STLC

(Software Testing Life Cycle)

tutorialspoint

SIMPLY EASY LEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia https://twitter.com/tutorialspoint

About the Tutorial

Software Testing Lifecycle is a standard procedure divided into different phases, followed by the QA Team to complete all testing activities. This is a brief tutorial that introduces the readers to the various phases of Software Testing Life Cycle.

Audience

This tutorial has been prepared for beginners to help them understand the software testing lifecycle. It would help all those professionals who would like to understand the Testing Framework in detail along with its types, methods, and levels.

Prerequisites

There are no specific prerequisites for this tutorial. Any software professional can go through this tutorial to get a bigger picture of how the high-quality software applications and products are designed. However, it would certainly be an advantage if the readers have a basic of understanding of some fundamental testing concepts.

Copyright & Disclaimer

Table of Contents

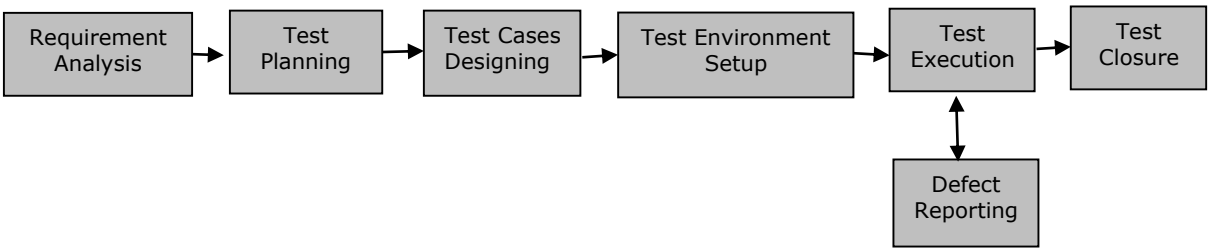
1. STLC – Overview

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.
- STLC provides a step-by-step process to ensure quality software.
- In the early stage of STLC, while the software or the product is developing, the tester can analyze and define the scope of testing, entry and exit criteria and also the Test Cases. It helps to reduce the test cycle time along with better quality.
- As soon as the development phase is over, the testers are ready with test cases and start with execution. This helps to find bugs in the initial phase.

STLC Phases

STLC has the following different phases but it is not mandatory to follow all phases. Phases are dependent on the nature of the software or the product, time and resources allocated for the testing and the model of SDLC that is to be followed.



There are 6 major phases of STLC:

- **Requirement Analysis** – When the SRD is ready and shared with the stakeholders, the testing team starts high level analysis concerning the AUT (Application under Test).
- **Test Planning** – Test Team plans the strategy and approach.
- **Test Case Designing** – Develop the test cases based on scope and criteria's.
- **Test Environment Setup**–When integrated environment is ready to validate the product.
- **Test Execution** – Real-time validation of product and finding bugs.
- **Test Closure** – Once testing is completed, matrix, reports, results are documented.

2. Comparison – STLC and SDLC

In this chapter, we will understand the factors of comparison between STLC and SDLC. Let us consider the following points and thereby, compare 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.

The following table lists down the factors of comparison between SDLC and STLC based on their phases:

Phase	SDLC	STLC
Requirement Gathering	• Business Analyst gathers requirements. • Development team analyzes the requirements. • After high level, the development team starts analyzing from the architecture and the design perspective.	• Testing team reviews and analyzes the SRD document. • Identifies the testing requirements - Scope, Verification and Validation key points. • Reviews the requirements for logical and functional relationship among various modules. This helps in the identification of gaps at an early stage.
Design	• The architecture of SDLC helps you develop a high-level and low-level design of the software based on the requirements. • Business Analyst works on the mocker of UI design. • Once the design is completed, it is signed off by the stakeholders.	• In STLC, either the Test Architect or a Test Lead usually plan the test strategy. • Identifies the testing points. • Resource allocation and timelines are finalized here.

