A New and Effective Method for Researching Various Software Testing Techniques and Strategies

Santosh Kumar Vududala

Sanqa19@gmail.com
Independent Researcher

Abstract

Software testing is an area that is critical to the reliability, functionality, and performance of software systems. As new technologies are developed for the software industry, we should focus on the quality of software to develop a quality and developed software. Software quality assurance in IT sector has been treated as backbone of software development. Software applications capabilities have taken quality assurance of software to new levels. Testing is also able to reduce the cost of software and minimize the chances of mistakes in the code. We all know that testing is one of the key components for software development life cycle. Today, the fast pace of the software industry is redefining how we view a plethora of software testing techniques and strategies, specifically sturdy or effective methodologies that can be adopted with minimal friction. It combines the familiar aspects, Manual and streamlines testing to the newly evolving techniques like AI-focused testing and DevOps-centric approaches. By carrying out a thorough review of the available literature and case studies, this research highlights the main factors that impact testing results and suggests a guideline for choosing appropriate techniques tailored to individual project needs. This paper offers new insights for practitioners and researchers in the expectation of promoting the evolution of modern software testing methodologies to enhance software quality assurance.

Keywords: Manual Testing, Automation Testing, Quality Assurance, Testing Techniques, Verification, Test Optimization

Introduction

The goal of software testing is to verify that the system meets all functional, performance, and security requirements as specified in the requirements documents. In a world where software is becoming ever more complex and development timelines ever shorter, so must the testing techniques used to support their development be more efficient and effective than ever. Regular techniques like manual and computerization, and exploratory testing, stay essential. Nevertheless, the landscape is changing due to more powerful trends such as automated testing, AI-based approaches and continuous testing in DevOps ecosystems. The following are less specific topics: an essential of the Software Testing. It is a very complicated activity that warrants a first-class citizen in the software development. Software Testing is nothing but finding the bugs. So this is something that one should keep in mind when ever to design and implement the computer based system or product that the testability. Testing is the type of process of measuring the system or its components in order to discover the differences between the present state and the desired state.

Figure.1. Software testing

A wide range of research endeavors, empirical investigations, and real-world applications are presented in the published work on this topic with the goal of utilizing historical data to increase the precision and dependability of faulty forecasting. Figure 1 lists the various forms of evaluation of software that have been explained.

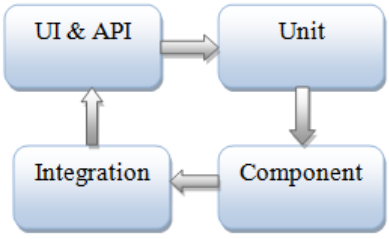

Figure.2. Types of software testing

It is impossible to eliminate the defects and usually identified as bugs. Testing must make sure it polishes all groups of user requirements and specifications [1]. It does highlight other features of software such as portability, security, maintainability etc. [2]. Some of the common practices that contribute to writing quality software areSoftware testing helps in preventing system failures. This process is used to evaluate a software to identify where an error occurred. It aided in formulating particular goals and objectives: Software testing aims to fulfill client needs and requirements. Following the software industry we see that software testers can perform testing thoroughly by generating test cases, test stubs and test plans to reduce the test sets. Testing sometimes costlier as risk of cost overrun and schedule slips are associated with it. Software testing aims to provide a good product concerning reliability estimation. The purpose of secondary testing is to execute the program to detect errors, and to develop a test case that can identify the error that has not been identified. This investigation gives stakeholders accurate data regarding product quality. Software testing can also be seen as a risk-based process. The significance of the software during this testing phase cannot be emphasized enough. Testers have to learn how to filter a large number of errors down into a couple of manageable sets of tests and make the right call on which risks need testing. If that proposed testing process could be automated, the cost of developing software could be a fraction of what it is now. Particularly in software testing, one of the issues include the issue of generating test data. Test data generation in software testing is the act of identifying program input data that maximizes coverage of a specific testing criterion. A test data generator is a tool that assists a programmer in producing test data for a program [3] Software testing aims to maximize software quality. In this paper, we talk about testing methods and approaches the most general ones. What they are used for and how to use them The following section explains how they work and what makes them different from each other. Model Based Testing:

