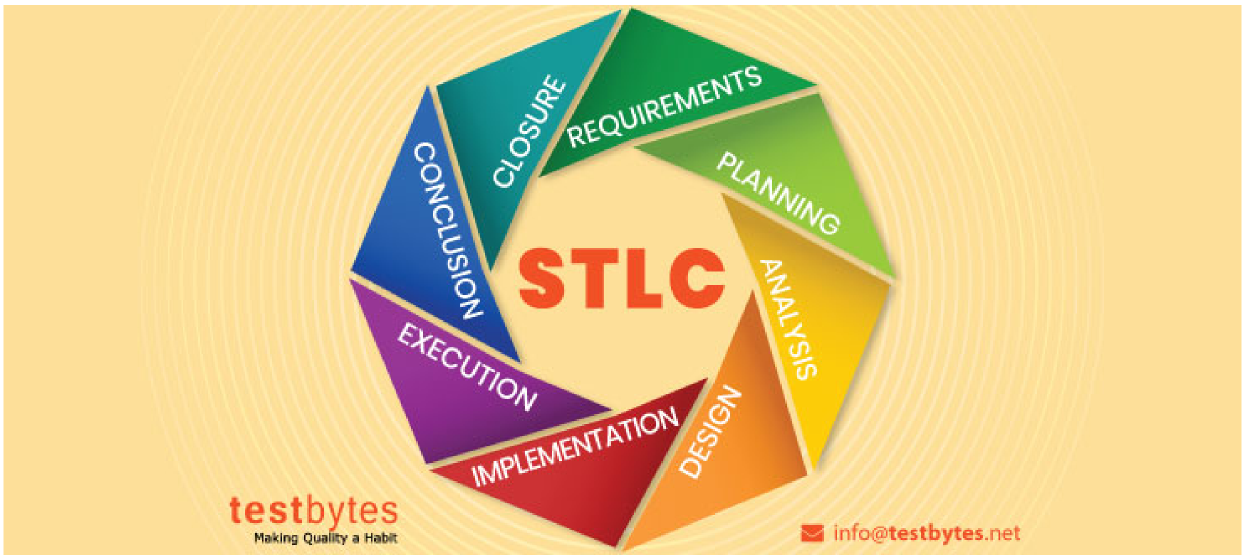
Software Testing Life Cycle (STLC)