Development	• Development team starts developing the software. • Integrate with different systems. • Once all integration is done, a ready to test software or product is provided.	• • Testing team writes the test scenarios to validate the quality of the product. • • Detailed test cases are written for all modules along with expected behaviour. • The prerequisites and the entry and exit criteria of a test module are identified here.
Environment Set up	• Development team sets up a test environment with developed product to validate.	• • The Test team confirms the environment set up based on the prerequisites. • Performs smoke testing to make sure the environment is stable for the product to be tested.
Testing	• The actual testing is carried out in this phase. It includes unit testing, integration testing, system testing, defect retesting, regression testing, etc. • The Development team fixes the bug reported, if any and sends it back to the tester for retesting. • UAT testing performs here after getting sign off from SIT testing.	• System Integration testing starts based on the test cases. • Defects reported, if any, get retested and fixed. • Regression testing is performed here and the product is signed off once it meets the exit criteria.
Deployment/ Product Release	• Once sign-off is received from various testing team, application is deployed in prod environment for real end users	• Smoke and sanity testing in production environment is completed here as soon as product is deployed. • Test reports and matrix preparation are done by testing team to analyze the product.
Maintenance	• It covers the post deployment supports, enhancement and updates, if any.	• In this phase, the maintaining of test cases, regression suits and automation scripts take place based on the enhancement and updates.

3. Testing – Fundamental Principles

The common objective of testing is finding bugs as early as possible. And, once the bugs are fixed, make sure it is working as expected and not breaking any other functionality.

To achieve these goals, seven basic principles are given for software testing:

What Testing shows?

Testing can show that defects are present but there is no way to prove that there is no defect in the product. Testing phases make sure that the application under test is working based on the given requirement and it helps to reduce the probability of undiscovered defects in the application. But, even if no defects are found, it does not mean that it is absolutely correct. We can assume that AUT is matching with our exit criteria and maintaining the requirements according to SRD.

Is Exhaustive Testing possible?

100% coverage or testing of all combinations of inputs and possible combinations are not possible except of trivial cases. Instead of exhaustive testing, risk analysis and priorities are used to define the scope of testing. Here, most of the real time scenarios can consider including most probable negative scenario as well. This will help us track the failure.

Early Testing

Testing activities should start as soon as possible and be focused on defined objectives in Test Strategy and expected results. Early stage of testing helps to identify Requirement Defect or design level discrepancies. If these types of bugs are captured in initial stage, it helps us save time and is cost-effective too. The answer to why testing should start at an early stage is very simple – as soon as the SRD is received, the testers can analyze the requirement from the testing perspective and can notice a requirement discrepancy.

Defect Clustering

Based on previous product defect analysis, it can be said that most of the defects are identified from small set of modules which are critical for application. These modules can be identified based on complexity, different system interaction or dependency on different other modules. If testers can identify these crucial modules, they can focus more on these modules to identify all possible bugs. In a study, it is found that 8 out of 10 defects are identified from 20% functionality of AUT.

Pesticide Paradox

What is pesticide paradox – if pesticides are frequently used on crops, there comes when the insects develop a certain kind of resistance and gradually the pesticides thus sprayed seem to be ineffective on the insects.

The same concept is applicable on testing also. Here, insects are bugs while pesticides are test cases that are used to run again and again. If the same sets of test cases are executed again and again, these test cases become ineffective after certain timeframe and the testers will not be able to identify any new defect.

To overcome these conditions, test cases should be revised and reviewed time to time and new and different test cases can be added. This will help in identifying new defects.

Testing is Context Dependent

This principle states that two different type of application can't be tested using same approach until both applications are of same nature. For example, if a tester uses the same approach for Web Based Application and Mobile Application, that is completely wrong and there is high risk of poor quality of product release. Testers should use different approaches, methodologies, techniques and coverage for different types and nature of applications.

Absence of Error – Fallacy

This principle states finding defects and fixing them until the application or system is stable, is time consuming and also eats up on the resources. Even after fixing 99% of the defects, there is a high risk of unstable application. The first essential thing is to verify the stability of the application and the prerequisites of the environment. If these two conditions fulfill, only then we can start with the detailed testing.

End of ebook preview

If you liked what you saw…

Buy it from our store @ https://store.tutorialspoint.com