **integration and testing**, abbreviated **I&T**) is the phase in [software testing](https://en.wikipedia.org/wiki/Software_testing" \o "Software testing) in which individual software modules are combined and tested as a group. It occurs after [unit testing](https://en.wikipedia.org/wiki/Unit_testing" \o "Unit testing) and before [validation testing](https://en.wikipedia.org/wiki/Software_verification_and_validation" \o "Software verification and validation).

## **Regression Testing:**

Regression testing a black box testing technique that consists of re-executing those tests that are impacted by the code changes. These tests should be executed as often as possible throughout the software development life cycle.

## **Performance Testing:**

Performance testing, a non-functional testing technique performed to determine the system parameters in terms of responsiveness and stability under various workload. Performance testing measures the quality attributes of the system, such as scalability, reliability and resource usage.

**Load testing:** It is the simplest form of testing conducted to understand the behaviour of the system under a specific load. Load testing will result in measuring important business critical transactions and load on the database, application server, etc., are also monitored.

**PRODUCT LINE:**

A product line is a group of related products under a single brand sold by the same company. Companies sell multiple product lines under their various brands. Companies often expand their offerings by adding to existing product lines, because consumers are more likely to purchase products from [brands](https://www.investopedia.com/terms/b/brand.asp) with which they are already familiar.