black, white and grey Techniques: Acceptance Testing, System Testing and Unit Testing software testing is a validation process to define quality of the software and also a technical process to attain that quality. an integral aspect of software engineering. It's been around for almost as long as software engineering. The quality of a software system depends upon the method by which it is tested. Companies and testers suggest that they should spend 40-50% of their time and money on testing. Because the system is tested correctly, so as to achieve a high level of reliability, maintainability, availability, security, survivability, portability, capability, efficiency, and integrity. [4] The system will work as expected due to software testing. Modern software must be accurate and provide all required functionality for critical applications. Testing is greatly beneficial to the organization, end user, developer and tester. Software testing is a vast domain with lots of testing types and techniques. which our paper discusses. [5] Software testing is also a risk-based activity that you can think of when doing it. During the testing process, it is important for software testers to know how to slice a huge set of tests into smaller sets and make informed decisions about which risks should be tested and which risks should not be tested Discovery testing is a demanding technique that would essentially examine the application utilization without delving into the degree of usage. You can continue to apply this process to every degree of testing within the SDLC. It mostly performs the testing with the goal that it covers every last trace of the usefulness of the application to check whether it meets the previously account prerequisites of the client or something else. It is suitable for identifying wrong functions. Examples by evaluating the effectiveness of each lower, upper and lower-case value. Widespread testing means testing even without symptoms; however, this is the most simple and around about testing measure used everywhere [6] [7].

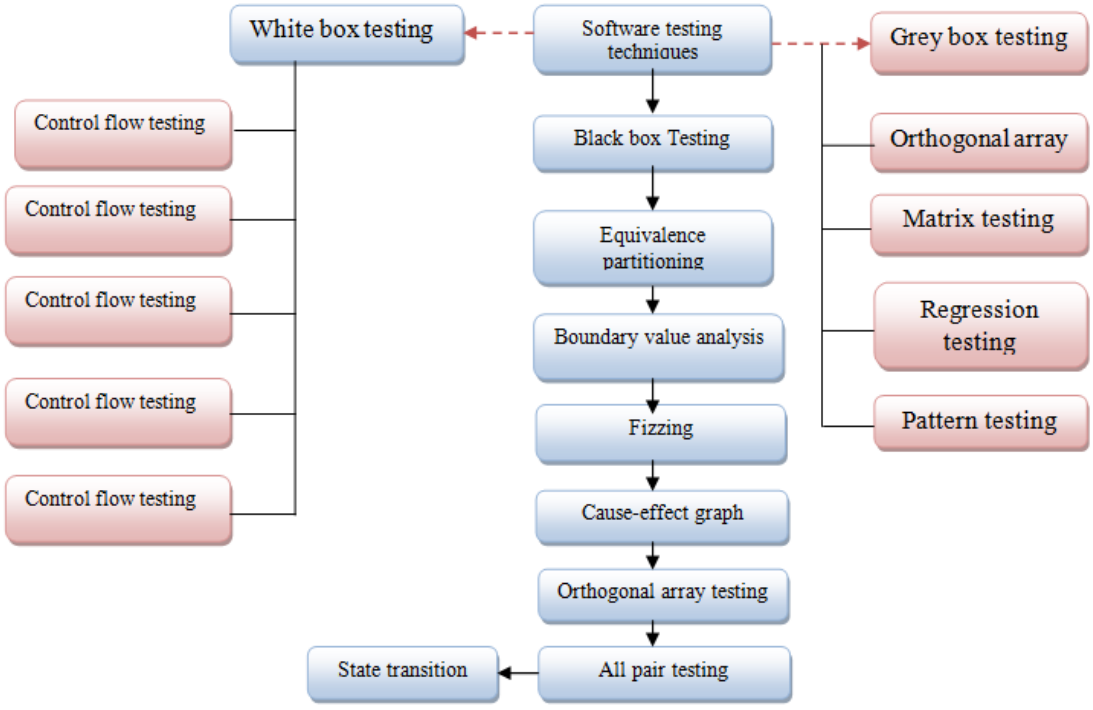

Figure.3. Software testing techniques [8]

White box and black box testing approaches are combined to create grey box testing, which combines their benefits. The requirement for this kind of testing was sparked by the fact that the tester is familiar with the underlying workings of the program and can therefore test its performance more effectively while taking everything into account. Figure 3

