Role of Testing in SDLC

What is the role of software testing in software development life cycle (SDLC)?

It is the whole process of investigating, planning, executing, and preserving the software product. This cycle continues until software goes through all these processes.

Testing is an essential part of this process. So, in this blog, I am going to discuss the Software Testing Life Cycle (STLC), the roles of Software Testing, and why it is important in the Software Development Life Cycle (SDLC).

Before proceeding further let us first look back at the 3 most common myths about Software Testing that are infamous all across-

1. Software Testing is equal to Quality Control:

This is a great myth that software testing is exactly equal to Quality Control. The actual fact is that Quality Control involves activities such as taking reviews and analyzing them.

2. Testing is considered as the easiest task:

This is absolutely not correct as the testing process involves coding as well. The tester has to write his own SQL queries or scripts to test the software or an application. So, testing is always not an easy task.

3. Anyone can be a Software Tester:

It is completely a hoax as testing requires some extra talent and skills to check and analyze the whole software or an application.

Yes, in small projects people can consider them as a tester but to be proficient and to deliver a valued beneficial product, one must be very skillful.

These are the myths that are most commonly spread and everyone thinks Software Testing is not a complete necessity. To deal with all these myths here are some of the important facts that will make you understand the importance of Software Testing and why it is the necessity of the IT and Software industry.

Software testing also follows a cycle that every QA or test team member follows. This cycle is known as Software Test Life Cycle (STLC).

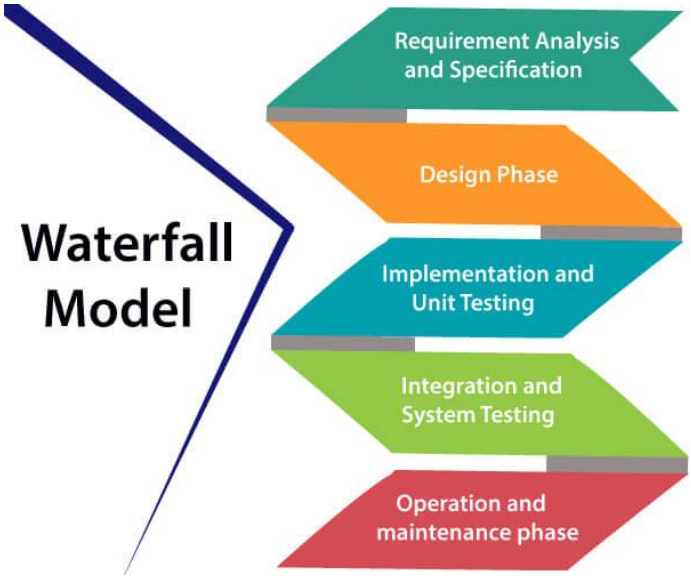
Review of software development models

Waterfall model

Winston Royce introduced the Waterfall Model in 1970.This model has five phases: Requirements analysis and specification, design, implementation, and unit testing, integration and system testing, and operation and maintenance. The steps always follow in this order and do not overlap. The developer must complete every phase before the next phase begins. This model is named "**Waterfall Model**", because its diagrammatic representation resembles a cascade of waterfalls.

1. Requirements analysis and specification phase: The aim of this phase is to understand the exact requirements of the customer and to document them properly. Both the customer and the software developer

work together so as to document all the functions, performance, and interfacing requirement of the software. It describes the "what" of the system to be produced and not "how."In this phase, a large document called **Software Requirement Specification (SRS)** document is created which contained a detailed description of what the system will do in the common language.



2. Design Phase: This phase aims to transform the requirements gathered in the SRS into a suitable form which permits further coding in a programming language. It defines the overall software architecture together with high level and detailed design. All this work is documented as a Software Design Document (SDD).

3. Implementation and unit testing: During this phase, design is implemented. If the SDD is complete, the implementation or coding phase proceeds smoothly, because all the information needed by software developers is contained in the SDD.

During testing, the code is thoroughly examined and modified. Small modules are tested in isolation initially. After that these modules are tested by writing some overhead code to check the interaction between these modules and the flow of intermediate output.

4. Integration and System Testing: This phase is highly crucial as the quality of the end product is determined by the effectiveness of the testing carried out. The better output will lead to satisfied customers, lower maintenance costs, and accurate results. Unit testing determines the efficiency of individual modules. However, in this phase, the modules are tested for their interactions with each other and with the system.