t **testbytes.net**/blog/software-testing-life-cycle
July 31, 2019
Process

Wednesday July 31, 2019



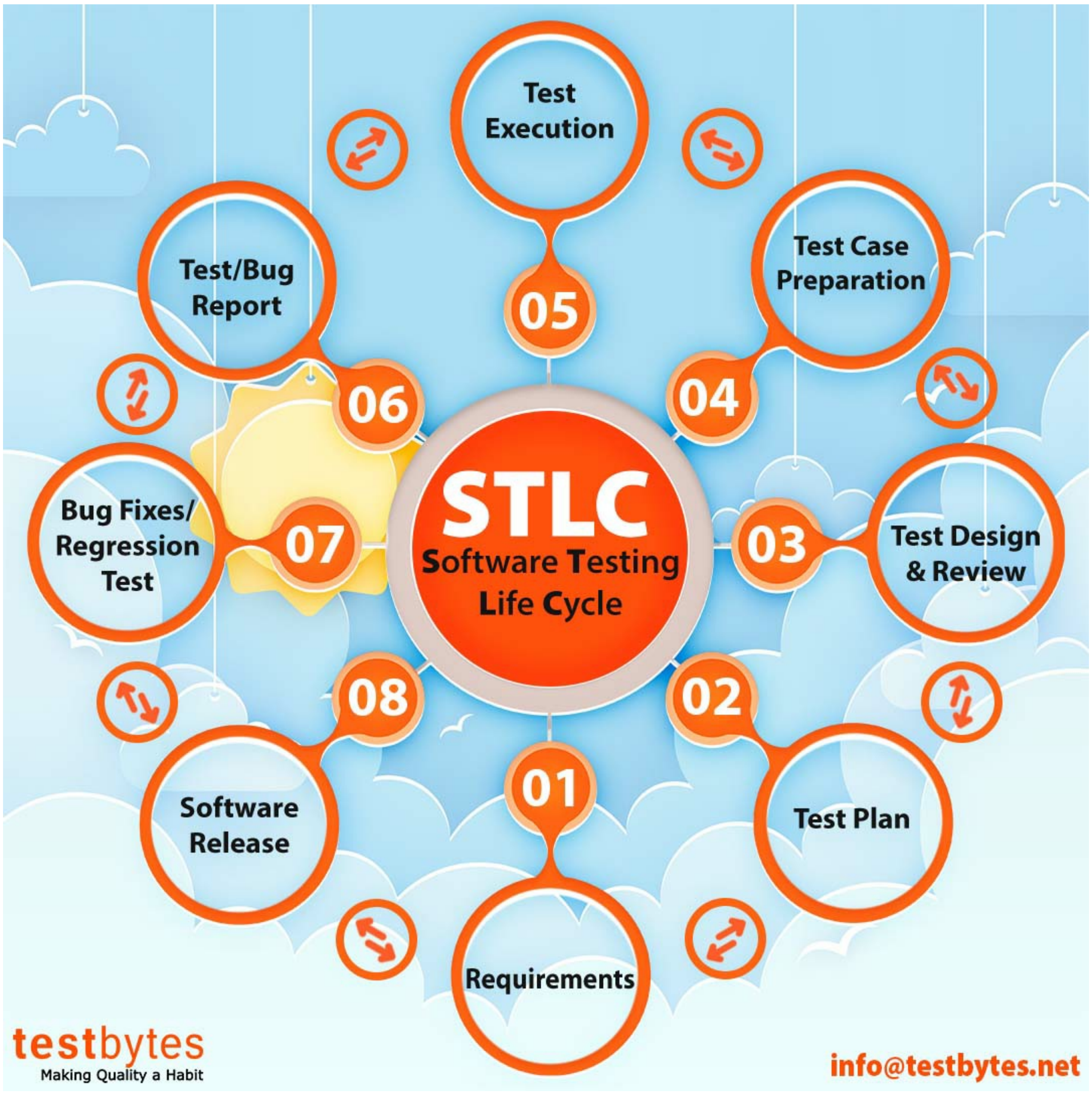
Software testing is a very important part of software development. owing to the huge demand and necessity of testing nowadays, The software testing process is now getting more intricate, and competent. Because of that, Software testing not only limited to finding bugs but plays an important role in delivering a quality product to the end-users and on time. This has resulted in making the whole testing process streamlined and organized. And that led to the development of software testing life cycle or what is commonly known to us as STLC.

What is STLC?

Software testing life cycle is a sequential flow of various testing processes that are to be executed to ensure maximum results from software testing. STLC ensures that all the testing processes are carried out in an orderly and planned manner.

We have understood that **STLC** is a process, now look at the different steps involved in this process:

1. Requirement Gathering
2. Test planning
3. Test case Development
4. Set up the test environment (Test, Int, Prod)
5. Test case Execution
6. Test cycle Closure/Maintenance



1. Requirement Gathering

Requirement Gathering is the very first Step in **STLC** process. The document containing these requirements is called the Specification Document. This is a very important part of the software because all the development would be revolving around this document.

It is always advised to get a sign off from the client for the requirements document, to reiterate the correctness of the specification that the client is looking for. It is with these requirements that testers would be able to come up with the test cases.

Requirements will be either in the form of a document or User Stories. In Agile a requirement is called User story. Each user story can be further broken down into several tasks and assigned to different members of the team.

The requirements are received from client/stakeholders who are in need of the product/service. A BA (business analyst) is normally the person who prepares this document after discussions with the client. It is advisable to have the client, the BA, product owner or manager along with team leads (including QA lead) and architects should be attending the requirement gathering meeting.

Entry Criteria

- Availability of requirement Document, acceptance criteria and application architectural document.

Activities

- Recognizing the tests to be executed.
- Identify test priorities and focuses
- Create Requirement [Traceability Matrix] (RTM).
- Recognize details about the test environment
- Carry out Automation feasibility for the testing.

