WATER FALL

The Waterfall model is the earliest SDLC approach that was used for software development. The waterfall Model illustrates the software development process in a linear sequential flow. This means that any phase in the development process begins only if the previous phase is complete. In this waterfall model, the phases do not overlap.

V-MODEL

The V-model is an SDLC model where execution of processes happens in a sequential manner in a V-shape. It is also known as **Verification and Validation model**. The V-Model is an extension of the waterfall model and is based on the association of a testing phase for each corresponding development stage. This means that for every single phase in the development cycle, there is a directly associated testing phase. This is a highly-disciplined model and the next phase starts only after completion of the previous phase.

SPIRAL MODEL

The spiral model combines the idea of iterative development with the systematic, controlled aspects of the waterfall model. This Spiral model is a combination of iterative development process model and sequential linear development model i.e. the waterfall model with a very high emphasis on risk analysis. It allows incremental releases of the product or incremental refinement through each iteration around the spiral.

BIGBANG

The Big Bang model is an SDLC model where we do not follow any specific process. The development just starts with the required money and efforts as the input, and the output is the software developed which may or may not be as per customer requirement. This Big Bang Model does not follow a process/procedure and there is a very little planning required. Even the customer is not sure about what exactly he wants and the requirements are implemented on the fly without much analysis.

AGILE

Agile SDLC model is a combination of iterative and incremental process models with focus on process adaptability and customer satisfaction by rapid delivery of working software product. Agile Methods break the product into small incremental builds. These builds are provided in iterations. Each iteration typically lasts from about one to three weeks.

**What is software testing?**

* Testing is the process of evaluating a system or its component(s) with the intent to find whether it satisfies the specified requirements or not.

. In simple words, testing is executing a system in order to identify any gaps, errors, or missing requirements in contrary to the actual requirements.

* According ANSI/IEEE 1059 standard, Testing can be defined as - A process of analyzing a software item to detect the differences between existing and required conditions (that is defects/errors/bugs) and to evaluate the features of the software item.

**Why testing is important?**

[Software Testing](http://istqbexamcertification.com/what-is-a-software-testing/) is necessary because we all make mistakes. Some of those mistakes are unimportant, but some of them are expensive or dangerous. We need to check  everything and anything we produce because things can always go wrong – [humans make mistakes all the time](http://istqbexamcertification.com/when-do-defects-in-software-testing-arise/).

There are several reasons which clearly tells us as why [Software  Testing[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://istqbexamcertification.com/why-is-testing-necessary/#87255437)](http://istqbexamcertification.com/why-is-testing-necessary/#87255437) is [important[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://s.igmhb.com/click?v=VVM6MTI4MjIzOjIxOTIyOmltcG9ydDoyZmU0ZDA1NDU5NzU2YjhlYzhkMWUwZTgzMDYwZjJlYjp6LTIyMTctODgyOTY3ODY6aXN0cWJleGFtY2VydGlmaWNhdGlvbi5jb206MzczOTcxOjA6ZDUxNDMwM2FjMmRkNDQ3NTk3NDdiMjM4YjU0ZTdmNWE6MTpkYXRhX3NzLDcyOHgxMzY2O2RhdGFfcmMsMTtkYXRhX2ZiLG5vOzo0OTQ4MTYwOjo6MC4wNQ&subid=g-88296786-7850546d59644870bdeb01a5ab2c7fae-&data_ss=728x1366&data_rc=1&data_fb=no&data_tagname=A&data_ct=link_only&data_clickel=link&data_sid=39baad8c8846551c60aaba081f3bdaa9)](http://s.igmhb.com/click?v=VVM6MTI4MjIzOjIxOTIyOmltcG9ydDoyZmU0ZDA1NDU5NzU2YjhlYzhkMWUwZTgzMDYwZjJlYjp6LTIyMTctODgyOTY3ODY6aXN0cWJleGFtY2VydGlmaWNhdGlvbi5jb206MzczOTcxOjA6ZDUxNDMwM2FjMmRkNDQ3NTk3NDdiMjM4YjU0ZTdmNWE6MTpkYXRhX3NzLDcyOHgxMzY2O2RhdGFfcmMsMTtkYXRhX2ZiLG5vOzo0OTQ4MTYwOjo6MC4wNQ&subid=g-88296786-7850546d59644870bdeb01a5ab2c7fae-&data_ss=728x1366&data_rc=1&data_fb=no&data_tagname=A&data_ct=link_only&data_clickel=link&data_sid=39baad8c8846551c60aaba081f3bdaa9) and what are the major things that we should consider while testing of any product or application.