5. Operation and maintenance phase: Maintenance is the task performed by every user once the software has been delivered to the customer, installed, and operational.

When to use SDLC Waterfall Model?

Some Circumstances where the use of the Waterfall model is most suited are:

- When the requirements are constant and not changed regularly.
- A project is short
- The situation is calm
- Where the tools and technology used is consistent and is not changing
- When resources are well prepared and are available to use.

Advantages of Waterfall model

- This model is simple to implement also the number of resources that are required for it is minimal.
- The requirements are simple and explicitly declared; they remain unchanged during the entire project development.
- The start and end points for each phase is fixed, which makes it easy to cover progress.
- The release date for the complete product, as well as its final cost, can be determined before development.
- It gives easy to control and clarity for the customer due to a strict reporting system.

Disadvantages of Waterfall model

- In this model, the risk factor is higher, so this model is not suitable for more significant and complex projects.
- This model cannot accept the changes in requirements during development.
- It becomes tough to go back to the phase. For example, if the application has now shifted to the coding phase, and there is a change in requirement, It becomes tough to go back and change it.
- Since the testing done at a later stage, it does not allow identifying the challenges and risks in the earlier phase, so the risk reduction strategy is difficult to prepare.

Spiral Model

The spiral model, initially proposed by Boehm, is an evolutionary software process model that couples the iterative feature of prototyping with the controlled and systematic aspects of the linear sequential model. It implements the potential for rapid development of new versions of the software. Using the spiral model, the software is developed in a series of incremental releases. During the early iterations, the additional release may be a paper model or prototype. During later iterations, more and more complete versions of the engineered system are produced.

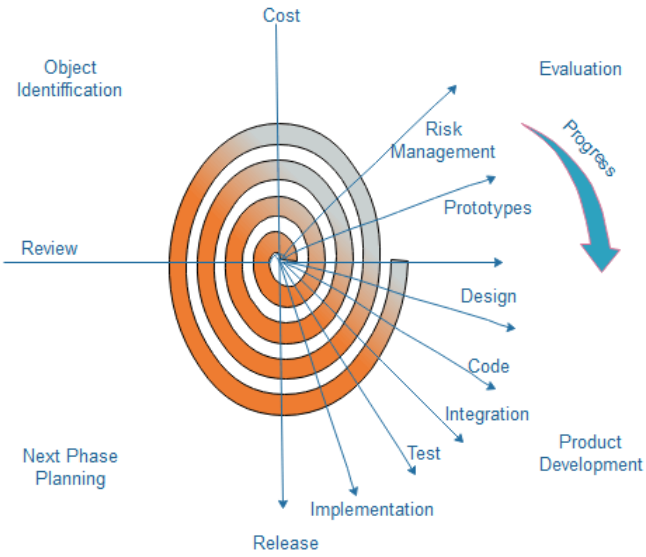


Fig. Spiral Model

Each cycle in the spiral is divided into four parts:

Objective setting: Each cycle in the spiral starts with the identification of purpose for that cycle, the various alternatives that are possible for achieving the targets, and the constraints that exists.

Risk Assessment and reduction: The next phase in the cycle is to calculate these various alternatives based on the goals and constraints. The focus of evaluation in this stage is located on the risk perception for the project.

Development and validation: The next phase is to develop strategies that resolve uncertainties and risks. This process may include activities such as benchmarking, simulation, and prototyping.

Planning: Finally, the next step is planned. The project is reviewed, and a choice made whether to continue with a further period of the spiral. If it is determined to keep, plans are drawn up for the next step of the project.

The development phase depends on the remaining risks. For example, if performance or user-interface risks are treated more essential than the program development risks, the next phase may be an evolutionary development that includes developing a more detailed prototype for solving the risks.

The **risk-driven** feature of the spiral model allows it to accommodate any mixture of a specification-oriented, prototype-oriented, simulation-oriented, or another type of approach. An essential element of the model is that each period of the spiral is completed by a review that includes all the products developed during that cycle, including plans for the next cycle. The spiral model works for development as well as enhancement projects.

When to use Spiral Model?

