**Application Building**
**Execute and Test your model**

| | |
|---|---|
| Date | 03 October 2022 |
| Team ID | PNT2022TMID433992 |
| Project Name | Project - Web Phishing Detection |

# What is the software testing life cycle

The software testing life cycle is a sequence of tasks designed to understand the state of a system and make recommendations for improvement. The STLC involves strategizing, planning, executing and completing test cycles.

Traditionally, QA testing occurred shortly before product release as a way to ensure digital products don't contain defects that negatively affect core functionality. However, as digital systems became more complex and businesses released batches of software and apps more often, the STLC evolved. In many organizations, testing no longer waits until a product is fully developed. Over the last couple of decades, some organizations have included STLC phases before and during development to maximize resources, employing some of the following tactics:

- Test automation
- Test-driven development
- Crowd testing
- Shift-left testing
- Shift-right testing

An effective STLC produces more comprehensive and valid results than a traditional post-development testing stage, helping organizations make changes that ultimately drive customer satisfaction and, thus, more revenue. The STLC process should be less a pre-release obligation than an effort to discover key insights that will benefit the business over the short- and long-term.

## 6 key STLC phases

The software testing life cycle provides confidence in a software release. The STLC delivers that confidence through a series of tasks that take validation through ideation to design and execution. Each STLC phase is useful in its own way to achieve high-quality software releases. Likewise, each part of the STLC process comes with its own goals and deliverables, all intended to catch defects and optimize test coverage.

Let's dig into these sequential phases of the software testing life cycle:

1. Requirement analysis
2. Test planning
3. Test case design and development
4. Test environment setup
5. Test execution
6. Test cycle closure

## 1. Requirement analysis

Most development initiatives begin with software requirements that specify what the business expects from the project. Software requirements often include high-level business needs, architectural requirements that detail how the feature will be designed and supported, and detailed system requirements from which developers build the product. System requirements include functional and non-functional specifications, both of which present opportunities to test and validate.

In this STLC phase, testers work both within their own teams and cross-functionally to contextualize how they will test the software. Requirement analysis often includes brainstorming sessions, identifying blind spots or unclear areas in the requirements, and prioritizing certain assessments.

When in doubt or lacking requirements documentation, the QA team will question the engineering or business side to clarify and calcify a testing strategy.

## 2. Test planning

The second STLC phase is important, as it guides much of the work to follow. Test planning takes the insights found during requirements or product analysis and turns them into a documented QA strategy.

The test team leadership determines what resources and efforts will evaluate the release. The resulting test plan documentation both informs testers and other departments how the testing work will commence, keeping everyone on the same page. This plan is especially helpful if other members of the organization will take part in testing and bug remediation, such as developers executing unit tests and writing hotfixes.

The test plan spells out several details of the QA work to be done, including the scope, objectives, types of functional and non-functional tests (both automated and manual), and details for the test environments. Once these details are determined, **test management** sets roles and timelines for the work. Finally, the

testing team can determine what deliverables it will provide upon completion of the STLC phases.

## 3. Test case design and development

With the test plan in place, testers can begin to write and create detailed test cases. In this STLC phase, the QA team fleshes out the details of the structured tests they will run, including any test data they will need to facilitate those tests. While tests must ultimately validate the areas defined by requirements, testers can exert their skills and creativity in *how* they achieve this task.

When conceptualizing test cases, the tester's goal should be to validate functionality within the allotted time and scope, especially core functionality. Test cases should be simple and well understood for any member of the team, but also unique from other test cases. Test cases should aim to achieve full coverage of the requirements in the specifications document — a traceability matrix can help track coverage. It's important that test cases be identifiable and repeatable, as developers will add new functionality to the product over time, requiring tests to run again. They must also not alter the test environment for future tests, especially when validating configurations.

**Test cases might also require maintenance** or updates over time to validate both new and existing functionality. This work also occurs at this STLC stage.

Once test cases are ready, a test team lead or peer can review them. They might also review and update automated test scripts at this STLC stage. Ultimately, the team prioritizes and organizes these test cases into test suites that run later.

## 4. Test environment setup

The test environment provides the setting where the actual testing occurs. This is a crucial software testing life cycle phase, and it requires help from other members of the organization. Testers must have access to bug reporting capabilities, as well as the application architecture to support the product. Without these elements, testers might not be able to do their jobs.

Once ready, testers establish the **parameters for the test environment**, which include the hardware, software, test data, frameworks, configurations and network. In this STLC phase, testers adjust these environment parameters depending on what the test case requires. For example, the majority of a product's users might be on an Android device, use a certain version of a Chrome browser and have a certain amount of processing power on those devices — these are parameters the test environment would include.

Smoke tests within these test environments provide a very early and rudimentary check that the software is ready for more comprehensive testing. These smoke tests against the builds are part of the deliverable in this STLC phase.

## 5. Test execution

Next in the software testing life cycle, it's time to fully test the product. At this STLC stage, testers execute all of the test cases, or as many as is possible within the allotted time. QA professionals and automated scripts execute a number of functional and non-functional tests.

Here in the STLC, testers will identify and **report detailed bugs** that arise from test case execution and log the system's performance compared to its requirements. As developers make fixes, testers often retest the product to make sure new defects don't materialize. With all of these tests piling up in the test execution STLC phase, it's important to make use of test automation where possible to achieve the test coverage and velocity you need.

## 6. Test cycle closure

The final STLC phase is test cycle closure. In this stage, the testing team provides a test closure report, which summarizes and communicates its findings with the rest of the team. This report typically includes summaries of the testing work and results, an assessment of the testing and the manager's approval.

During the test cycle closure, the testing team checks its deliverables, which include details relevant to the testing work, such as the test strategy, test case documents, automated test scripts and test results. The team will then complete and close incident reports, which detail unusual or unexpected behaviour that the test team observes during testing. The team must also archive the resources it used during testing, such as scripts, tools and environments, for later use.

From there, the organization plans the product **for support and release**, which often includes acceptance and feedback from customer representatives.

Communication is key in this STLC phase, as additional perspectives might uncover a quality, cost or coverage issue that the rest of the group missed. These discussions can yield additional analysis or inform how to improve QA work in the future.

### Agile affects the software testing life cycle

The common software testing life cycle phases above follow a sequential approach similar to Waterfall application development. However, as many businesses rethink

how they develop products, testing must also adapt to align with iterative organizational practices and pace of releases.

The QA team might follow an **Agile testing method** instead, which impacts the STLC phases above in various ways. Most notably, an Agile testing team might not file a report or an assessment of the testing work — the release would simply be planned for delivery.

Additionally, Agile testing places an emphasis on shift-left and shift-right testing to alleviate QA bottlenecks. While test automation is included in the STLC phases above, agile testing might place a higher priority on methods like in-sprint testing and test-driven development, both of which generally result in cleaner, simpler bits of code. Production testing, or shift-right testing, helps the QA team identify defects after the test cycle closure. While these defects are often more costly to correct, it's better late than never when it comes to fixing a bug. Shift-right often involves exploratory testing and user testing to find defects that test cases failed to uncover.

In short, the STLC phases above might change slightly depending on the organization's development and testing philosophies.