Deliverables

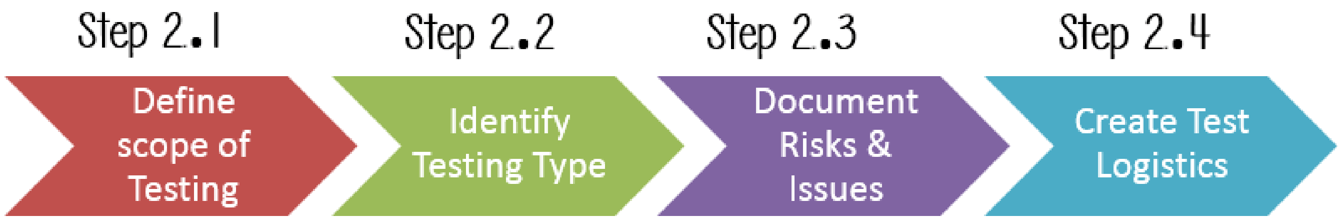
- RTM
- Automation feasibility report.

Exit Criteria

- RTM
- Automation Feasibility report

2. Test Planning

Once the requirement analysis is over next is the planning phase. Team lead/Test Lead or sometimes Manager will prepare the Test Plan document. This plan will be kept in share drives or test management tools like Rally, ALM, etc. so that they are accessible to everyone.



The test plan document is as a single point of reference when it comes to all testing activities. It also works like the document that is shared between the Dev team, Business Analysts, Project Managers, and the other teams for concurrence.

Owing to this transparency of the QA team's work to the external teams will increase to a great extent. The main input needed for the test plan is the signed of requirement document.

Test plan Document may have the following Sections:

1. Entry Criteria

It specifies the conditions which must be satisfied for the testing process to begin. The most common entry for testing is a stable build which has passed the smoke test within the minimum number of failures or the data setup is ready or the hardware is ready etc.

2. Scope

Scope of the Testing will be listed in this section.

The scope of the testing refers to the list of requirements from the requirement document that are feasible for testing in that particular release or iteration.

There will also be a traceability matrix created which will map the requirements to feasibility, test cases, test case runs, test case status and defects. This matrix will clearly indicate the status of the testing and quality of the product.

3. Out of Scope

Out of Scope feature will be listed in this section.

Any features or requirements which are not covered as part of the current release will be marked as out of scope. There can be several reasons for marking an item as out of scope. Data setup not available, dependency on other modules, application not ready, not feasible for testing are some of these reasons.

4. Assumptions

Assumptions are listed in this section.

As the name suggests this section would contain the list of things we think would be ready when we are starting the testing. Some of the items covered in this section are application is up and running, resources are available, hardware setup is up and running. Each project can have a different set of assumptions.

5. Schedules

Here we can write about how many test cycles are needed. Start date and end date of each cycle, the sequence of testing, etc. This section will contain an estimate of the number of test cases available and how long it will take to script and test each of these.

Then based on the number of resources available a timeline is reached for test case creation, data preparation (if applicable), and test case execution and reporting.

6. Roles and Responsibilities

Team members and their role and responsibilities are listed in this section. Since this test plan document is shared with the development and other teams, it helps them to know whom to contact in case needed.

7. Defect management

This section talks about the process followed for reporting and closing defects. Most companies use test management or defect tracking tools like ALM, Bugzilla, and Rally, etc. This section also discusses to whom the defects will be assigned and the steps to closure.

8. Deliverables

This section contains the list of documents the testing team promises to deliver after the testing process is completed. Test results summary, defect list, and testing metrics are some the deliverables that can be mentioned here.

9. Hardware and Environments

This section mentions the hardware requirements and setup that is needed for testing and the environment in which the testing will be done in each phase. In most cases, the testing is done in QA or Stage environment with a dummy copy of production data.

10. Risks

This is a very important section which lists out the risks and contingencies along with the mitigation plan. There can be many types of risks like hardware failures, application downtime, data unavailable, etc. These basically refer to issues that can impact your testing activities and schedule. For all risks mentioned, it is advisable to a mitigation plan. This is like a plan B when plan A fails.

11. Exit Criteria

Here we need to mention when the testing will be stopped. There are 2 situations when testing is stopped. One, when the testing is completed.

In this, the exit criteria would be testing completed for all test cases and x% of the defects closed. (X is the % of defects closed and would be decided by the management). Second, when testing needs to be stopped due to various reasons. Build instability, too many Sev 1 and Sev 2 defects or reduced timelines can be some of these reasons.

3. Test Case development

What is test case?