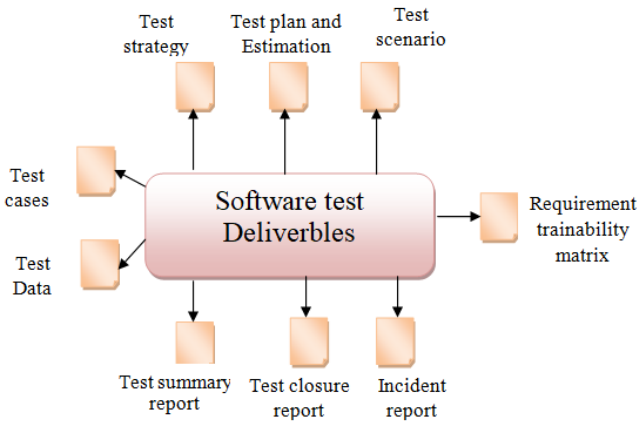

Figure.4. Software testing deliverables

Figures 4 exhibits a relationship between testing cost and errorsfunctional and nonfunctional types testing significantly increases costs based on "Figure 4" and this fact should not be missed. Deciding what to test or how much to test can often lead to many bugs being missed. The goal of efficient testing is to perform as many tests as possible and reduce the cost of additional testing [9]. Software testing is a significant component of software quality assurance, as illustrated in Figure 1. This is particularly relevant for life-critical software such as safety critical flight control software, which can show the importance of testing by demonstrating the risk of schedule delays, cost overruns or even cancellation [10], and further regarding this[11-12]. There are several stages and steps to the testing, and the examinee has different from level to level. Unit testing, integration testing, and system testing are the three sequential phases of software testing. Software developers or quality assurance engineers, sometimes known as software testers, test the steps [13]. The Software Development Lifecycle (SDLC) includes all of the testing procedures listed above. Software development must be broken down into modules, which are managed by separate teams or individuals.

After a developer completes his development, checking each module or unit behavior to see if it satisfies business expectations is called Unit Testing. Integration Testing is the 2[nd] stage of the SDLC testing process. It is common to have errors during the build, when the modules of a single software system are developed independently and integrated after development is one of the goals of software engineering. System Testing or testing the software as a whole from all perspectives is the last testing stage in the software development lifecycle (SDLC). Efficacious software testing mechanism for costs optimization [14]. The testing cycle primarily comprises of multiple phases ranging from Test Planning to analyzing Test Results. The approach to each test activity that must be carried out throughout the entire test process is the main emphasis of phase one of test planning. Test Development: The test cases that will be used in the testing process are created during this second stage of the testing life cycle. Test Execution, which includes carrying out the test cases, is the following stage in the test cycle. The following phase, Test Reporting, is where the corresponding defects are reported. The last stage of the testing process is Test Result Analysis. In this phase the defect is analyzed by the software or system developer. This step may also be done with the client as this will allow them to gain an understanding of what they have to ignore as opposed to what they have to fix, improve or just change [15].

Literature Review

The first step in testing procedure is creation of test cases. A plethora of testing methods is utilized to compose the test cases in order to ensure efficient and accurate testing. Black box, white box, and grey box testing are the most often used testing techniques [8]. White box testing evaluates the internal organization of the application in addition to its functionality. It very effective. But, the necessity of programming

knowledge during the testing process renders white box assessment a complex procedure [16-17]. Black Box testing is a type of testing that tests the functionality of the applications without involving the details of the implementation. This technique is applicable in any phase of SDLC testing. It primarily does the testing in a manner it tests all the features of the application to find out whether it meets the statutory requirement of the user initially. It identifies incorrect functions by testing their functionality at each minimum, maximum and base case value. It is the simplest and most routine method of testing in the world. The test case is the creation of the test designing phase which ends the test planning process And the QA team manually write relevant test cases or sometimes even write automated test cases. A test case defines a set of test inputs or data, the conditions of execution, and the expected results. It's crucial to choose the required group of test data so that it gives you both the desired outcome and data that is purposely not correct and regulates an error in the course of the test. This is usually done to understand the failure scenario [18] This life cycle comes under the umbrella of STLC and adds on more tests like Alpha and Beta tests. Alpha Testing: Alpha refers to the stage of the software application when it is in the first stage of developer-level testing; it can be done through either white box testing or grey box testing. System or integration testing could employ a black box approach (alpha release). Alpha testing ends when a feature freeze is in place and no more features are introduced to enhance functionality, or for any other reason [19-20]. Since the beta testing step is conducted by the user following the Alpha release, it can be considered official acceptance testing.

It comes just after the Alpha testing period. For testing, a portion of the target users are given access to the software or application. Before a program is formally released, a beta version is typically made available for the target audience to provide input. The target mailing list is beta testers. In contrast, the application will mostly be a software prototype for demonstration reasons. Following beta testing, the software's final version is released.

