International Journal Of Engineering Research And Development
e- ISSN: 2278-067X, p-ISSN: 2278-800X, www.ijerd.com
Volume 21, Issue 4 (April 2025), PP 146-150

Software Testing a Challenging Task

Dr. Vijay Deep Gaur
Assistant Professor of Computer Science
Govt. College, Narnaul (Distt-Mahendergarh) Haryana

Abstract
In this paper my aims to study diverse as well as improved software testing techniques for better quality assurance purposes. Software testing process is the intent of finding software bugs (errors or other defects). Software applications demand has pushed the quality assurance of developed software towards new heights. It has been considered as the most critical stage of the software development life cycle. Testing can analyze the software item to identify the disparity between actual and prescribed conditions and to assess the characteristics of the software. Software testing leads to minimizing errors/debugs and cut down software costs.
Keywords: *Verification, validation, debugging, methodologies, software testing life cycle.*

Date of Submission: 05-04-2025 Date of acceptance: 15-04-2025

I. INTRODUCTION

Software testing the process of evaluating a software application or system to ensure it functions as expected, identifies defects, and meets user requirements and quality standards. Software development involves developing software against a set of requirements. Software testing is needed to verify and validate that the software that has been built to meet these specifications. Software testing helps in the prevention of errors in a system. It refers to the process of evaluating the software to find out the error in it. It is also used to analyze the software for other aspects of the software like usability, compatibility, reliability, integrity, efficiency, security, capability, portability, maintainability, etc. Software testing aims at achieving specific goals and principles which are to be followed. In simple words, testing is the process of locating errors in the program. Software Testing is executing the software to perform following three tasks-

- **Verification:** It is the process of checking the software concerning the specification. Verification: Are we making the product, right?
- **Error Detection:** It is the process of deliberately performing the wrong inputs to check the system's performance.
- **Validation:** It is the process of checking software concerning the customer's expectation. Validation: Are we making the right product

Testing is characterized as a process of assessment that either the definitive system meets it's specified fulfillments initially or not. It is mainly a system beset with the validation and verification process that whether the developed system meets the fulfillments defined by the customers. Therefore, this activity draws the difference between the actual and expected result. So, this is an analysis that provides the associates with the proper knowledge about the quality of the product. Software Testing can also be defined as a risk-based activity. The testing cost and errors can be found in a relationship in figure 1. It is apparently demonstrated that cost rises dramatically in a testing (functional and nonfunctional). The compelling testing goal is to do the optimal amount of tests so that additional testing endeavor can be minimized. From Figure 1, we can say that Software testing is an essential factor in software quality assertion.

II. SOFTWARE TESTING METHODOLOGIES

The importance of software testing to software quality cannot be overemphasized. After the development of the code, it is mandatory to test the software to identify all the errors, and they must be debugged before the release of the software. Although it is impossible to identify and debug all the errors in the significant software at every phase, it is tried to remove all the mistakes as possible. Testing helps in finding the bugs; it cannot conclude that the software is bug-free. We broadly categorized testing techniques into two parts:

1. Static Testing
2. Dynamic Testing

2.1. Manual Testing (Static Testing): It refers to the method of testing where the code is not executed. It does not require highly skilled professionals since the actual execution of the system is not done in this process. It starts with the initial phase of the Software Development Life Cycle (SDLC); hence, it is also known as verification testing. The main objective of static testing is to enhance the quality of software products by helping software professionals

to Identify and resolve their errors early in the software development process. Static testing is performed on the documents like Software Requirement Specification (SRS), design documents, source code, test suites, and web page content. It is performed before code deployment. As a result, it provides the evaluation of code as well as documentation. Static testing techniques include:

2.1.1. Inspection: It is primarily done to locate defects. Moderators conduct the code walkthrough. In this type of formal review, a checklist is prepared to check the working document.

2.1.2. Walkthrough: It is not a formal process. The authors lead this process. The Author advises the participants through the document according to his or her thought process so that they can accomplish a common perception. It is especially useful for higher-level documents like requirement specification, etc.

2.1.3. Technical Reviews: A professional round of review is operated to check if the code is made in consonance with the technical specifications and norms which may include the test plans, test strategy, and test scripts. Informal Reviews: It's the static testing technique in which the document is reviewed unofficially, and helpful comments are implemented.