Software testing is very important because of the following reasons:

1. Software testing is really required to point out the [defects](http://istqbexamcertification.com/what-is-defect-or-bugs-or-faults-in-software-testing/) and errors that were made during the [development phases](http://istqbexamcertification.com/what-are-the-software-development-life-cycle-sdlc-phases/).
2. It’s essential since it makes sure of the Customer’s reliability and their satisfaction in the application.
3. It is very [important[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://s.igmhb.com/click?v=VVM6MTI4MjIzOjIxOTIyOmltcG9ydGluZzo5ZDZjYWQ0ZTllYWZiZmZjZmJmYjVkNTMxOWZhNDQzMTp6LTIyMTctODgyOTY3ODY6aXN0cWJleGFtY2VydGlmaWNhdGlvbi5jb206MzczOTcxOjA6MjE0MTA1N2YxY2JkNDFiM2I0NWJmMTdlY2U4OWRjZjI6MTpkYXRhX3NzLDcyOHgxMzY2O2RhdGFfcmMsMTtkYXRhX2ZiLG5vOzo1ODExNDEyOjo6MC4wNQ&subid=g-88296786-7850546d59644870bdeb01a5ab2c7fae-&data_ss=728x1366&data_rc=1&data_fb=no&data_tagname=A&data_ct=link_only&data_clickel=link&data_sid=39baad8c8846551c60aaba081f3bdaa9)](http://s.igmhb.com/click?v=VVM6MTI4MjIzOjIxOTIyOmltcG9ydGluZzo5ZDZjYWQ0ZTllYWZiZmZjZmJmYjVkNTMxOWZhNDQzMTp6LTIyMTctODgyOTY3ODY6aXN0cWJleGFtY2VydGlmaWNhdGlvbi5jb206MzczOTcxOjA6MjE0MTA1N2YxY2JkNDFiM2I0NWJmMTdlY2U4OWRjZjI6MTpkYXRhX3NzLDcyOHgxMzY2O2RhdGFfcmMsMTtkYXRhX2ZiLG5vOzo1ODExNDEyOjo6MC4wNQ&subid=g-88296786-7850546d59644870bdeb01a5ab2c7fae-&data_ss=728x1366&data_rc=1&data_fb=no&data_tagname=A&data_ct=link_only&data_clickel=link&data_sid=39baad8c8846551c60aaba081f3bdaa9) to ensure the Quality of the product.  Quality product delivered to the customers helps in [gaining[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://istqbexamcertification.com/why-is-testing-necessary/#59535978)](http://istqbexamcertification.com/why-is-testing-necessary/#59535978) their confidence.
4. Testing is necessary in [order[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://istqbexamcertification.com/why-is-testing-necessary/#92997528)](http://istqbexamcertification.com/why-is-testing-necessary/#92997528) to provide the facilities to the customers like the delivery of high quality product or software application which requires lower maintenance cost and hence results into more accurate, consistent and reliable results.
5. Testing is required for an effective performance of [software application[http://cdncache-a.akamaihd.net/items/it/img/arrow-10x10.png](http://istqbexamcertification.com/why-is-testing-necessary/#47094992)](http://istqbexamcertification.com/why-is-testing-necessary/#47094992) or product.
6. It’s important to ensure that the application should not result into any [failures](http://istqbexamcertification.com/what-is-a-failure-in-software-testing/) because it can be very expensive in the future or in the later stages of the development.
7. It’s required to stay in the business.

**Entry Criteria**

Entry Criteria for STLC phases can be defined as specific conditions; or, all those documents which are required to start a particular phase of STLC should be present before entering any of the STLC phase.

Entry criteria is a set of conditions that permits a task to perform, or in absence of any of these conditions, the task cannot be performed.

While setting the entry criteria, it is also important to define the time-frame when the entry criteria item is available to start the process.

For Instance, to start the Test Cases development phase, the following conditions should be met −

* The requirement document should be available.
* Complete understanding of the application flow is required.
* The Test Plan Document should be ready.

**Exit Criteria**

Exit Criteria for STLC phases can be defined as items/documents/actions/tasks that must be completed before concluding the current phase and moving on to the next phase.

Exit criteria is a set of expectations; this should be met before concluding the STLC phase.

For Instance, to conclude the Test Cases development phase, following expectations should be met −

* Test Cases should be written and reviewed.
* Test Data should be identified and ready.
* Test automation script should be ready if applicable.

## What is Software Testing Life Cycle (STLC)

Software Testing Life Cycle refers to a testing process which has specific steps to be executed in a definite sequence to ensure that the quality goals have been met. In the STLC process, each activity is carried out in a planned and systematic way. Each phase has different goals and deliverables. Different organizations have different phases in STLC; however, the basis remains the same.

