Unit-4

Software Testing

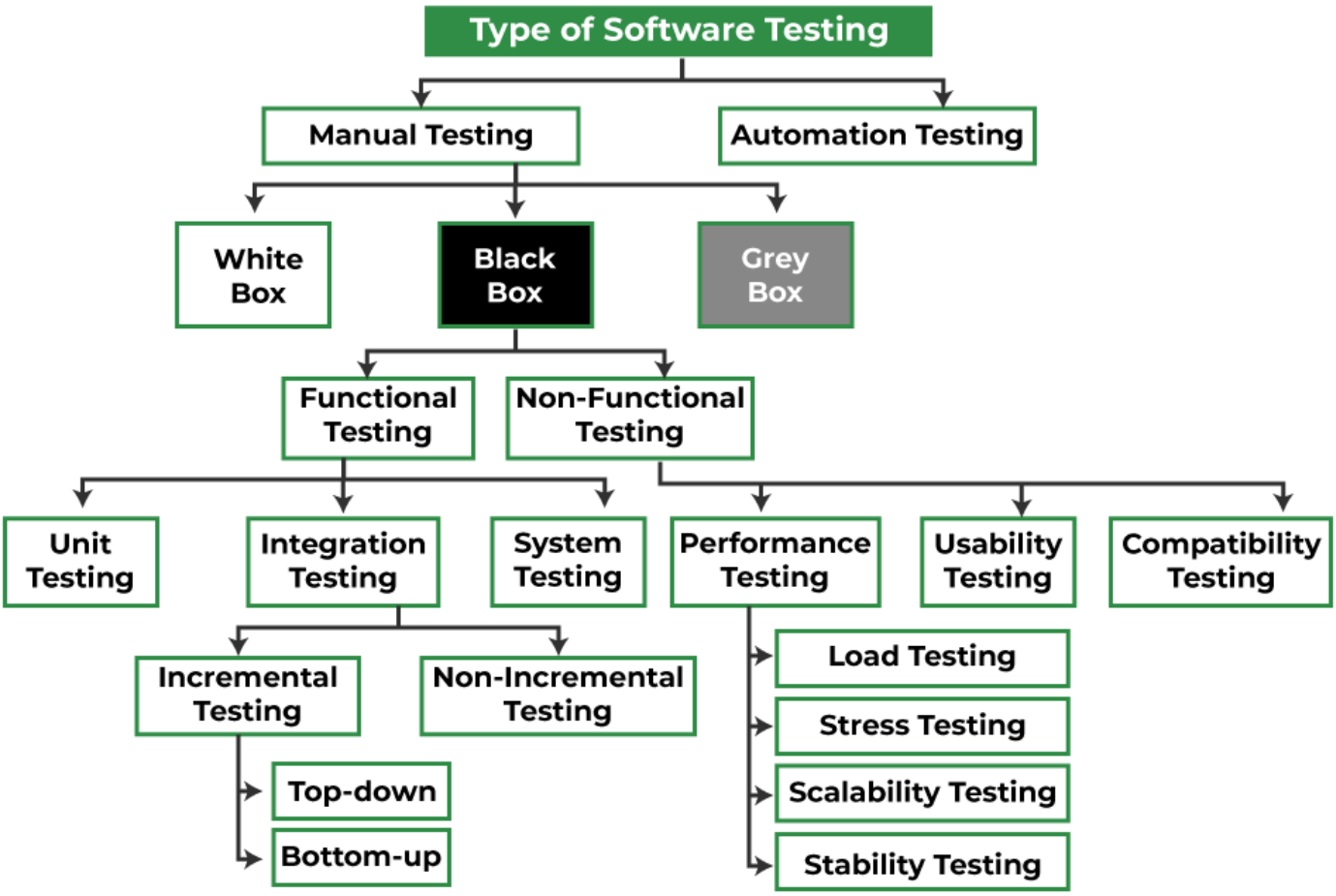
Software testing can be stated as the process of verifying and validating whether a software or application is bug-free, meets the technical requirements as guided by its design and development, and meets the user requirements effectively and efficiently by handling all the exceptional and boundary cases. 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.

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.

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?".



Debugging – Software Engineering

Debugging is the process of identifying and resolving errors, or bugs, in a software system. It is an important aspect of software engineering because bugs can cause a software system to malfunction, and can lead to poor performance or incorrect results. Debugging can be a time-consuming and complex task, but it is essential for ensuring that a software system is functioning correctly.