- When deliverance is required to be frequent.
- When the project is large
- When requirements are unclear and complex
- When changes may require at any time
- Large and high budget projects

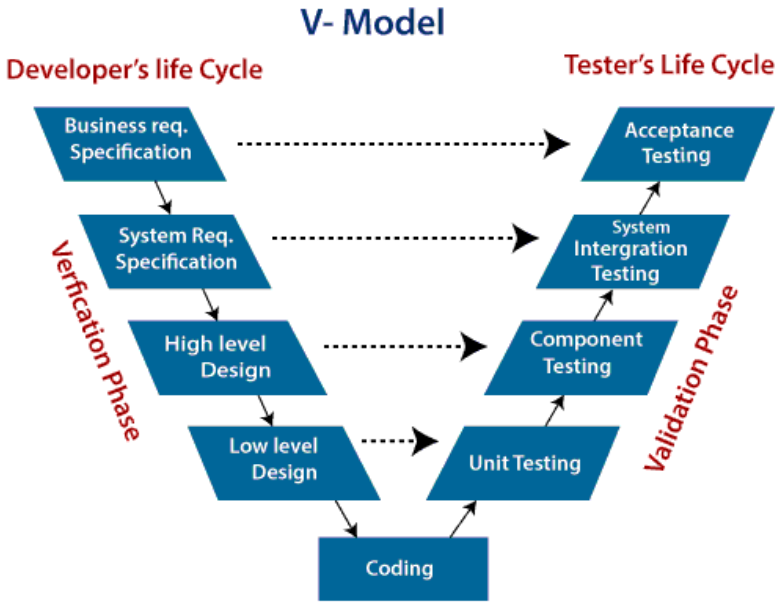
Advantages

- High amount of risk analysis
- Useful for large and mission-critical projects.

Disadvantages

- Can be a costly model to use.
- Risk analysis needed highly particular expertise
- Doesn't work well for smaller projects.

V-Model



V-Model also referred to as the Verification and Validation Model. In this, each phase of SDLC must complete before the next phase starts. It follows a sequential design process same as the waterfall model. Testing of the device is planned in parallel with a corresponding stage of development.

Verification: It involves a static analysis method (review) done without executing code. It is the process of evaluation of the product development process to find whether specified requirements meet.

Validation: It involves dynamic analysis method (functional, non-functional), testing is done by executing code. Validation is the process to classify the software after the completion of the development process to determine whether the software meets the customer expectations and requirements.

So V-Model contains Verification phases on one side of the Validation phases on the other side. Verification and Validation process is joined by coding phase in V-shape. Thus it is known as V-Model.

There are the various phases of Verification Phase of V-model:

1. **Business requirement analysis:** This is the first step where product requirements understood from the customer's side. This phase contains detailed communication to understand customer's expectations and exact requirements.
2. **System Design:** In this stage system engineers analyze and interpret the business of the proposed system by studying the user requirements document.
3. **Architecture Design:** The baseline in selecting the architecture is that it should understand all which typically consists of the list of modules, brief functionality of each module, their interface relationships, dependencies, database tables, architecture diagrams, technology detail, etc. The integration testing model is carried out in a particular phase.
4. **Module Design:** In the module design phase, the system breaks down into small modules. The detailed design of the modules is specified, which is known as Low-Level Design
5. **Coding Phase:** After designing, the coding phase is started. Based on the requirements, a suitable programming language is decided. There are some guidelines and standards for coding. Before checking in the repository, the final build is optimized for better performance, and the code goes through many code reviews to check the performance.

There are the various phases of Validation Phase of V-model:

1. **Unit Testing:** In the V-Model, Unit Test Plans (UTPs) are developed during the module design phase. These UTPs are executed to eliminate errors at code level or unit level. A unit is the smallest entity which can

independently exist, e.g., a program module. Unit testing verifies that the smallest entity can function correctly when isolated from the rest of the codes/ units.

2. **Integration Testing:** Integration Test Plans are developed during the Architectural Design Phase. These tests verify that groups created and tested independently can coexist and communicate among themselves.
3. **System Testing:** System Tests Plans are developed during System Design Phase. Unlike Unit and Integration Test Plans, System Tests Plans are composed by the client?s business team. System Test ensures that expectations from an application developer are met.
4. **Acceptance Testing:** Acceptance testing is related to the business requirement analysis part. It includes testing the software product in user atmosphere. Acceptance tests reveal the compatibility problems with the different systems, which is available within the user atmosphere. It conjointly discovers the non-functional problems like load and performance defects within the real user atmosphere.