[21-22] Alpha Testing: The term "alpha" refers to the initial testing of an application by the developer using either the white box or gray box techniques. Alpha releases, also referred to as black box testing, may be conducted at the integration or system level. Usually, a feature freeze marks the conclusion of alpha testing, during which no new features will be added to expand functionality or for any other reason.[23–24] After Alpha testing, Beta Testing phase comes which can be seen as pretty much a formal acceptance testing since it is performed by the user, after the Alpha release. Testing: The software or the application is launched to a limited set of users for testing purpose. And typically, the beta stage apps will be distributed to the intended audience for feedback before getting the official release. This audience is usually referred to as Beta Testers and the app is rarely referred to as anything other than a prototype version of the software and even more rarely for demonstration purposes. So the final build of the software is released after Beta Testing [25-26]. Software Testing Life Cycle (STLC) Figure 5: STLC (Software Testing Life Cycle) steps stages and phases a software undergo during the testing process However, there cannot be a dedicated standard of the software or application that undergo STLC, it may differ accordingly in different places in world [27].

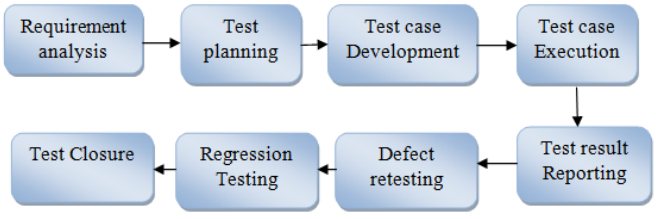

Figure.5.Software Testing Life Cycle [28]

The product prerequisites survey is done by the Quality Assurance group during a significant part of the STLC where they get the center requirements based on which the test execution will happen. To better understand and address any conflicts that may arise, the team should coordinate with the development team. As the phase that characterizes the entire testing system, test arrangement is the second and most important stage in the STLC. The test plan, which is the stage's last output, is organized under the direction of this phase. The testing process would be impossible without the Test Plan, a required document that is geared toward the application's functional testing [27].

Testing Methodologies

The implementation of software testing may have a significant impact on the assurance of software quality. It is required to test the program once it has been coded in order to identify and fix any bugs prior to it is released. While it is impossible to find and fix every fault in crucial software at every step, we make every effort to get rid of as many errors as we can. Typically, we separate the testing methods into two sections.:

a. **Manual Testing**
b. **Automation testing**

Manual Testing (Static)

The primary form of testing is manual testing. In manual testing, the tester executes the test case manually the representation is displayed as in the figure 6. You have to do manual testing for each product before you run automated tests. You catch product errors through manual testing. If you are an early stage project, you may choose manual testing as manual testing Even if you are working with a short term project, especially if you visually want to test the user interface. Manual testing is recommended for short-lived projects that are costly to script. Step 1 Understand requirements document to successfully do manual testing. Hence, by knowing the needs you know what you need to check and you need to categorize as a bug or bug You are being tested on an essential detail of the test. Step2: Create a test case. After knowing the requirements, you will be able to create the test cases/conduct testing. It basically helps in guiding the sequence of tests, test functions and various scenarios mentioned in the software products or applications. Hence the importance of creating good test cases, which helps in executing all test cases smoothly and improving test coverage. Step3: Now after creating the test case then you just run it. If the test cases are created and the test environment is set up correctly, start тестING. You are also tracking test cases and can mark every test pass, fail or skip.` When testing manually, you see a lot of things that you need to write down. This is very important. When the test fails the facts do not go away. Step4: Then log the corresponding bug report. As a tester, I do more than just testing that includes error logging. So if you have a bug, you will send the development team the log information about your bug. This goes a long way in helping you and your team at a later point — good bug report. This way you can answer these error questions and save time later. Of course, there is basically a document to point to. A well-written bug report should contain the title of the report, how to reproduce/fix the bug, expected vs. actual output, and related image or video attachments that would help the development team understand the issue better. Step5: Also you have to form a detailed test report. To understand the test at a high level, it is good to know the test after running it. For example, the number of tests passed, the number of tests failed, and the number of tests passed, the number of test skipped, the list of tests in that manner, etc. This procedure is very simple.

