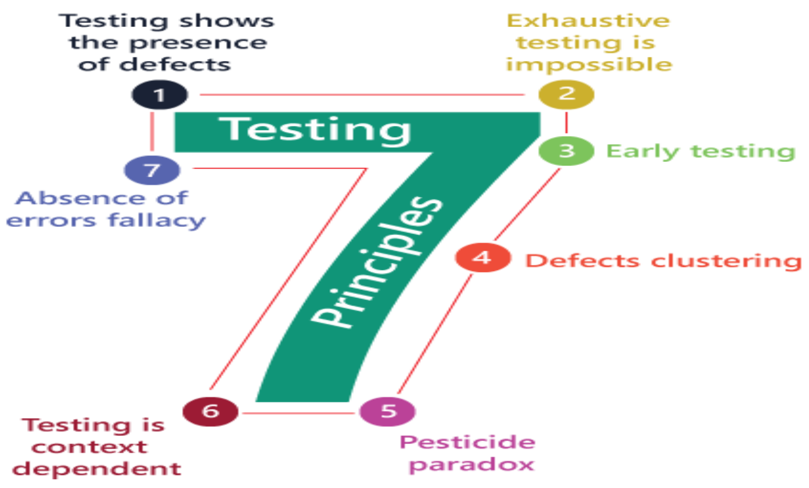
Software Testing Principles

Software testing is a procedure of implementing software or the application to identify the defects or bugs. For testing an application or software, we need to follow some principles to make our product defects free, and that also helps the test engineers to test the software with their effort and time. Here, in this section, we are going to learn about the seven essential principles of software testing.

Let us see the seven different testing principles, one by one:

- Testing shows the presence of defects
- Exhaustive Testing is not possible
- Early Testing
- Defect Clustering
- Pesticide Paradox
- Testing is context-dependent
- Absence of errors fallacy


Testing shows the presence of defects
1
Exhaustive testing is impossible
2
Testing
3
Early testing
7
Absence of errors fallacy
Principles
4
Defects clustering
6
5
Testing is context dependent
Pesticide paradox

Testing shows the presence of defects

The test engineer will test the application to make sure that the application is bug or defects free. While doing testing, we can only identify that the application or software has any errors. The primary purpose of doing testing is to identify the numbers of unknown bugs with the help of various methods and testing techniques because the entire test should be traceable to the customer requirement, which means that to find any defects that might cause the product failure to meet the client's needs.

By doing testing on any application, we can decrease the number of bugs, which does not mean that the application is defect-free because sometimes the software seems to be bug-free while performing multiple types of testing on it. But at the time of deployment in the production server, if the end-user encounters those bugs which are not found in the testing process.

Exhaustive Testing is not possible

Sometimes it seems to be very hard to test all the modules and their features with effective and non- effective combinations of the inputs data throughout the actual testing process.

Hence, instead of performing the exhaustive testing as it takes boundless determinations and most of the hard work is unsuccessful. So we can complete this type of variations according to the importance of the modules because the product timelines will not permit us to perform such type of testing scenarios.

Early Testing

Here early testing means that all the testing activities should start in the early stages of the software development life cycle's **requirement analysis stage** to identify the defects because if we find the bugs at an early stage, it will be fixed in the initial stage itself, which may cost us very less as compared to those which are identified in the future phase of the testing process.

To perform testing, we will require the requirement specification documents; therefore, if the requirements are defined incorrectly, then it can be fixed directly rather than fixing them in another stage, which could be the development phase.

Defect clustering

The defect clustering defined that throughout the testing process, we can detect the numbers of bugs which are correlated to a small number of modules. We have various reasons for this, such as the modules could be complicated; the coding part may be complex, and so on.

These types of software or the application will follow the **Pareto Principle**, which states that we can identify that approx. Eighty percent of the complication is present in 20 percent of the modules. With the help of this, we can find the uncertain modules, but this method has its difficulties if the same tests are performing regularly, hence the same test will not able to identify the new defects.

Pesticide paradox

This principle defined that if we are executing the same set of test cases again and again over a particular time, then these kinds of the test will not be able to find the new bugs in the software or the application. To get over these pesticide paradoxes, it is very significant to review all the test cases frequently. And the new and different tests are necessary to be written for the implementation of multiple parts of the application or the software, which helps us to find more bugs.

Testing is context-dependent

Testing is a context-dependent principle states that we have multiple fields such as e-commerce websites, commercial websites, and so on are available in the market. There is a definite way to test the commercial site as well as the e-commerce websites because every application has its own needs, features, and functionality. To check this type of application, we will take the help of various kinds of testing, different technique, approaches, and multiple methods. Therefore, the testing depends on the context of the application.