When to use V-Model?

- When the requirement is well defined and not ambiguous.
- The V-shaped model should be used for small to medium-sized projects where requirements are clearly defined and fixed.
- The V-shaped model should be chosen when sample technical resources are available with essential technical expertise.

Advantage (Pros) of V-Model:

1. Easy to Understand.
2. Testing Methods like planning, test designing happens well before coding.
3. This saves a lot of time. Hence a higher chance of success over the waterfall model.
4. Avoids the downward flow of the defects.
5. Works well for small plans where requirements are easily understood.

Disadvantage (Cons) of V-Model:

1. Very rigid and least flexible.
2. Not a good for a complex project.
3. Software is developed during the implementation stage, so no early prototypes of the software are produced.
4. If any changes happen in the midway, then the test documents along with the required documents, has to be updated.

W-Model

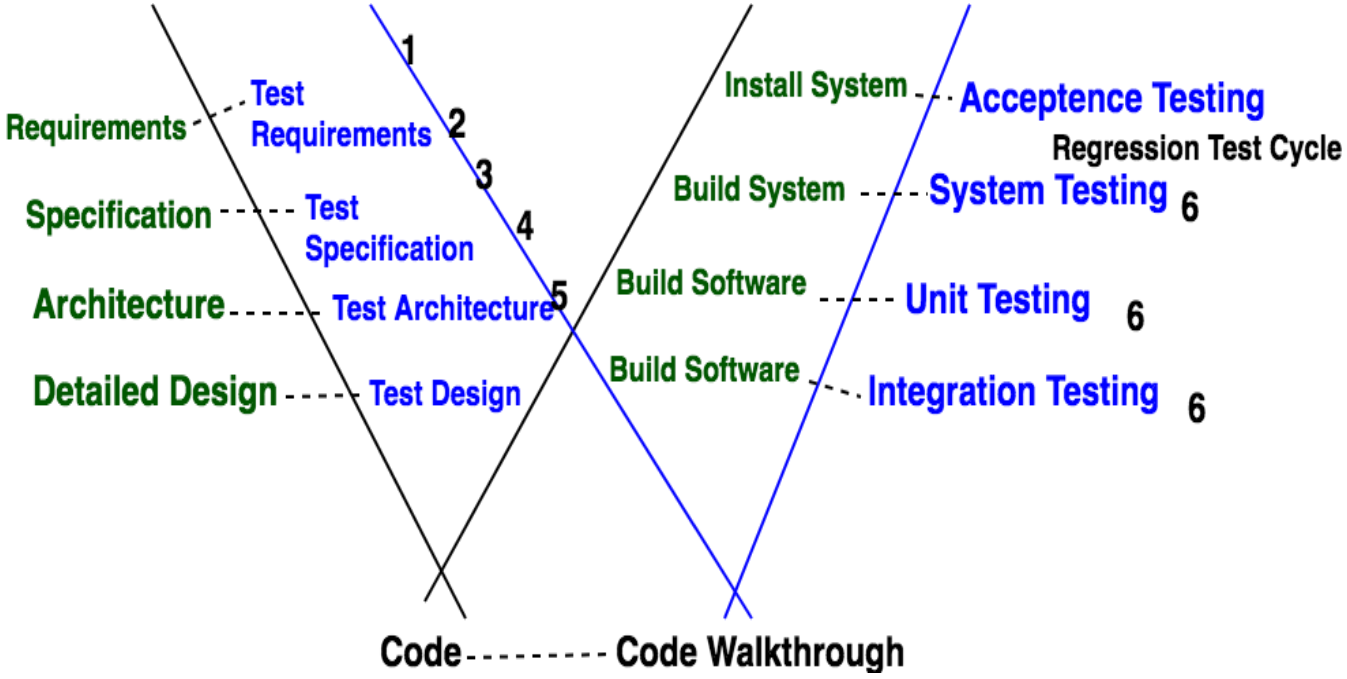
Paul Herzlich introduced W-Model in 1993.