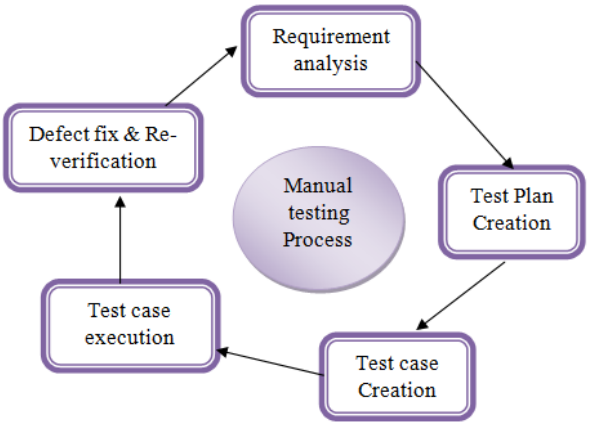

Figure.6. Manual testing

Automation Testing (Dynamic)

Automated testing is testing that is done using automated tools Automated testing aims to alleviate as much as possible of the testing burden with a few scripts shown figure 7. As a case in point, a lot of resources for a quality assurance (QA) team go into regression testing, so this could be a good candidate for automation. The goal of automated testing is to improve accuracy, efficiency, and quality assurance at the end [29]. Automation involves third-party software which generates the required script to manage the execution for the automated test. During manual testing, the tester sits and runs the test suits one by one. By this stage the tester will need to write a test script. Automated tests can be run significantly faster than manual tests. In automated testing, you cannot use random testing. Automation tests are performed on tool bases. It is a field requiring extremely competent testers who understand the field well. Tests that are automated are known as validation tests. To change the tester to modify the script Tests can be executed on multiple hardware systems, which are time-saving. Automation testers need to have good programming language knowledge to execute automation testing. Automated testing is cheap. Manual testing is always defeated with automation testing. There needs to be more investment in tools and automation engineers. Cock test automation has lower complexity test environment setup requirements to validate the script.

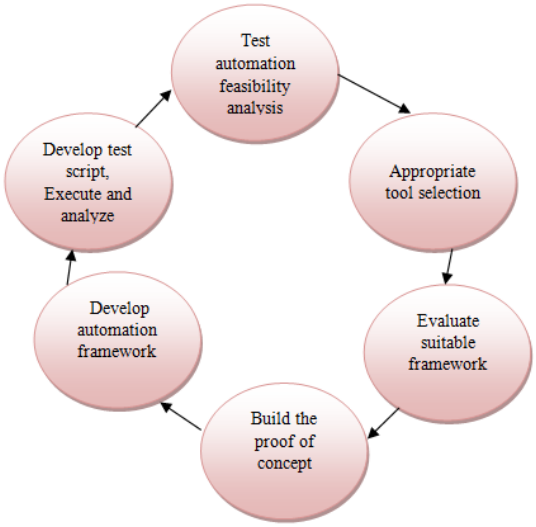

Figure.7. Automation testing

Test Automation

The focus of testing process to be towards Test Automation is the most important upgrade in the overall testing process. More precisely, execute the test process with a piece of software in mind and compare real

results with expected results. Your time is saved by test automation techniques, and you have a lot of boring time saved by the test automation techniques. In SDLC, test automation is performed both in implementation as well as in testing. Like us, Test automation is practiced worldwide instead of Manual testing as they save the time where they work for testing for less time. The emergence of test automation largely replaced the manual testing process by lessening its requirement and exposing the quantity of bugs and defects that the manual testing process fails to identify. One of the most significant tests is regression testing, and running it manually can consume a lot of time. After fixing errors and bugs, we usually test if the software or application is being executed properly. After debugging, the code or application error or error rate can be higher than that as well. That's why, to prevent the time required for regression testing, a number of automated test suites are prepared and a typical regression test suite is considered for that reason. As a result, test automation finds issues significantly earlier, which saves enormous change costs and energy in the future.

Methods for Testing

Black-box testing: a process of software testing in which testing is done without seeing the internal tab and data of the software. Testing without the knowledge of the internal design or the access of the source code is called as Black box tester Testers possess system architecture expertise This way, this method aims to ensure that all inputs to the system are as expected, so that the output, as expected, is correct. Structural Testing and Code-Based Testing are other names for this method. This technique is primarily concerned with the internal logic and structure of the actual source code. It can use for integration, unit, and integration level

Testing Framework in the Agile

Another progress in programming testing is the coordinated lifecycle when refrain over the cerebrum testing shopping district of test cycles that sorry and quick with regularly changing necessities. In this way, the lithe condition may incorporate any testing system; however, because of the continuous cycles and spirited change in expected prerequisites, it upholds test robotization suite turns out to be very tedious. Testing structures making the dreadful fit remains a troublesome test.

