

7/18/2023

DELL

[company name]

**Software testing**

**Manual Testing**

* Software Testing Introduction
* SDLC life cycle
* Software Testing Methodologies
* Test Case Design Techniques
* Levels of Testing
* Software Testing Life Cycle
* QA & QC & Testing
* Agile methodology

**Software Testing Introduction**

Software Testing is a process of evaluating the functionality of a software application to find any software bugs. It checks whether the developed software met the specified requirements and identifies any defect in the software in order to produce a quality product.

****

software testing-software-testing-life-cycle-edureka

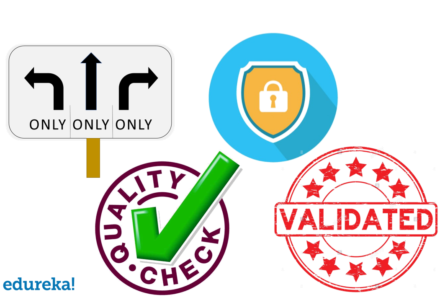
It is also stated as the process of verifying and validating a software product. It checks whether the software product:

* Meets the business and technical requirements that guided its design and development
* Works as per the requirement
* Can be implemented with the same characteristics

Now that we know what is software testing, let’s move ahead with our software testing life cycle blog and see why do we need testing.

**Why is Testing important?**

The importance of software testing is imperative. Software Testing is important because of the following reasons:



* Software Testing points out the defects and errors that were made during the development phases. It looks for any mistake made by the programmer during the implementation phase of the software.
* It ensures that the customer finds the organization reliable and their satisfaction in the application is maintained. Sometimes contracts include monetary penalties with respect to the timeline and quality of the product and software testing prevent monetary losses. Importance of software testing-software-testing-life-cycle
* It also ensures the Quality of the product. Quality product delivered to the customers helps in gaining their confidence. It makes sure that the software application requires lower maintenance cost and results in more accurate, consistent and reliable results.
* Users are not inclined to use software that has bugs. They may not adopt software if they are not happy with the stability of the application. Testing is important for the product to stay in business.
* It’s important to ensure that the application should not result in any failures because it can be very expensive in the future or in the later stages of the development.
* Now that we know what is software testing and why it is important, let’s get into the details of the software testing life cycle and know more about the different phases of testing.

**Objectives**

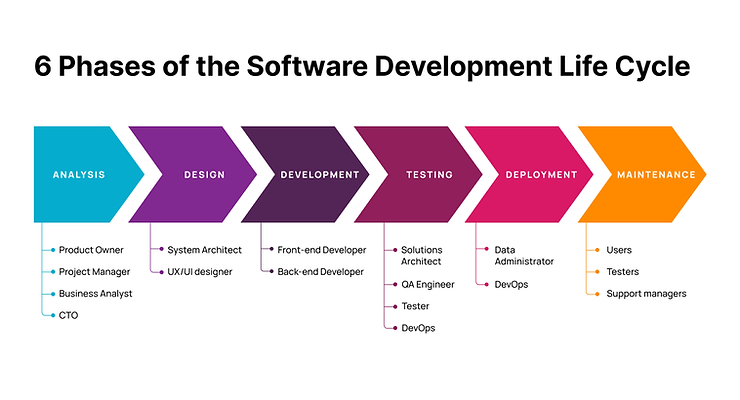
* Uncover as many as errors (or bugs) as possible in a given product.
* Demonstrate a given software product matching its requirement specifications.
* Validate the quality of a software testing using the minimum cost and efforts.
* Generate high quality test cases, perform effective tests, and issue correct and helpful problem reports.

**Error:** It is a human action that produces the incorrect result that produces a fault.

**Bug:** The presence of error at the time of execution of the software.

**Fault:** State of software caused by an error. Failure: Deviation of the software from its expected result. It is an event.

**Failure:** Deviation of the software from its expected result.it is an event.

**SDLC life cycle**

The Software Development Life Cycle (SDLC) is a process used to develop software in a systematic and organized manner. It consists of several phases that guide the software development process from start to finish.

Here's a simple explanation of the SDLC phases with an example:

**Requirements Gathering (Analysis):** In this phase, the project team collects and analyzes the requirements for the software. They interact with stakeholders to understand what the software needs to do and how it should perform. For example, imagine a team is developing a mobile app for a restaurant. They would gather requirements like menu options, order placement, and table reservation features.

**Design:** Once the requirements are known, the design phase begins. Here, the team creates a blueprint of the software, including its architecture, user interface, and database structure. Continuing with the restaurant app example, the design phase would involve creating wireframes or mock-ups of the app screens and designing the database schema.

**Development (Implementation):** In this phase, the actual coding and programming of the software take place. Developers write the code according to the design specifications. For the restaurant app, the development team would write the code for different features like menu display, order processing, and reservation management.

