**TESTING**

**SOFTWARE**

Software is a set of instructions, data or programs used to operate computers and execute specific tasks

It is a collection of computer programs

**SOFTWARE TESTING**

Software testing can be stated as the process of verifying and validating that a software or application is bug free, meets the technical requirements.

**SDLC STEPS**

1. [Requirements & Analysis](https://www.emergentsoftware.net/blog/the-7-stages-of-the-software-development-life-cycle-sdlc/#stage1)
2. [Project Planning](https://www.emergentsoftware.net/blog/the-7-stages-of-the-software-development-life-cycle-sdlc/#stage2)
3. [Design](https://www.emergentsoftware.net/blog/the-7-stages-of-the-software-development-life-cycle-sdlc/#stage3)
4. [Coding & Implementation](https://www.emergentsoftware.net/blog/the-7-stages-of-the-software-development-life-cycle-sdlc/#stage4)
5. [Testing](https://www.emergentsoftware.net/blog/the-7-stages-of-the-software-development-life-cycle-sdlc/#stage5)
6. [Deployment](https://www.emergentsoftware.net/blog/the-7-stages-of-the-software-development-life-cycle-sdlc/#stage6)
7. [Maintenance](https://www.emergentsoftware.net/blog/the-7-stages-of-the-software-development-life-cycle-sdlc/#stage7)

**MODELS**

**WATERFALL MODEL** is a straightforward and basic structure which can be easily understandable by software developers and testers

**SPIRAL MODEL** Spiral Model helps to adopt software development elements of multiple process models for the software project based on unique risk patterns ensuring efficient development process.

**V SHAPE MODEL** In [software development](https://en.wikipedia.org/wiki/Software_development), the **V-model**[[2]](https://en.wikipedia.org/wiki/V-Model_(software_development)#cite_note-2) represents a [development process](https://en.wikipedia.org/wiki/Software_development_process) that may be considered an extension of the [waterfall model](https://en.wikipedia.org/wiki/Waterfall_model), and is an example of the more [general V-model](https://en.wikipedia.org/wiki/V-model).

**INCREMENTAL MODEL (ITERATIVE)**

 Incremental Model is a process of software development where requirements are broken down into multiple standalone modules of software development cycle. Incremental development is done in steps from analysis design, implementation, testing/verification, maintenance.

**AGILE** is a framework that defines how software development needs to be done. It is not a single or specific method, and it is the collection of various methodologies and best practices that follow the value statement signed with the customer

**SCRUM**

Scrum is a structured framework for product development that is frequently used by agile software development teams.

**BUGS, DEFECTS AND FAILURES**

The **bug** means that software or application is not working as per the requirement.

A **defect** is an error found during the development phase whereas a **bug** is a defect found during testing phase. Not all programming errors are detectable. Hence not all errors cause a defect.

A **failure** is when the system does not meet its requirements. It is an error that reached a user.

**VERIFICATION AND VALIDATION**

**Verification** is the process of determining if the software meets standards. Are we building the product, right?

**Validation** is the process of determining if software satisfies the expectations and needs of the client. Are we building the right product?

**QUALITY CONTROL AND QUALITY ASSURANCE**

Quality control systems measure parts, including the outputs of the system. QC efforts may also be focused on parts used to create the final product, such as raw materials from a supplier.

Quality assurance control systems are the methods and procedures which are used to safeguard quality standards

**DIFFERENT SOFTWARE TESTING TECHNIQUES**

Software testing techniques can be majorly classified into two categories:

**Black Box Testing:** The technique of testing in which the tester does not have access to the source code of the software and is conducted at the software interface without any concern with the internal logical structure of the software is known as black-box testing.

**White-Box Testing:** The technique of testing in which the tester is aware of the internal workings of the product, has access to its source code, and is conducted by making sure that all internal operations are performed according to the specifications is known as white box testing.

**DIFFERENT LEVELS OF SOFTWARE TESTING**

**Unit tests**

Unit tests are very low level and close to the source of an application. They consist in testing individual methods and functions of the classes, components, or modules used by your software. Unit tests are generally quite cheap to automate and can run very quickly by a continuous integration server.

**Integration tests**

Integration tests verify that different modules or services used by your application work well together. For example, it can be testing the interaction with the database or making sure that microservices work together as expected. These types of tests are more expensive to run as they require multiple parts of the application to be up and running.

**Functional tests**

Functional tests focus on the business requirements of an application. They only verify the output of an action and do not check the intermediate states of the system when performing that action.

There is sometimes a confusion between integration tests and functional tests as they both require multiple components to interact with each other. The difference is that an integration test may simply verify that you can query the database while a functional test would expect to get a specific value from the database as defined by the product requirements.

**End-to-end tests**

End-to-end testing replicates a user behavior with the software in a complete application environment. It verifies that various user flows work as expected and can be as simple as loading a web page or logging in or much more complex scenarios verifying email notifications, online payments, etc...

End-to-end tests are very useful, but they're expensive to perform and can be hard to maintain when they're automated. It is recommended to have a few key end-to-end tests and rely more on lower level types of testing (unit and integration tests) to be able to quickly identify breaking changes.

**Acceptance testing**

Acceptance tests are formal tests that verify if a system satisfies business requirements. They require the entire application to be running while testing and focus on replicating user behaviors. But they can also go further and measure the performance of the system and reject changes if certain goals are not met.

**Performance testing**

Performance tests evaluate how a system performs under a particular workload. These tests help to measure the reliability, speed, scalability, and responsiveness of an application. For instance, a performance test can observe response times when executing a high number of requests, or determine how a system behaves with a significant amount of data. It can determine if an application meets performance requirements, locate bottlenecks, measure stability during peak traffic, and more.

**Smoke testing**

Smoke tests are basic tests that check the basic functionality of an application. They are meant to be quick to execute, and their goal is to give you the assurance that the major features of your system are working as expected.

Smoke tests can be useful right after a new build is made to decide whether you can run more expensive tests, or right after a deployment to make sure that they application is running properly in the newly deployed environment.

**Sanity Testing**

Sanity testing is a**subset of regression testing**. After receiving the software build, sanity testing is performed to ensure that the code changes introduced are working as expected. This testing is a checkpoint to determine if testing for the build can proceed or no

**Exploratory testing**

The more features and improvements go into your code, the more you'll need to test to make sure that all your system works properly. And then for each bug you fix, it would be wise to check that they don't get back in newer releases. Automation is key to make this possible and writing tests sooner or later will become part of your development workflow.