Test Driven Development

This approach discusses the decoupling current and guides the code architecture with automated unit tests. The certainty level regarding the framework achieving its center particulars is increased when TDD provides a completely clear proportion of structure achievement at the point when the test no longer fails and at least one flaw or error is typically discovered using the standard testing measure analyzer. It is possible that a considerable portion of the time spent on TDD is lost during the troubleshooting cycle. [30].

Testing Metrics

Prioritization metrics

Because they can significantly impact your testing cycle, test metrics are of utmost importance. They are an important indicator of the sufficiency, accuracy, and research of macro-defined metrics. They can also help identify areas that require improvement and the subsequent steps that must be made to eradicate them. A one phase in the STLC, test metrics also act as a catch-all for the ongoing improvement of the testing process as a whole [31–32]. Software testing metrics, which are divided into two categories Process Quality Metrics and Product Quality Metrics are essentially a collection of standards used to determine the quality of a process and a product. Their goals are to enhance the testing cycle and requirements for item quality.

Cycle Quality metrics

The most obvious component is a cycle, which is designed to yield a high-quality outcome in the least amount of time and in the most economical way. This provides a clear explanation of why organizations all over the world have focused on improving the execution of cycles, which is precisely where the need for measures arose because it is necessary to accurately evaluate the cycle from various measurements. Cycle quality is evaluated using the primary testing quality measurement in the cycle, which generates specific component estimations such as the Test Progress Curve, which deals with the planned progression of the Testing Phase by the test plan [33–34].

Testing is the next big phase of the metric both stage savvy and segment wise after the expense of Testing. Which is a major aim to help identify the components that need targeted testing and the expenditure they will incur as a result. Another indicator that shows the typical time it takes for the testing team to confirm the flaws is the Normal Defect Turnaround Time. One crucial indicator of operating proficiency is the normal defect response time. This represents the proportion of the group's normal time devoted to fixing the errors. Through measurements, Process Effectiveness ensures that the resulting application or items will produce a high yield. The main components that improve the testing process are test automation, defect elimination efficiency, requirement volatility index, and fizzled and execute experiments.

New Approaches

The requirements of contemporary software development, which are typified by shorter release cycles, increased complexity, and the requirement for high reliability, have given rise to new methods in software testing (see figure 8). These techniques combine cutting-edge technology with cooperative procedures to improve productivity, scalability, and reliability.