**Testing:** Once the software is implemented, it undergoes testing to identify and fix any defects or issues. Testers execute various test cases to ensure that the software functions correctly and meets the requirements. In the restaurant app scenario, testers would check if the menu items are displayed correctly, orders can be placed successfully, and reservations are properly managed.

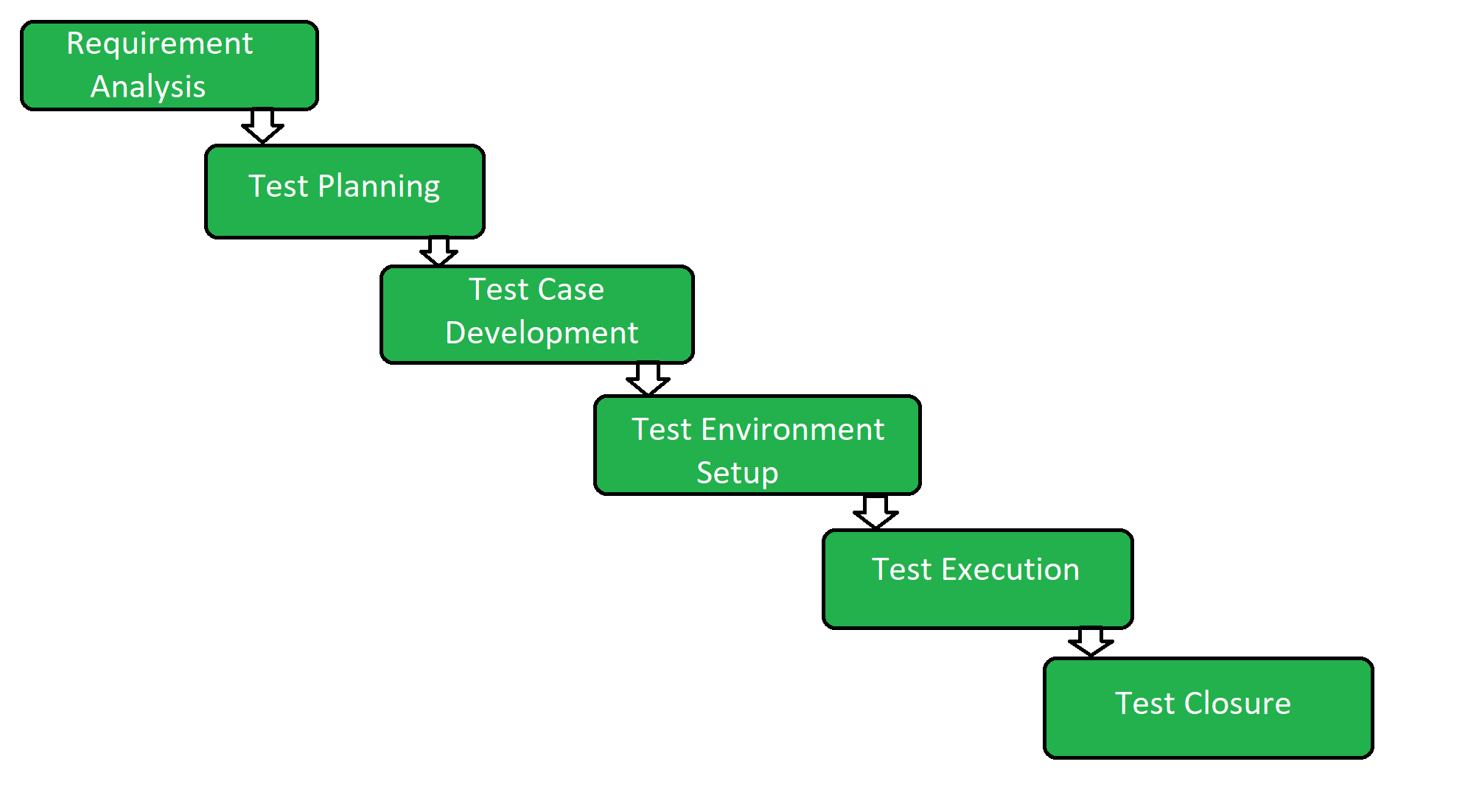
**Deployment:** After the software has been thoroughly tested and approved, it is deployed to the production environment. It is made available to end-users or customers. In the case of the restaurant app, the developers would release it on app stores so that people can download and use it on their mobile devices.

**Maintenance:** Once the software is deployed, it requires regular maintenance and updates. This phase involves fixing any issues that arise, addressing user feedback, and adding new features or improvements over time. For the restaurant app, maintenance could include fixing bugs, adding new menu items, or enhancing the reservation system.