There are several common methods and techniques used in debugging, including:

1. **Code Inspection:** This involves manually reviewing the source code of a software system to identify potential bugs or errors.
2. **Debugging Tools:** There are various tools available for debugging such as debuggers, trace tools, and profilers that can be used to identify and resolve bugs.
3. **Unit Testing:** This involves testing individual units or components of a software system to identify bugs or errors.
4. **Integration Testing:** This involves testing the interactions between different components of a software system to identify bugs or errors.
5. **System Testing:** This involves testing the entire software system to identify bugs or errors.
6. **Monitoring**: This involves monitoring a software system for unusual behavior or performance issues that can indicate the presence of bugs or errors.
7. **Logging**: This involves recording events and messages related to the software system, which can be used to identify bugs or errors.

Advantages of Debugging:

Several advantages of debugging in software engineering:

1. **Improved system quality**: By identifying and resolving bugs, a software system can be made more reliable and efficient, resulting in improved overall quality.

2. **Reduced system downtime:** By identifying and resolving bugs, a software system can be made more stable and less likely to experience downtime, which can result in improved availability for users.

3. **Increased user satisfaction**: By identifying and resolving bugs, a software system can be made more user-friendly and better able to meet the needs of users, which can result in increased satisfaction.

4. **Reduced development costs**: Identifying and resolving bugs early in the development process, can save time and resources that would otherwise be spent on fixing bugs later in the development process or after the system has been deployed.

5. **Increased security**: By identifying and resolving bugs that could be exploited by attackers, a software system can be made more secure, reducing the risk of security breaches.

6. **Facilitates change**: With debugging, it becomes easy to make changes to the software as it becomes easy to identify and fix bugs that would have been caused by the changes.

7. **Better understanding of the system**: Debugging can help developers gain a better understanding of how a software system works, and how different components of the system interact with one another.

8. **Facilitates testing**: By identifying and resolving bugs, it makes it easier to test the software and ensure that it meets the requirements and specifications.

In summary, debugging is an important aspect of software engineering as it helps to improve system quality, reduce system downtime, increase user satisfaction, reduce development costs, increase security, facilitate change, a better understanding of the system, and facilitate testing.

Disadvantages of Debugging:

While debugging is an important aspect of software engineering, there are also some disadvantages to consider:

1. **Time-consuming**: Debugging can be a time-consuming process, especially if the bug is difficult to find or reproduce. This can cause delays in the development process and add to the overall cost of the project.

2. **Requires specialized skills**: Debugging can be a complex task that requires specialized skills and knowledge. This can be a challenge for developers who are not familiar with the tools and techniques used in debugging.

3. **Can be difficult to reproduce:** Some bugs may be difficult to reproduce, which can make it challenging to identify and resolve them.

4. **Can be difficult to diagnose**: Some bugs may be caused by interactions between different components of a software system, which can make it challenging to identify the root cause of the problem.

5. **Can be difficult to fix**: Some bugs may be caused by fundamental design flaws or architecture issues, which can be difficult or impossible to fix without significant changes to the software system.

6. **Limited insight**: In some cases, debugging tools can only provide limited insight into the problem and may not provide enough information to identify the root cause of the problem.

7. **Can be expensive:** Debugging can be an expensive process, especially if it requires additional resources such as specialized debugging tools or additional development time.

White box Testing – Software Engineering

White box testing techniques analyze the internal structures the used data structures, internal design, code structure, and the working of the software rather than just the functionality as in black box testing. It is also called glass box testing or clear box testing or structural testing. White Box Testing is also known as transparent testing or open box testing.

White box testing is a software testing technique that involves testing the internal structure and workings of a software application. The tester has access to the source code and uses this knowledge to design test cases that can verify the correctness of the software at the code level.

Process of White Box Testing

1. **Input:** Requirements, Functional specifications, design documents, source code.
2. **Processing:** Performing risk analysis to guide through the entire process.
3. **Proper test planning:** Designing test cases to cover the entire code. Execute rinse-repeat until error-free software is reached. Also, the results are communicated.
4. **Output:** Preparing final report of the entire testing process.

Testing Techniques

1. Statement Coverage

In this technique, the aim is to traverse all statements at least once. Hence, each line of code is tested. In the case of a flowchart, every node must be traversed at least once. Since all lines of code are covered, it helps in pointing out faulty code.

2. Branch Coverage:

In this technique, test cases are designed so that each branch from all decision points is traversed at least once. In a flowchart, all edges must be traversed at least once.