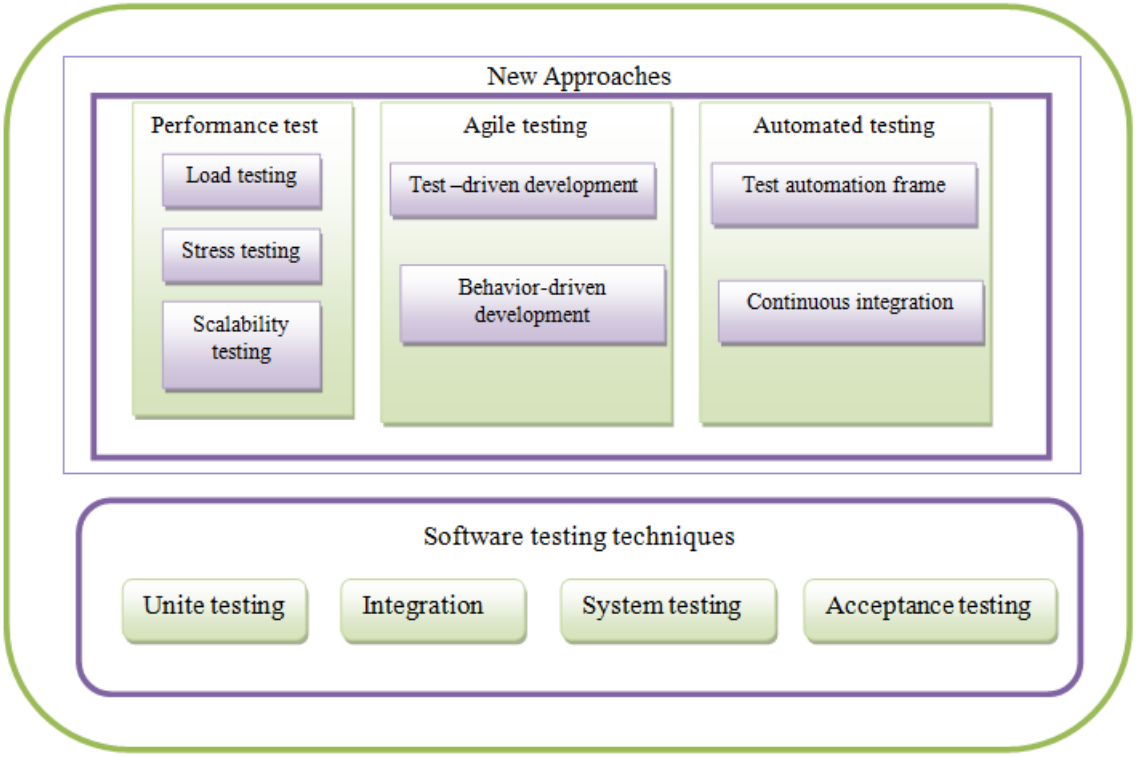

Figure.8. Structural model of New approaches

Comparative study

To compare the efficacy and efficiency of innovative and classic testing methods, a comparison study was carried out. Analysis was done on metrics like affordability, test implementation duration, and finding defects rate.

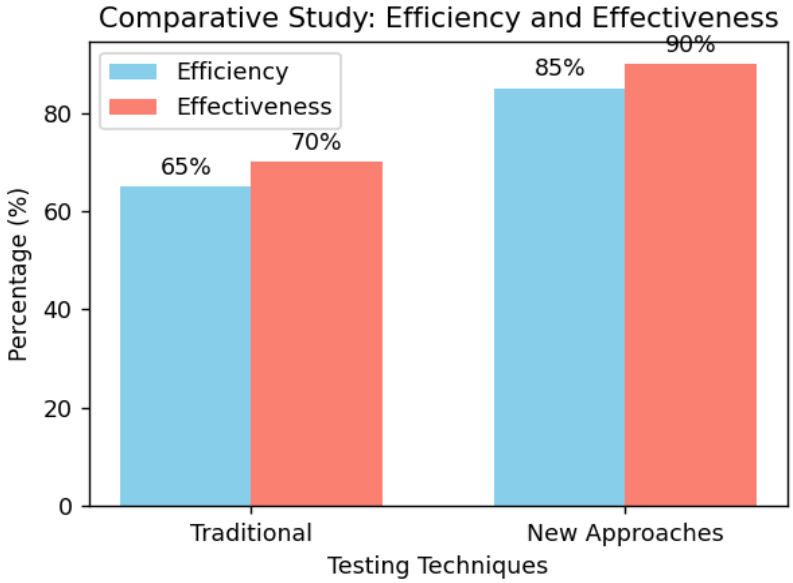

Figure.9. assessment of conventional and novel approaches

Figure 9 illustrates the comparative study of efficiency and effectiveness between traditional and new testing approaches. In this example code, we are using mock data for demonstration. They are similar to software testing approaches that have been used and perfected in the industry for decades. These approaches mainly test whether a piece of software behaves according to non-functional requirements through manual processes. Some of the most important traditional methods :

1. **Black-Box Testing**

Ignores the internal code foundation of the software and concentrates on testing it. Input/output definitions serve as the foundation for test cases. Evaluation of usability and functional testing are two examples.

2. **White-Box Testing**

Needs familiarity with the software's core operations. To make that the logic, loops, and requirements in the code are operating correctly, tests are carried out. Examples include integration and unit testing.

3. **Manual Testing**

Performed done by manual test case execution by human testers without the use of automation technologies. Incorporates exploratory testing, in which programmers actively search for flaws in the application.

4. **Regression Testing**

Verifies that current functioning has not been negatively impacted by recent code modifications. repeated following upgrades or bug fixes

5. **Waterfall Testing**

According to the waterfall approach of software construction, testing is carried out in a sequential manner. Only once the manufacturing stage is over are tests carried out.