2.2. Automated Testing (Dynamic Testing): Dynamic Testing is a kind of software testing technique in which the dynamic behavior of the code is analyzed. In dynamic testing, also known as validation where the actual system is considered. It requires the highly skilled professional with the proper domain knowledge. Dynamic testing involves testing the software for the input values, and output values are analyzed. Progressive testing is divided into three categories:

1. White Box Testing
2. Black Box Testing
3. Grey Box Testing

2.2.1. White Box testing

Internal specifications and structures of the system are created conspicuous. So, it's acutely cost-effective in detecting and resolving problems. Bugs will be found before they cause bickering. Thus, we will summarize this approach as testing software with the data of its internal structure and coding. White box testing is also familiar as precise box analysis or white box analysis or glass box testing or transparent box testing, and structural testing. It's an approach for finding errors within which the tester has complete data. This technique isn't used much for debugging in large systems and networks. Different types of white box testing include basis path, loop testing, control structure testing, etc. White-box testing tests internal constructions or workings of a program because programming skills and the domestic context of the system are used to design test cases. The tester appoints inputs to apply paths through the code and finalize the appropriate outputs. This is akin to testing nodes in a circuit. White-box Testing can be used at the unit, integration and system levels of the software testing process. It usually is done at the unit level. Although this approach of testing can expose many errors, it may not identify the missing requirements and unimplemented parts of the specification.

White-box Testing includes the following approaches:

Application Programming Interface testing tests the application using public and private APIs by creating tests to satisfy some criteria for code coverage.

Fault Injection Methods – Introducing faults to gauge the efficacy of testing strategies intentionally.

Code coverage tools can assess the integrity of a test suite that was created with any method, including black-box testing. This grants the software team to check the parts of a system that are rarely tested and assures that the most important function points have been verified.

Function coverage is the approach which informs on functions that have been executed.

Statement coverage is the approach which informs the number of lines executed to complete the test 100 per cent. It assures that all code paths or branches are executed at least once. This helps to ensure correct functionality.

Advantages:

1. It exposes an error that is hidden in code by eliminating extra lines of code.
2. Maximum coverage is obtained during test outline writing.
3. The developer discreetly gives reasons for implementation.

Disadvantages:
1. An experienced tester is required to carry out this procedure because knowledge of internal structure is needed.
2. Many paths will remain untested since it is challenging to look into every pros and con.

2.2.2. Black Box testing

A black box testing is a testing in which internal specifics and workings aren't known or accessible to its customer. It supports specifications and output needs. The fundamental purpose is to identify the requirements of the system. Black box testing has very little or no data on the inner logical structure of the system. Thus, it solely checks the basic features of the system. It assures that every input is appropriately accepted and outputs are correctly produced. Black-box testing checks the functionality without any knowledge of the internal implementation. The testers only have an understanding of what the software is supposed to do, not how it does it. This is solely done based on customers' aspect. The tester only knows the set of inputs and specific outputs.

Black-box testing methods include:
- **Equivalence Partitioning:** This technique divides the input domain of a program into equivalent classes from which test cases can be derived. Thus, it can minimize the number of test cases.
- **Boundary Value Analysis:** It targets the testing at borders, or where the extreme boundary values are chosen. It comprises of minimum, maximum, error values, and typical values.
- **Fuzzing:** This approach takes random input to the application. It is used for characterizing implementation bugs, using malformed or semi- malformed data injection in an automated or semi-automated session.
- **Orthogonal Array Testing**: In this technique, the input domain is minimal but too large to accommodate exhaustive testing.
- **Cause-Effect Graph:** This testing technique begins by generating a graph and establishing the relation between effect and its causes.
- **All Pair Testing:** The main objective is to have a set of test cases that covers all the pairs. Here, test cases are designed to execute all possible discrete combinations of each couple of input parameters.
- **State Transition Testing:** This testing approach is useful for the navigation of a graphical user interface.

Advantages:
1. Testers do not need to know specific programming languages. Testing is done from a user's point of view.
2. It helps to find out any ambiguities or inconsistencies in the requirement specifications.
3. Programmer and validators both are independent of each other.

Disadvantages:
1. Test cases are difficult to design without fair stipulations.
2. Probability of having the repetition of tests that are already done by the programmer.
3. Here, some parts of the back end are not tested at all.

2.2.3. Grey-Box Testing

Gray box testing is the technique of testing the software with limited knowledge of the internal structure and design of the application. It is defined as a testing software package which has some data of its internal logic and underlying code. It uses internal information structures and algorithms. This approach holds necessary conducting integration testing between two or more modules of code written by totally different developers. This approach includes reverse engineering to work out on the boundary values. Grey box testing is unbiased and non-intrusive.

Grey-box Testing has the knowledge of internal data structures and algorithms for purposes of designing tests while executing those tests at the user level. The tester does not have full access to the software's source code. The following are some subtypes of grey-box testing:
State-Model-Testing: It examines each method of an object, transition & transition paths at each state of an object.

III. SOFTWARE TESTING STRATEGIES

Software Testing strategies provide a method of integrating software test case design methods into a well-planned Series of steps that can result in the successful construction of software. It provides the road map for testing. The software testing Strategy should be pliable enough to develop a customized testing approach.