3. Condition Coverage

In this technique, all individual conditions must be covered as shown in the following example:

- READ X, Y
- IF(X == 0 || Y == 0)
- PRINT ‘0’
- #TC1 – X = 0, Y = 55
- #TC2 – X = 5, Y = 0

4. Multiple Condition Coverage

In this technique, all the possible combinations of the possible outcomes of conditions are tested at least once.

5. Basis Path Testing

In this technique, control flow graphs are made from code or flowchart and then Cyclomatic complexity is calculated which defines the number of independent paths so that the minimal number of test cases can be designed for each independent path.

6. Loop Testing

Loops are widely used and these are fundamental to many algorithms hence, their testing is very important. Errors often occur at the beginnings and ends of loops.

Advantages of Whitebox Testing

1. **Thorough Testing**: White box testing is thorough as the entire code and structures are tested.
2. **Code Optimization:** It results in the optimization of code removing errors and helps in removing extra lines of code.
3. **Early Detection of Defects:** It can start at an earlier stage as it doesn't require any interface as in the case of black box testing.
4. **Integration with SDLC:** White box testing can be easily started in Software Development Life Cycle.
5. **Detection of Complex Defects:** Testers can identify defects that cannot be detected through other testing techniques.
6. **Comprehensive Test Cases:** Testers can create more comprehensive and effective test cases that cover all code paths.
7. Testers can ensure that the code meets coding standards and is optimized for performance.

Disadvantages of White box Testing

1. **Programming Knowledge and Source Code Access:** Testers need to have programming knowledge and access to the source code to perform tests.
2. **Overemphasis on Internal Workings:** Testers may focus too much on the internal workings of the software and may miss external issues.
3. **Bias in Testing:** Testers may have a biased view of the software since they are familiar with its internal workings.
4. **Test Case Overhead:** Redesigning code and rewriting code needs test cases to be written again.
5. **Dependency on Tester Expertise:** Testers are required to have in-depth knowledge of the code and programming language as opposed to black-box testing.
6. **Inability to Detect Missing Functionalities:** Missing functionalities cannot be detected as the code that exists is tested.
7. **Increased Production Errors:** High chances of errors in production.

Black box testing – Software Engineering

Black-box testing is a type of software testing in which the tester is not concerned with the internal knowledge or implementation details of the software but rather focuses on validating the functionality based on the provided specifications or requirements.

Black box testing can be done in the following ways:
1. Syntax-Driven Testing – This type of testing is applied to systems that can be syntactically represented by some language. For example, language can be represented by context-free grammar. In this, the test cases are generated so that each grammar rule is used at least once.
2. Equivalence partitioning – It is often seen that many types of inputs work similarly so instead of giving all of them separately we can group them and test only one input of each group. The idea is to partition the input domain of the system into several equivalence classes such that each member of the class works similarly, i.e., if a test case in one class results in some error, other members of the class would also result in the same error.

Black Box Testing Type

The following are the several categories of black box testing:

1. Functional Testing
2. Regression Testing
3. Nonfunctional Testing (NFT)

Functional Testing: It determines the system's software functional requirements.

Regression Testing: It ensures that the newly added code is compatible with the existing code. In other words, a new software update has no impact on the functionality of the software. This is carried out after a system maintenance operation and upgrades.

Nonfunctional Testing: Nonfunctional testing is also known as NFT. This testing is not functional testing of software. It focuses on the software's performance, usability, and scalability.

Advantages of Black Box Testing:

- The tester does not need to have more functional knowledge or programming skills to implement the Black Box Testing.
- It is efficient for implementing the tests in the larger system.
- Tests are executed from the user's or client's point of view.
- Test cases are easily reproducible.
- It is used in finding the ambiguity and contradictions in the functional specifications.

Disadvantages of Black Box Testing:

- There is a possibility of repeating the same tests while implementing the testing process.
- Without clear functional specifications, test cases are difficult to implement.
- It is difficult to execute the test cases because of complex inputs at different stages of testing.
- Sometimes, the reason for the test failure cannot be detected.
- Some programs in the application are not tested.
- It does not reveal the errors in the control structure.
- Working with a large sample space of inputs can be exhaustive and consumes a lot of time.

Model Based Testing in Software Testing

Model-based testing is nothing but a simple testing technique in which we get different test cases that are described by the model.