Absence of errors fallacy

Once the application is completely tested and there are no bugs identified before the release, so we can say that the application is 99 percent bug-free. But there is the chance when the application is tested beside the incorrect requirements, identified the flaws, and fixed them on a given period would not help as testing is done on the wrong specification, which does not apply to the client's requirements. The absence of error fallacy means identifying and fixing the bugs would not help if the application is impractical and not able to accomplish the client's requirements and needs.

Role of Software Testing in software quality

Software testing is an **important process** in the **software development lifecycle** . It involves **verifying** and **validating** that a **software application** is free of bugs, meets the technical requirements set by its **design** and **development** , and satisfies user requirements efficiently and effectively.

This process ensures that the application can handle all exceptional and boundary cases, providing a robust and reliable user experience. By systematically identifying and fixing issues, software testing helps deliver high-quality software that performs as expected in various scenarios.

The process of software testing aims not only at finding faults in the existing software but also at finding measures to improve the software in terms of efficiency, accuracy, and usability. The article focuses on discussing Software Testing in detail.
It is important to learn the basics whether it is software testing or anything else you always need to start from the basics and make your foundation strong and then proceed to the advance level. If you want to learn software testing in a structure approach then you can enroll in our manual to automation testing course.

What is Software Testing?

Software Testing is a method to assess the functionality of the software program. The process checks whether the actual software matches the expected requirements and ensures the software is bug-free. The purpose of software testing is to identify the errors, faults, or missing requirements in contrast to actual requirements. It mainly aims at measuring the specification, functionality, and performance of a software program or application.
Perform end-to-end test automation, including AI-powered codeless testing, mobile app, cross-browser, visual UI testing, and more with TestGrid . It is a highly secure and scalable software testing tool that offers extensive integration with CI/CD pipelines for continuous testing.

Software testing can be divided into two steps

1. **Verification:** It refers to the set of tasks that ensure that the software correctly implements a specific function. It means “Are we building the product right?”.
2. **Validation:** It refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements. It means “Are we building the right product?”.
3.

Importance of Software Testing

- **Defects can be identified early:** Software testing is important because if there are any bugs they can be identified early and can be fixed before the delivery of the software.
- **Improves quality of software:** Software Testing uncovers the defects in the software, and fixing them improves the quality of the software.
- **Increased customer satisfaction:** Software testing ensures reliability, security, and high performance which results in saving time, costs, and customer satisfaction.
- **Helps with scalability:** Software testing type non-functional testing helps to identify the scalability issues and the point where an application might stop working.
- **Saves time and money:** After the application is launched it will be very difficult to trace and resolve the issues, as performing this activity will incur more costs and time. Thus, it is better to conduct software testing at regular intervals during software development.

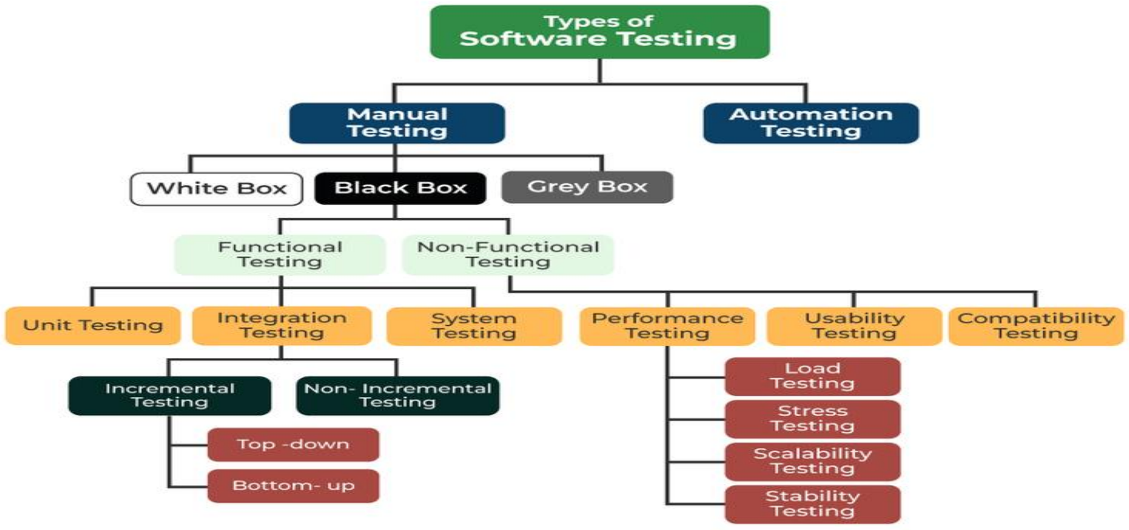
Need for Software Testing

Software bugs can cause potential monetary and human loss. There are many examples in history that clearly depicts that without the testing phase in software development lot of damage was incurred. Below are some examples:

- **1985:** Canada's Therac-25 radiation therapy malfunctioned due to a software bug and resulted in lethal radiation doses to patients leaving 3 injured and 3 people dead.
- **1994:** China Airlines Airbus A300 crashed due to a software bug killing 264 people.
- **1996:** A software bug caused U.S. bank accounts of 823 customers to be credited with 920 million US dollars.
- **1999:** A software bug caused the failure of a $1.2 billion military satellite launch.
- **2015:** A software bug in fighter plan F-35 resulted in making it unable to detect targets correctly.
- **2015:** Bloomberg terminal in London crashed due to a software bug affecting 300,000 traders on the financial market and forcing the government to postpone the 3bn pound debt sale.
- Starbucks was forced to close more than 60% of its outlet in the U.S. and Canada due to a software failure in its POS system.
- Nissan cars were forced to recall 1 million cars from the market due to a software failure in the car's airbag sensory detectors.

Different Types Of Software Testing

Explore diverse software testing methods including manual and automated testing for improved quality assurance . Enhance software reliability and performance through functional and non-functional testing, ensuring user satisfaction. Learn about the significance of various testing approaches for robust software development.



Types Of Software Testing

Software Testing can be broadly classified into 3 types:

1. **Functional testing** : It is a type of software testing that validates the software systems against the functional requirements. It is performed to check whether the application is working as per the software's functional requirements or not. Various types of functional testing are Unit testing, Integration testing, System testing, Smoke testing, and so on.

2. **Non-functional testing** : It is a type of software testing that checks the application for non-functional requirements like performance, scalability, portability, stress, etc. Various types of non-functional testing are Performance testing, Stress testing, Usability Testing, and so on.
3. **Maintenance testing** **:** It is the process of changing, modifying, and updating the software to keep up with the customer's needs. It involves **regression testing** that verifies that recent changes to the code have not adversely affected other previously working parts of the software.

Apart from the above classification software testing can be further divided into 2 more ways of testing:

1. **Manual testing** : It includes testing software manually, i.e., without using any automation tool or script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug. There are different stages for manual testing such as unit testing, integration testing, system testing, and user acceptance testing. Testers use test plans, test cases, or test scenarios to test software to ensure the completeness of testing. Manual testing also includes exploratory testing, as testers explore the software to identify errors in it.
2. **Automation testing** : It is also known as Test Automation, is when the tester writes scripts and uses another software to test the product. This process involves the automation of a manual process. Automation Testing is used to re-run the test scenarios quickly and repeatedly, that were performed manually in manual testing.

Apart from **Regression testing** , **Automation testing** is also used to test the application from a load, performance, and stress point of view. It increases the test coverage, improves accuracy, and saves time and money when compared to manual testing.

Different Types of Software Testing Techniques

Software testing techniques can be majorly classified into two categories:

1. **Black box Testing** : Testing in which the tester doesn't 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 known as black-box testing.
2. **White box Testing** : 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.
3. **Grey Box Testing** : Testing in which the testers should have knowledge of implementation, however, they need not be experts.

S No.	Black Box Testing	White Box Testing
1	Internal workings of an application are not required.	Knowledge of the internal workings is a must.
2	Also known as closed box/data-driven testing.	Also known as clear box/structural testing.
3	End users, testers, and	Normally done by testers

S No.	Black Box Testing	White Box Testing
	developers.	and developers.
4	This can only be done by a trial and error method.	Data domains and internal boundaries can be better tested.

Different Levels of Software Testing

Software level testing can be majorly classified into 4 levels:

1. **Unit testing** : It a level of the software testing process where individual units/components of a software/system are tested. The purpose is to validate that each unit of the software performs as designed.
2. **Integration testing** : It is a level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.
3. **System testing** : It is a level of the software testing process where a complete, integrated system/software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements.
4. **Acceptance testing** : It is a level of the software testing process where a system is tested for acceptability. The purpose of this test is to evaluate the system's compliance with the business requirements and assess whether it is acceptable for delivery.