**Below are the phases of STLC:**

1. Requirements phase
2. Planning Phase
3. Analysis phase
4. Design Phase
5. Implementation Phase
6. Execution Phase
7. Conclusion Phase
8. Closure Phase

**#1.** [**Requirement Phase**](https://www.softwaretestinghelp.com/rview-srs-document-and-create-test-scenarios-software-testing-training-course-day-2/)**:**

During this phase of STLC, analyze and study the requirements. Have brainstorming sessions with other teams and try to find out whether the requirements are testable or not. This phase helps to identify the scope of the testing. If any feature is not testable, communicate it during this phase so that the mitigation strategy can be planned.

**#2.** [**Planning Phase**](https://www.softwaretestinghelp.com/how-to-write-test-plan-document-software-testing-training-day3/)**:**

In practical scenarios, Test planning is the first step of the testing process. In this phase, we identify the activities and resources which would help to meet the testing objectives. During planning we also try to identify the metrics, the method of gathering and tracking those metrics.

On what basis the planning is done? Only requirements?

The answer is NO. Requirements do form one of the bases but there are 2 other very important factors which influence test planning. These are:

– Test strategy of the organization.  
– Risk analysis / Risk Management and mitigation.

**#3. Analysis Phase:**

This STLC phase defines “WHAT” to be tested. We basically identify the test conditions through the requirements document, product risks, and other test bases. The test condition should be traceable back to the requirement.

**There are various factors which affect the identification of test conditions:**

– Levels and depth of testing  
– The complexity of the product  
– Product and project risks  
– Software development life cycle involved.  
– Test management  
– Skills and knowledge of the team.  
– Availability of the stakeholders.

We should try to write down the test conditions in a detailed way. For example, for an e-commerce web application, you can have a test condition as “User should be able to make a payment”. Or you can detail it out by saying “User should be able to make payment through NEFT, debit card, and credit card”.

The most important advantage of writing the detailed test condition is that it increases the test coverage since the test cases will be written on the basis of the test condition, these details will trigger to write more detailed test cases which will eventually increase the coverage.

Also, identify the exit criteria of the testing, i.e determine some conditions when you will stop the testing.

**#4. Design Phase:**

This phase defines “HOW” to test. This phase involves the following tasks:

– Detail the test condition. Break down the test conditions into multiple sub-conditions to increase coverage.  
– Identify and get the test data  
– Identify and set up the test environment.  
– Create the requirement traceability metrics  
– Create test coverage metrics.

**#5. Implementation Phase:**

The major task in this STLC phase is of creation of the detailed test cases. Prioritize the test cases also identify which test case will become part of the regression suite. Before finalizing the test case, It is important to carry out the review to ensure the correctness of the test cases. Also, don’t forget to take the sign off of the test cases before actual execution starts.

If your project involves automation, identify the candidate test cases for automation and proceed for scripting the test cases. Don’t forget to review them!

**#6.** [**Execution Phase**](https://www.softwaretestinghelp.com/test-execution-software-testing-qa-training-on-a-live-project-day-5/)**:**

As the name suggests, this is the Software Testing Life Cycle phase where the actual execution takes place. But before you start your execution, make sure that your entry criterion is met. Execute the test cases, log defects in case of any discrepancy. Simultaneously fill your traceability metrics to track your progress.

**#7.** [**Conclusion Phase**](https://www.softwaretestinghelp.com/bug-tracking-test-metrics-and-test-sign-off-free-qa-training-day-6/)**:**

This STLC phase concentrates on the exit criteria and reporting. Depending on your project and stakeholders choice, you can decide on reporting whether you want to send out a daily report of the weekly report, etc.

There are different types of reports ( DSR – Daily status report, WSR – Weekly status reports) which you can send, but the important point is, the content of the report changes and depends upon whom you are sending your reports.

If Project managers belong to testing background then they are more interested in the technical aspect of the project, so include the technical things in your report ( number of test cases passed, failed, defects raised, severity 1 defects, etc.).

But if you are reporting to upper stakeholders, they might not be interested in the technical things so report them about the risks that have been mitigated through the testing.

**#8. Closure Phase:** Tasks for the closure activities include the following: – Check for the completion of the test. Whether all the test cases are executed or mitigated deliberately. Check there is no severity 1 defects opened.  
– Do lessons learned meeting and create lessons learned document. ( Include what went well, where are the scope of improvements and what can be improved)