Significance of Model-Based Testing

1. **Early Defect Detection:** Model-Based Testing (MBT) makes it possible for testers to find problems during the requirements or design phases by using model validation.
2. **Lower Maintenance Costs**: Since test cases are derived from models, any modifications to the system can immediately update the appropriate test cases, which in turn can reflect any changes made to the models.
3. **Reusable Test Assets**: Models and test cases developed during the software development lifecycle can be utilized again for regression testing.
4. **Encouragement of Agile and DevOps Methods:** Because it facilitates quick feedback loops and continuous testing, it works well with Agile and DevOps approaches.
5. **Enhanced Test Coverage:** It generates test cases from models that describe all potential system behaviors and scenarios, assisting in ensuring thorough test coverage.

Advantages of Model-Based Testing

1. The automation efficiency is so much higher in this type and the higher level also be acquired by the model.
2. Comprehensive testing is also possible in this type and the changes that have been made can be easily tested by model.
3. Different types of machines like finite state machines, unified model diagrams, and state charts are mostly taking part in this testing technique.
4. By reducing the cost of the process available in this type. Simultaneously many numbers of processes are running together for performance increase.
5. The defects that are made in the beginning stage are identified and the defect counts increase accordingly the testing undergoes in a progressing manner.

Disadvantages of Model-Based Testing

1. For testing purposes system always needs formal specifications and the changes are made according to different sets in a combined manner.
2. To understand the concept is so much difficult for the user and also for utilization. So, the learning curve of the model will be more i.e. the biggest failure of the model.
3. To overcome this situation, the model should be thoroughly improvised and trained

Object Oriented Testing in Software Testing

object-oriented testing in software testing is an approach that focuses on testing individual classes in an object-oriented program. It involves testing at different levels, including algorithmic, class, cluster, and system-level testing.

Object oriented design (OOD) is a way of designing software systems based on the principles of abstraction, encapsulation, inheritance, and polymorphism. These principles help you create reusable, maintainable, and extensible code that can handle complex problems and changing requirements.

Object-Oriented systems, there are the following additional dependencies:

- Class-to-class dependencies
- Class to method dependencies
- Class to message dependencies
- Class to variable dependencies
- Method to variable dependencies
- Method to message dependencies
- Method to method dependencies

USER INTERFACE TESTING

UI testing or user interface testing is a type of software testing that focuses on checking the appearance, functionality, and usability of different kinds of user interfaces, such as: Graphical user interface (GUI) Command line interface (CLI) Voice user interface (VUI)

CONFIGURATION TESTING IN SOFTWARE TESTING

Configuration Testing is the process of testing the system under each configuration of the supported software and hardware. Here, the different configurations of hardware and software mean the multiple operating system versions, various browsers, various supported drivers, distinct memory sizes, different hard drive types, various types of CPU, etc.
The various configurations are Win XP, Win 7 32/64 bit, Win 8 32/64 bit, Win 10, etc.

1. **Database Configuration:** Oracle, DB2, MySQL, MSSQL Server, Sybase etc.
2. **Browser Configuration:** IE 8, IE 9, FF 16.0, Chrome, Microsoft Edge etc.

1. **Software Configuration Testing:** Software configuration testing is done over the Application Under Test with various operating system versions and various browser versions etc. It is a time-consuming testing as it takes long time to install and uninstall the various software which are to be used for testing. When the build is released, software configuration begins after passing through the unit test and integration test.
2. **Hardware Configuration Testing:** Hardware configuration testing is typically performed in labs where physical machines are used with various hardware connected to them. When

a build is released, the software is installed in all the physical machines to which the hardware is attached and the test is carried out on each and every machine to confirm that the application is working fine.

Security Testing

Security testing is an important aspect of software testing focused on identifying and addressing security vulnerabilities in a software application. It aims to ensure that the software is secure from malicious attacks, unauthorized access, and data breaches.

The goal of Security Testing:

The goal of security testing is to:

- To identify the threats in the system.
- To measure the potential vulnerabilities of the system.
- To help in detecting every possible security risk in the system.
- To help developers fix security problems through coding.
- The goal of security testing is to identify vulnerabilities and potential threats in a system or application and to ensure that the system is protected against unauthorized access, data breaches, and other security-related issues.

Performance Testing

Performance testing is a non-functional software testing technique that determines how the stability, speed, scalability, and responsiveness of an application holds up under a given workload.

Performance Testing Attributes:

- Speed: It determines whether the software product responds rapidly.
- Scalability: It determines the amount of load the software product can handle at a time.
- Stability: It determines whether the software product is stable in case of varying workloads.
- Reliability: It determines whether the software product is secure or not.