W-model is the most recent software development model where we start real testing activity simultaneously software development process starts. Where as software development process is a

method in which a software or product is made through various stages of planning, development and testing before the final software or product is delivered. testing is such a stage that is extremely crucial to ensure the delivery of an optimum quality product.

.

- V-model and W-model are two of the most important models that are used in software testing.
- W-Model covers those activities which are skipped by V-Model and also, it deals with problems which couldn't be catch by V-Model.
- W-Model approach attempts to address and tackle the shortcomings W-Model approach attempts to address and tackle the shortcomings of V-Model.
- W-model can be done only once the development of the product is complete with no modifications required to be done in between. This type of testing is most suitable for short projects.
- With the help of W-Model, we ensure that the testing of the product starts from the very first day of the inception of product and each phase of the product development is verified and validated.



Phases of W-Model:
Each phase is verified/validated. Dotted line shows that every phase in green is validated/tested through every phase in sky blue. Now, in the above figure,

- Point 1 refers to – Build Test Plan & Test Strategy.
- Point 2 refers to – Scenario Identification.
- Point 3 refers to –Test case preparation from Specification document and design documents.
- Point 4 refers to – Test case preparation from Specification document and design documents.
- Point 5 refers to – review of test cases and update as per the review comments.
- Point 6 refers to – Various testing methodologies such as Unit/integration testing, path testing, equivalence partition, boundary value, specification based testing, security testing, usability testing, performance testing.
- After this, there are regression test cycles and then User acceptance testing.

Testing Techniques Used in W-Model:

1. Regression Testing
2. Static Testing:
   Static Testing is further divided into two parts:
   a. Review
   b. Static Analysis
3. Dynamic Testing

Advantages of W-Model:

- In W-Model there is no strict division between constructive tasks on the left-hand side and the more destructive tasks on the right-hand side.
- During the test phase, the developer is responsible for the removal of defects and the correction of the implementation.
- Emphasis the fact that testing is more than just construction, execution and evaluation of test cases.
- The importance of the tests and the ordering of the individual activities for testing are clear.

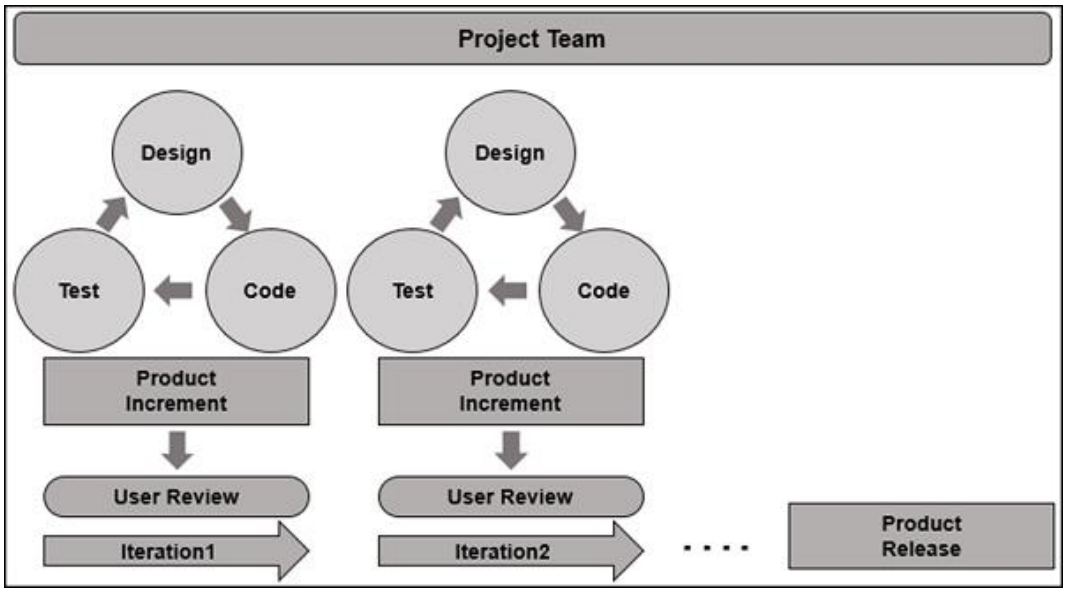
Disadvantages Of W-Model:

- The real facts are simplified in this model.
- There is a need for a simple model if all people involved in a project are to accept it.
- For highly critical applications the test activities certainly have higher weighting or at least equal weighting with other activities.