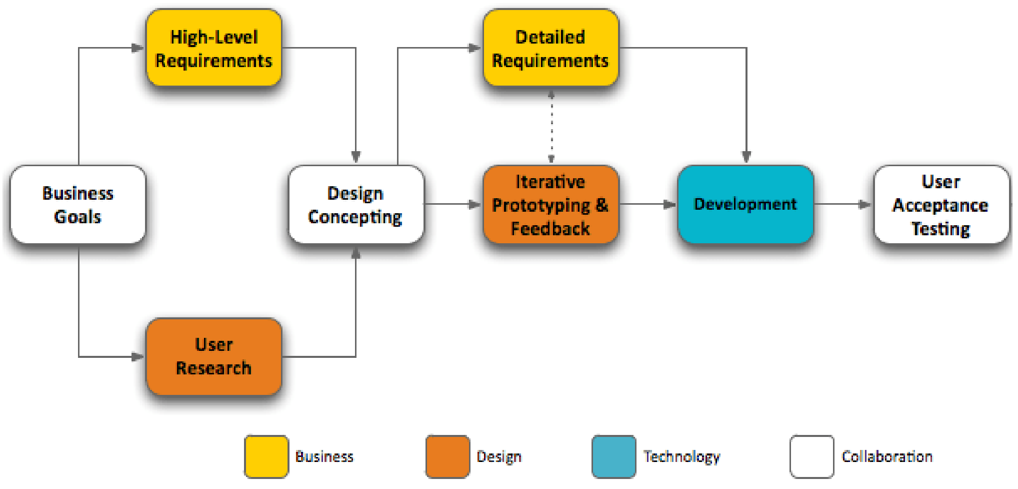
A TEST CASE can be defined as a set of variables with which is used by a tester to determine whether the system under test satisfies requirements and works correctly. The process of developing test cases can also help find problems in the requirements or design of an application.

Who will write the test case?

Usually, Test case document will be developed by the testers. The test cases are written in such a way that it validates each requirement.

Test case is based on what document?

Test cases will be based on the requirement document (SRS- Software Requirement Specification) or in agile where we do not have proper requirement document the test case will be based on the User stories



How do you identify Test cases?

Ideally, for each requirement, there should be at least one test case. In Agile User Stories will be broken into tasks. Usually, there will be multiple test cases for each requirement/task. The mapping of the Test case to Requirement is called Traceability Matrix.

While writing the test cases one should be careful to make sure that a maximum number of options, navigations, and validations are covered. Ideally, the test cases should be written such that every branch of the code is exercised.

Different Types of Test cases

Regression Test cases:

Regression test cases are those which cover stable and previously rolled out functionalities. Regression test cases are usually automated and are called Regression Suite. Also known as Sanity check.

Functional Test cases:

Functional test cases are test cases which validate the functional aspect of the AUT. Most of the UI and interface-related test cases would be functional. Login, logout, navigational flow and error scenarios are examples of functional test cases.

Non Functional Test Cases:

The test cases which are not related to the functionality of the application are known as non-functional test cases. For example the layout of the page, font, color, text size, image size, etc.

Integration Test cases:

When data flows from one system to another, then the test cases need to be elaborately written to cover this. These are called integration tests as they validate the integration between one or more systems.

Performance Test cases:

Test cases related to Performance of the site or application like the loading time, load it can take, etc.

White Box Test cases:

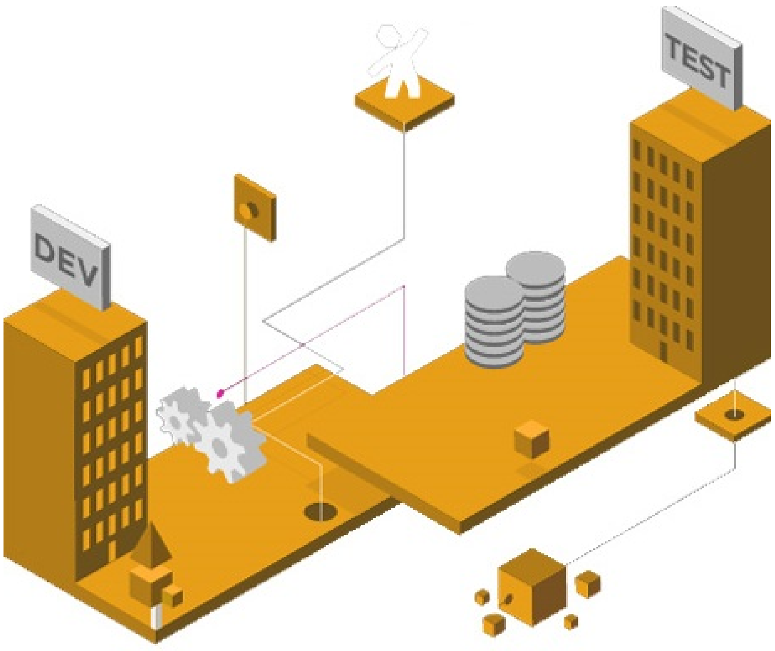
Some amount of technical knowledge is needed to write white box test cases. These test cases validate the developer code and the functional modules developed.

Let us now try to write a typical test case. It should have the following fields

- Test case number
- Test case Name
- Input steps
- Test case Type(functional, performance, Regression)-optional
- Expected Result
- Actual Result
- Status
- Requirement No
- Bugs associated (Bug ids)

If Expected Result and Actual result matches Status will be pass else fail.

4. Setting up Test Environment



Most of the companies will have a different environment to test against examples of environment would be TEST, INT, and PROD (testing, integration, and production).

There could be multiple test environments. The setup would also include any data and device setup that is needed to kick start the execution process.

Once the test cases are ready next test environment needs to be set. For example, if we are going to execute the test cases against "Test Environment "the build which developer sends for testing need to be deployed in Test Environment.

Sanity check or Regression needs to be run and there should be no new failure which means the new feature is not affecting the existing application. Only then QA can execute test cases against it.

5. Test case Execution

Here is where the actual execution takes place. Each tester is assigned a separate set of test cases. Based on the step to reproduce as mentioned in the test case document, the tester would validate whether the mention functionality is pass or fail. In case it is a fail, he needs to open a defect for the defect. He may also be required to attend triage meetings to help the developers to better understand the problems and hence fix it too.

Read also : Test Management Process: A Complete Guide

While the testing is in progress the team would be having a daily status reporting to the test lead or manager to track the progress. The team may also need to be constantly in touch with the developers to help them understand the issues and defects.

In case any bug fixes are received, the team needs to make sure the issues are actually fixed and also validate that the fix has not adversely affected any other functionality

6. Testing closure/Maintenance

Once all the test cases are completed and most of the defects are tracked to closure, the test process is also closed. All the deliverable’s as mentioned in the test plan is shared with the development team along with the certification of whether or not the build is a go or no-go to the next stage.

Hope this section was useful to you in understanding the STLC process. Though we have talked about the 6 steps there are always customizations that we need to do to make things smoother for the project.

With Scrum and agile being the buzz words, the STLC has been reduced to a minimum with no proper test plan and in most cases no detailed test cases as well. But always remember the process is important and the more you do away with the greater is the risk.

Why do we need STLC?

Software testing ensures the high quality of the software product and is essentially a very important part of software development. The software testing life cycle ensures that the testing process is carried out in a well-planned and sequential order thus promising better testing results.

The Software Testing Life Cycle can embrace 4 different testing models namely the agile model, the waterfall model, the V model, and the spiral model.

- **Agile model:** Agile Testing model is being largely adopted by testing teams. It is based on continuous integration between testers and developers. Agile Testing does not follow a sequential flow of testing processes.
- **Waterfall testing model:** Waterfall Model or Linear Sequential Life Cycle follows a sequential order of testing. We only move to the next phase of testing when the previous phase is completed. It is good for small projects where the requirements are very clear.
- **V model:** V Model is an advanced version of the classic waterfall model. In the V model, each phase of the software development life-cycle is validated before going to the next level. In the V model, software testing starts as soon as the requirements are written. V model helps in detecting defects in the early stages of life-cycle and it also minimizes probable future defects.
- **Spiral Model:** Spiral Model is based on incremental and prototype technique. It is usually applied in large and complicated projects with high-risk probability.

Conclusion

A systematic and sequential approach is always beneficial for any process. Software testing life cycle is such a systematic and sequential way of conducting software testing that promises more effective and efficient testing results.