**TEST PLAN**

|  |  |  |
| --- | --- | --- |
| **S.No.** | **Parameter** | **Description** |
| 1. | Test plan identifier | Unique identifying reference. |
| 2. | Introduction | A brief introduction about the project and to the document. |
| 3. | Test items | A test item is a software item that is the application under test. |
| 4. | Features to be tested | A feature that needs to tested on the testware. |
| 5. | Features not to be tested | Identify the features and the reasons for not including as part of testing. |
| 6. | Approach | Details about the overall approach to testing. |
| 7. | Item pass/fail criteria | Documented whether a software item has passed or failed its test. |
| 8. | Test deliverables | The deliverables that are delivered as part of the testing process,such as test plans, test specifications and test summary reports. |
| 9. | Testing tasks | All tasks for planning and executing the testing. |
| 10. | Environmental needs | Defining the environmental requirements such as hardware, software, OS, network configurations, tools required. |
| 11. | Responsibilities | Lists the roles and responsibilities of the team members. |
| 12. | Staffing and training needs | Captures the actual staffing requirements and any specific skills and training requirements. |
| 13. | Schedule | States the important project delivery dates and key milestones. |
| 14. | Risks and Mitigation | High-level project risks and assumptions and a mitigating plan for each identified risk. |
| 15. | Approvals | Captures all approvers of the document, their titles and the sign off date. |

**Requirement Traceability Matrix (RTM)** is a document that maps and traces user requirement with test cases. It captures all requirements proposed by the client and requirement traceability in a single document, delivered at the conclusion of the Software development life cycle. The main purpose of Requirement Traceability Matrix is to validate that all requirements are checked via test cases such that no functionality is unchecked during Software testing.

**TYPE OF TESTING**

**Functional Testing**

Functional testing is a type of testing which verifies that each function of the software application operates in conformance with the requirement specification. This testing mainly involves black box testing and it is not concerned about the source code of the application. Each and every functionality of the system is tested by providing appropriate input, verifying the output and comparing the actual results with the expected results. This testing involves checking of User Interface, APIs, Database, security, client/ server applications and functionality of the Application Under Test. The testing can be done either manually or using automation

**NON-FUNCTIONAL TESTING**

is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.

**BLACK BOX TESTING**

Black box testing involves testing a system with no prior knowledge of its internal workings. A tester provides an input, and observes the output generated by the system under test. This makes it possible to identify how the system responds to expected and unexpected user actions, its response time, usability issues and reliability issues.

**White Box Testing**

is software testing technique in which internal structure, design and coding of software are tested to verify flow of input-output and to improve design, usability and security. In white box testing, code is visible to testers so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing and Glass box testing.

**Smoke Testing**

is a software testing process that determines whether the deployed software build is stable or not? Smoke testing is a confirmation for QA team to proceed with further software testing. It consists of a minimal set of tests run on each build to test software functionalities. Smoke testing is also known as "Build Verification Testing" or “Confidence Testing.”

**SANITY TESTING**

is a subset of regression testing. Sanity testing is performed to ensure that the code changes that are made are working as properly. Sanity testing is a stoppage to check whether testing for the build can proceed or not. The focus of the team during sanity testing process is to validate the functionality of the application and not detailed testing. Sanity testing is generally performed on build where the production deployment is required immediately like a critical bug fix.

**INTEGRATION TESTING**

is a level of software testing where individual units / components are combined and tested as a group the purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

**Ad hoc Testing**

is an informal or unstructured software testing type that aims to break the testing process in order to find possible defects or errors at an early possible stage. Ad hoc testing is done randomly and it is usually an unplanned activity which does not follow any documentation and test design

**SYSTEM TESTING**

is a level of testing that validates the complete and fully integrated software product. The purpose of a system test is to evaluate the end-to-end system specifications. Usually, the software is only one element of a larger computer-based system. Ultimately, the software is interfaced with other software/hardware systems. System Testing is actually a series of different tests whose sole purpose is to exercise the full computer-based system.

**REGRESSION TESTING**

is defined as a type of software testing to confirm that a recent program or code change has not adversely affected existing features. Regression Testing is nothing but a full or partial selection of already executed test cases which are re-executed to ensure existing functionalities work fine. This testing is done to make sure that new code changes should not have side effects on the existing functionalities. It ensures that the old code still works once the latest code changes are done.

techniques to create test cases.

**PERFORMANCE TESTING**

is a software testing process used for testing the speed, response time, stability, reliability, scalability and resource usage of a software application under particular workload. The main purpose of performance testing is to identify and eliminate the performance bottlenecks in the software application.

**Types of Performance Testing**

* **Load testing -** checks the application's ability to perform under anticipated user loads. The objective is to identify performance bottlenecks before the software application goes live.
* **Stress testing -** involves testing an application under extreme workloads to see how it handles high traffic or data processing. The objective is to identify the breaking point of an application.
* **Endurance testing -** is done to make sure the software can handle the expected load over a long period of time.
* **Spike testing -** tests the software's reaction to sudden large spikes in the load generated by users.
* **Volume testing** - Under Volume Testing large no. of. Data is populated in a database and the overall software system's behavior is monitored. The objective is to check software application's performance under varying database volumes.
* **Scalability testing** - The objective of scalability testing is to determine the software application's effectiveness in "scaling up" to support an increase in user load. It helps plan capacity addition to your software system.