Agile Testing - Methodologies

Agile is an iterative development methodology, where the entire project team participates in all the activities. The requirements evolve as the iterations progress, through collaboration between the customer and the self-organizing teams. As Coding and Testing are done interactively and incrementally, during the course of development, the end-product would be of quality and ensures customer requirements.

Every iteration results in an integrated working product increment and is delivered for User Acceptance Testing. The customer feedback thus obtained would be an input to the next / subsequent Iterations.



Continuous Integration, Continuous Quality

Continuous Integration is the key for Agile Development success. Integrate frequently, at least daily such that you are ready for a release as and when required. Testing in Agile becomes an essential component of all the phases of the development, ensuring continuous quality of the product. Constant feedback from everyone involved in the project adds to the quality of the product.

In Agile, communication is given utmost importance and the customer requests are received as and when necessary. This gives the satisfaction to the customer that all the inputs are considered and working quality product is available throughout the development.

Agile Methodologies

There are several Agile Methodologies that support Agile Development. The Agile Methodologies include −

Scrum

Scrum is an Agile development method that emphasizes on team-centric approach. It advocates participation of the entire team in all the project development activities.

XP

eXtreme Programming is customer-centric and focuses on constantly changing requirements. With frequent releases and customer feedback, the end-product will be of quality meeting customer requirements that are made clearer during the process.

Crystal

Crystal is based on chartering, cyclic delivery and wrap up.

- Chartering involves forming a development team, carrying out a preliminary feasibility analysis, arriving at an initial plan and the development methodology.
- Cyclic Delivery with two or more delivery cycles focuses on the development phase and final integrated product delivery.
- During Wrap up, deployment into the user environment, post-deployment reviews and reflections are performed.

FDD :-Feature Driven Development (FDD) involves designing and building features. The difference between FDD and other Agile Development Methodologies is that the features are developed in specific and short phases separately.

DSDM

Dynamic Software Development Method (DSDM) is based on Rapid Application Development (RAD) and is aligned to the Agile Framework. DSDM focuses on frequent delivery of the product, involving users actively and empowering the teams to make quick decisions.

Lean Software Development

In Lean Software Development, focus is on eliminating waste and giving value to the customer. This results in rapid development and product of value.

Waste includes partially done work, irrelevant work, features that are not used by the customer, defects, etc. that add to delays in delivery.

The **Lean Principles** are −

- Eliminate Waste
- Amplify Learning
- Delay Commitment
- Empower the Team

- Deliver Fast
- Build Integrity in
- See the Whole

Kanban

Kanban focuses on managing work with an emphasis on just-in-time (JIT) delivery, while not overloading the team members. The tasks are displayed for all the participants to see and for the Team Members to pull work from a queue.

Kanban is based on −

- Kanban Board (Visual and Persistent across the Development)
- Work-in-progress (WIP) Limit
- Lead Time

Agile Testing Methodologies

The testing practices are well defined for every project, whether Agile or not, to deliver quality products. Traditional Testing principles are quite often used in Agile Testing. One of them is Early Testing that focuses on −

- Writing Test Cases to express the behavior of the system.
- Early Defect Prevention, detection and removal.
- Ensuring that the right test types are run at the right time and as part of the right test level.

In all the Agile Methodologies we discussed, Agile Testing in itself is a Methodology. In all the approaches, Test Cases are written before Coding.

In this tutorial, we will focus on Scrum as the Agile Testing Methodology.