The software testing strategy is actually produced by project managers, software engineers, and testing specialists. There are four different types of software testing strategies:

1. Unit testing
2. Integration testing
3. Acceptance/Validation testing

3.1. Unit testing

Unit is the smallest testable part, i.e. the most modest collection of lines of code which can be tested. Unit testing is done by the developer as the proper knowledge about the core programming designing is required. Generally, unit testing is considered as a white-box testing class because it is partisan to evaluate the code as implemented rather than assessing conformance to some set of requirements.

Benefits of Unit Testing:

1. Cost-effective testing technique.
2. Simple testing technique because the smallest testable unit of the code is tested here.
3. Individual parts are tested when necessary, without waiting for another part of the system.
4. Unit testing can be performed in parallel by fixing problems simultaneously by many engineers.
5. Detection and removal of defects are much cost- effective compared to other levels of testing.
6. Be able to take advantage of several formal testing approaches available for unit testing.
7. Clarify debugging by limiting to a small unit the possible code areas in which to search for bugs.
8. Be able to test internal logic that is not easily reached by external inputs in the broader integrated systems.
9. Attain a high level of structural coverage of the code.
10. When debugging severe problems, it avoids lengthy compile-build-debug cycles.

Unit testing techniques: Unit testing uses several effective testing techniques. The testing techniques categorize into three types:

1. Functional Testing
2. Structural Testing
3. Heuristic or Intuitive Testing

3.2. Integration testing

Integration testing is an efficient technique for constructing the program structure as well as to perform tests to uncover errors related to interfacing. The objective of integration testing is to integrate the unit tested component and tested them as a group. Different Integration testing Strategies are:

- Top-down Integration testing
- Bottom-up Integration testing

Top-down Integration: It is an incremental procedure to build a program format. Formats are integrated by moving downward through the fabric, beginning with the main control module. Modules subordinate to the central control module are incorporated into the structure in either a depth-first or breadth-first manner. The integration activity is performed in a series of five steps:

- The main control format is used as a test driver, and stubs are substituted for all components directly subordinate to the central control module.
- Depending on the integration techniques, selected subordinate stubs are replaced one at a time with actual parts.
- Tests are conducted as each element is integrated.
- On completion of each set of experiments, another stub is replaced with the real component.

Bottom-up Integration: It begins development and testing with atomic modules. Since modules are integrated from the bottom-up, processing required for elements subordinate to a required level is always available. The following steps are implemented with bottom-up integration strategy:

- Low-level components are combined into clusters so that they can perform a specific software sub-function.
- A driver is reported to coordinate test case input and output.
- The group is tested.
- Drivers are detached, and clusters are combined moving upward in the program structure.

3.3. Acceptance testing

In this approach, testing is carried out to authenticate whether the product is developed as per the standards and detailed criteria and meets all the requirements specified by the user. The user carries this

IV. CONCLUSION

To conclude our survey, we would like to find that Software testing is an essential activity of the Software Development Life Cycle (SDLC). We can never say that a product is "Perfect". Testing is a never-ending process. Testing only shows the presence of errors, not the absence. It is time-consuming and an intensive process, therefore, upgraded techniques and innovative methodologies are necessary to maintain the quality of the software. This leads to performing Automated Testing implementation before and during the testing process. This paper aims to describe in detail various software testing techniques, the need for software testing, software testing goals, and objectives. Software testing is often less formal and meticulous than it should. In order to perform software testing effectively and efficiently, those who are involved in testing must be familiar with the basic concept, goals and principle of software testing.

We further discuss different software testing techniques such as white-box Testing, black-box Testing, grey-box Testing, Security testing etc. Hence, the future work in relevance with the testing process will be much more technology-dependent harnessing the simulation and automated testing model-based approach, not only expediting the testing life cycle but also providing optimum bug prevention and efficient quality assurance.

REFERENCES

[1]. Nahid anwar & susmita kar, softwaretesting techniques & strategies, global journal of computer science and technology, Software & data engineering, 2019
[2]. Kaner, Cem (November 17, 2006). "Exploratory Testing" (PDF). Florida Institute of Technology, Quality Assurance Institute Worldwide Annual Software Testing Conference, Orlando, FL. Retrieved November 22, 2014.
[3]. Glenford J. Myers, "The Art of Software Testing, Second Edition" Published by John Wiley & Sons, Inc., Hoboken, New Jersey. 1.1. P. Mathur, "Foundation of Software Testing", Pearson/Addison Wesley, 2008.
[4]. P. Mitra, S. Chatterjee, and N. Ali, "Graphical analysis of MC/DC using automated software testing, in Electronics Computer Technology (ICECT), 2011 3rd International Conference on, 2011, vol. 3, pp. 145 –149.
[5]. F. Saglietti, N. Oster, and F. Pinte, "White and grey- box verification and validation approaches for safety- and security-critical software systems," Information Security Technical Report, vol. 13, no. 1,
[6]. Mohd. Ehmer Khan, "Different Forms of Software Testing Techniques for Finding Errors", IJCSI International Journal of Computer Science Issues, Vol. 7, Issue 3, No 1, May 2010.