**Software Testing Life Cycle**

Characteristics of STLC

* STLC is a fundamental part of the Software Development Life Cycle (SDLC) but STLC consists of only the testing phases.
* STLC starts as soon as requirements are defined or software requirement document is shared by stakeholders.
* STLC yields a step-by-step process to ensure quality software



**Requirement Analysis:** In this phase, the testing team studies and analyzes the software requirements to understand what the software should do. For example, if you're building a mobile app that tracks fitness activities, the requirement could be to accurately record the number of steps taken by the user.

**Test Planning:** Here, the testing team creates a detailed plan outlining how testing will be conducted. It includes determining what types of tests to perform, what resources are needed, and the test environment. For instance, in our fitness app example, the test plan might include testing the step tracking feature on different smartphones and operating systems.

**Test Case Development:** Test cases are specific instructions or steps to be executed to verify different aspects of the software. Testers create these cases based on the requirements and test plan. In our example, a test case would be to simulate walking with a predefined number of steps and checking if the app accurately records them.

**Test Environment Setup:** This phase involves setting up the necessary hardware, software, and test data to perform the tests. For our fitness app, it would involve installing the app on various devices and creating test accounts with sample data.

**Test Execution:** In this phase, the test cases are executed based on the test plan. Testers run the app, perform the steps outlined in the test cases, and compare the actual results with the expected results. If the recorded step count matches the expected value, the test case passes; otherwise, it fails.

**Test Reporting:** Testers document the test results, including any issues or defects found during testing. They provide detailed information about the failed test cases and communicate them to the development team. For our fitness app, the test report might mention that the app fails to accurately count steps on a specific device.

**Defect Retesting and Closure:** Once the development team fixes the reported defects, testers retest the affected areas to ensure the issues are resolved. If everything works as expected, the testing team verifies the fixes and closes the defects.

**Software Testing Life Cycle Methodologies**

In software testing, there are various methodologies to carry out the software testing processes.

There are four types of methodologies:

* Agile Model
* Waterfall Model
* V Model
* Spiral Model

**Agile Model**

The Agile model is an iterative and incremental approach to software development. It focuses on collaboration, flexibility, and continuous improvement throughout the entire development process. Instead of working in fixed phases, Agile divides the development into smaller, manageable iterations called "sprints." Each sprint usually lasts a few weeks and results in a working increment of the software.

Key aspects of the Agile model:

a. Iterative Development: The project progresses through multiple iterations, with each one adding new features or improving existing ones.

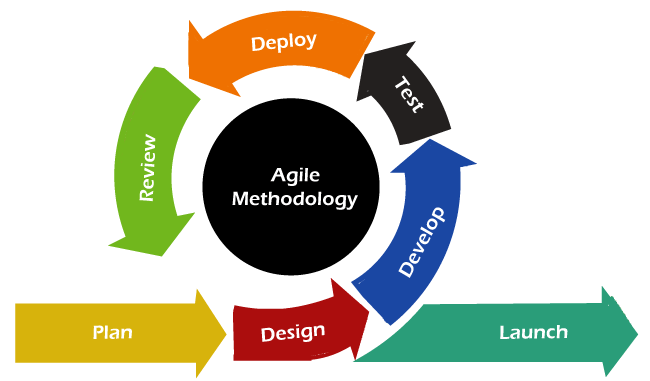
b. Incremental Delivery: Working software is delivered at the end of each sprint, allowing early and continuous feedback from stakeholders.

c. Customer Collaboration: Regular interactions with customers and end-users ensure the delivered product aligns with their needs.

d. Responding to Change: Agile is designed to embrace changes in requirements and adapt accordingly.

Advantages of the Agile model include its ability to accommodate changing requirements, quicker delivery of working software, and enhanced customer satisfaction through continuous involvement. However, it may require more frequent communication and collaboration among team members.

In summary, the Waterfall model is a linear and rigid approach, while the Agile model is iterative, flexible, and emphasizes customer collaboration and continuous improvement. The choice between the two depends on the nature of the project, its requirements, and the team's preferences and capabilities.



**Waterfall model**

The Waterfall model is a traditional, linear approach to software development. It is structured and follows a sequential process, where each phase of the development cycle must be completed before moving on to the next one. The typical phases in the Waterfall model are:

a. Requirements: Gathering and documenting all the project requirements at the beginning.

b. Design: Creating a detailed plan and design based on the gathered requirements.

c. Implementation: The actual coding and development of the software.

d. Testing: Verifying that the software works correctly and meets the specified requirements.

e. Deployment: Releasing the finished product to the users.

f. Maintenance: Providing ongoing support and updates as needed.

Advantages of the Waterfall model include its simplicity and easy-to-understand nature, making it suitable for small projects with well-defined requirements. However, it can be rigid and less adaptable to changes, and the final product might not fully meet users' needs if requirements change during development.