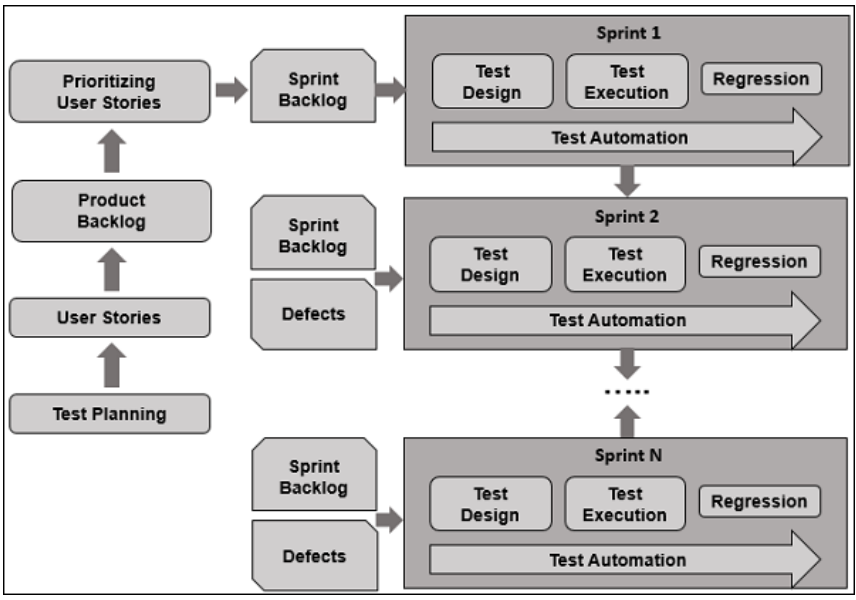
The other commonly used Agile Testing Methodologies are −

- **Test-Driven Development (TDD)** − Test-Driven Development (TDD) is based on coding guided by tests.
- **Acceptance Test-Driven Development (ATDD)** − Acceptance Test-Driven Development (ATDD) is based on communication between the customers, developers and testers and driven by pre-defined Acceptance Criteria and Acceptance Test Cases.
- **Behavior-Driven Development (BDD)** − In Behavior-Driven Development (BDD) testing is based on the expected behavior of the software being developed.

Agile Testing Lifecycle

In Scrum, the Testing activities include −

- Contributing to User Stories based on the expected behavior of the System depicted as Test Cases
- Release Planning based on Test Effort and Defects
- Sprint Planning based on User Stories and Defects
- Sprint Execution with Continuous Testing
- Regression Testing after the completion of Sprint
- Reporting Test Results
- Automation Testing



Impact Of Agile Methodology On Testing

Agile testing is an integral part of the agile methodology. Agile runs continuously and simultaneously with development and is a collaborative effort between testers, developers, product owners and even customers. Unlike traditional testing, Agile testing is not an individual stage, where the testers run the test life cycle. In this evolution process, the testers are not part of the QA team, but they are part of the scrum team.

They not only test the product but closely work with Product Owners, BA, Automation Engineers, Developers throughout the development cycle. Agile testing is not only about testing the product but should be capable to do more than what traditional testers do, they should be dynamic and highly interactive. They should have highly adaptable and flexible during the process to quickly accept the changes and adjust accordingly. Agile testing is not only a process or guideline, but it is a practice or mindset which need to adhere with dedication and an ambitious mentality

Levels of Testing

In this section, we are going to understand the various **levels of software testing**.

As we learned in the earlier section of the software testing tutorial that testing any application or software, the test engineer needs to follow multiple testing techniques.

In order to detect an error, we will implement software testing; therefore, all the errors can be removed to find a product with more excellent quality.

What are the levels of Software Testing?

Testing levels are the procedure for finding the missing areas and avoiding overlapping and repetition between the development life cycle stages. We have already seen the various phases such as **Requirement collection, designing, coding testing, deployment, and maintenance** of SDLC (Software Development Life Cycle)

In order to test any application, we need to go through all the above phases of SDLC. Like SDLC, we have multiple levels of testing, which help us maintain the quality of the software.

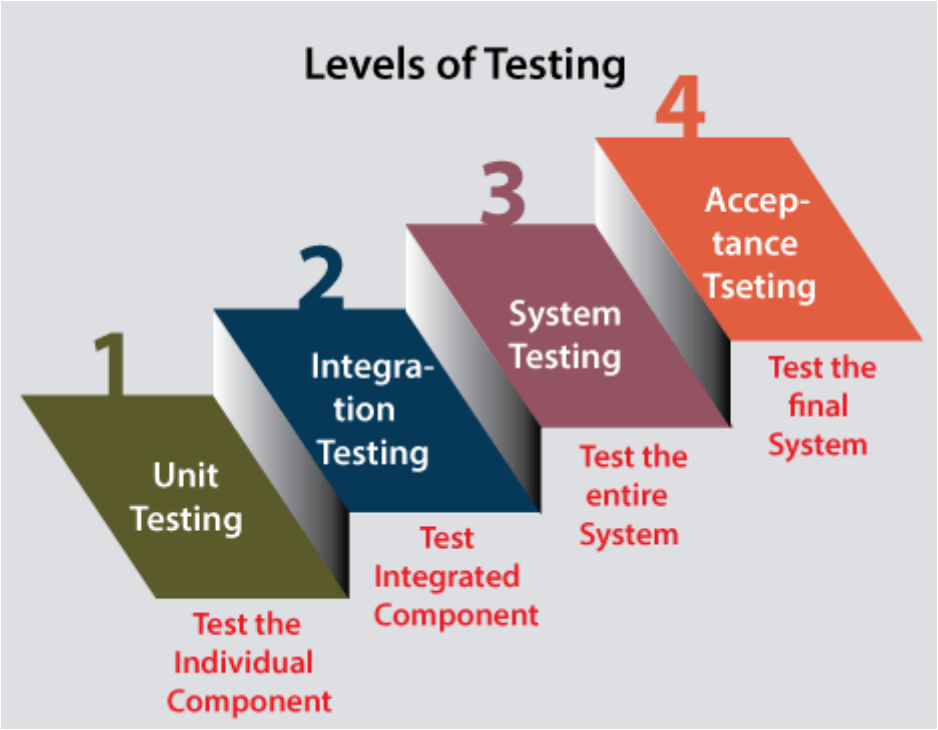
Different Levels of Testing

The levels of software testing involve the different methodologies, which can be used while we are performing the software testing.

In software testing

, we have four different levels of testing, which are as discussed below:

1. **Unit Testing**
2. **Integration Testing**
3. **System Testing**
4. **Acceptance Testing**



As we can see in the above image that all of these testing levels have a specific objective which specifies the value to the software development lifecycle.

For our better understanding, let's see them one by one:

Level1: Unit Testing

Unit testing is the first level of software testing, which is used to test if software modules are satisfying the given requirement or not.

The first level of testing involves **analyzing each unit or an individual component** of the software application.

Unit testing is also the first level of **functional testing**

. The primary purpose of executing unit testing is to validate unit components with their performance.

A unit component is an individual function or regulation of the application, or we can say that it is the smallest testable part of the software. The reason of performing the unit testing is to test the correctness of inaccessible code.

Unit testing will help the test engineer and developers in order to understand the base of code that makes them able to change defect causing code quickly. The developers implement the unit..

Level2: Integration Testing

The second level of software testing is the **integration testing.** The integration testing process comes after **unit testing**.

It is mainly used to test the **data flow from one module or component to other modules.**

In integration testing, the **test engineer** tests the units or separate components or modules of the software in a group.

The primary purpose of executing the integration testing is to identify the defects at the interaction between integrated components or units.

When each component or module works separately, we need to check the data flow between the dependent modules, and this process is known as **integration testing**.

We only go for the integration testing when the functional testing has been completed successfully on each application module.

In simple words, we can say that **integration testing** aims to evaluate the accuracy of communication among all the modules.

Level3: System Testing

The third level of software testing is **system testing**, which is used to test the software's functional and non-functional requirements.

It is **end-to-end testing** where the testing environment is parallel to the production environment. In the third level of software testing, **we will test the application as a whole system.**

To check the end-to-end flow of an application or the software as a user is known as **System testing**.

In system testing, we will go through all the necessary modules of an application and test if the end features or the end business works fine, and test the product as a complete system.

In simple words, we can say that System testing is a sequence of different types of tests to implement and examine the entire working of an integrated software computer system against requirements.

Level4: Acceptance Testing

The **last and fourth level** of software testing is **acceptance testing**, which is used to evaluate whether a specification or the requirements are met as per its delivery.

The software has passed through three testing levels (**Unit Testing, Integration Testing, System Testing**). Some minor errors can still be identified when the end-user uses the system in the actual scenario.

In simple words, we can say that Acceptance testing is the **squeezing of all the testing processes that are previously done.**

The acceptance testing is also known as **User acceptance testing (UAT)** and is done by the customer before accepting the final product.

Usually, UAT is done by the domain expert (customer) for their satisfaction and checks whether the application is working according to given business scenarios and real-time scenarios.