New Approaches in Software Testing

Technology breakthroughs and evolving development methodologies have given rise to novel software testing methodologies that tackle contemporary issues including automation, flexibility, and rapid delivery. These methods frequently make use of automation, artificial intelligence, and integration with DevOps and agile processes.

a. **Automated Testing**

Utilizes scripts and tools that execute test scenarios. Examples include JUnit, Appium, and Selenium.

b. **AI-Driven Testing**

Uses machine learning along with artificial intelligence to create test cases, forecast errors, and streamline testing procedures. Some examples are Applitools and Test.ai.

c. **Continuous Testing in DevOps**

The constant integration/continuous delivery (CI/CD) pipeline incorporates testing. guarantees quick feedback on the quality of the code at each advancement step.

d. **Shift-Left Testing**

places a strong emphasis on testing at the beginning of the development process. encourages testers and developers to work together to find flaws early.

e. **Behavior-Driven Development (BDD) and Test-Driven Development (TDD)**

BDD: Ensures that tests are written in a way that non-technical stakeholders can comprehend. Developing test cases prior to developing the actual code is the main goal of TDD.

f. **Cloud-Based Testing**

Cloud infrastructure is tested, allowing for scalability as well as testing across many devices and situations. Sauce Labs and Browser Stack are two examples.

g. **Performance and Load Testing with Modern Tools**

Focuses on evaluating software's dependability and scalability across a range of scenarios. Examples include LoadRunner and Apache JMeter.

h. **Mobile and IoT Testing**

Internet of Things (IoT) devices and mobile applications require specialized testing. comprises security, functionality, and compatibility testing for various systems and devices.

i. **Security Testing**

on locating software flaws Examples include static and dynamic application security testing (SAST/DAST), ethical hacking, and penetration testing.

Key Difference

Aspect	**Traditional Approaches**	**New Approaches**
Automation	Mostly Manual	Highly Automated
Development Models	Water fall V Model	Agile, DevOps
Speed	Time Intensive	Faster due to CI/CD integration
Scalability	Limited	Cloud based testing enables scalability
Technology Integration	Minimal	AI, ML and Cloud

Table.1. Key differences for Traditional vs New approaches

According to the needs and limitations of the project, organizations frequently combine innovative and classic methodologies, each of which has advantages and disadvantages of its own. Table 1 is displayed.
Conventional Methods: Ideal for modest, less complex projects with clearly specified needs.
Novel Methods: Perfect for large-scale, intricate, and fast-paced projects that demand constant delivery and high-quality guarantee

Conclusion

In this paper, it is elaborated what is software testing and how many methods and strategies it has. Software testing is the act of operating a system under controlled conditions and evaluating the results. Since testing is a prerequisite for the final delivery of the product, it is the most important aspect of the software development lifecycle. It is a deep process that takes time to absorb, thus smarter ways & new methods are needed. So automatic checking out and different various checking out at metrics implementation previous and in the checking out system at preferential. it can lovely the current inspecting methods, both for time productivity beside for natural and dependable end products that no more handiest fits required requirements one however moreover gifts with the maximum running performance. Black-box, white-box, and manual or exploratory testing is the three pillars of software quality assurance. These approaches work well in contexts where specifications are clear, systems are relatively simple, and testing can go in succession. They allow the software to be tested against the definitions creating handy structure to identify functional errors and if the software is performing as it should as per the specifications. However, classic methods were often clumsy in the face of modern software development, which demands fast iterations, scaling, and real-time adaptability. This can lead to higher costs and prolonged delivery cycles as they still depend on manual processes, late defect detection, and lack of automation. Further, such approaches may not be sufficient to address complexities posed by cloud technologies, mobile platforms, and AI-powered systems.

References

1. Jamil, M. A., Arif, M., Abubakar, N. S. A., & Ahmad, A. Software testing techniques: A literature review. In 2016 6th international conference on information and communication technology for the Muslim world (ICT4M) (pp. 177-182). IEEE. 2016.
2. B. Agarwal et al., “Software engineering and testing”. Jones & Bartlett Learning, 2010.
3. K. Bogdan. “Automated software test data generation”. Software Engineering, IEEE Transactions on 16.8 (1990): 870-879
4. Guide to the Software Engineering Body of Knowledge, Swebok, A project of the IEEE Computer Society Professional Practices Committee, 2004.
5. E. F. Miller, “Introduction to Software Testing Technology”, Software Testing & Validation Techniques, IEEE, 1981, pp. 4-16
6. J.Irena. “Software Testing Methods and Techniques”, 2008, pp. 30-35
7. P. Ron. Software testing. Vol. 2. Indianapolis: Sam’s, 2001.
8. S. Amland, "Risk-based testing:" Journal of Systems and Software, vol. 53, no.3, pp. 287–295, Sep. 2000.
9. Redmill and Felix, “Theory and Practice of Risk-based Testing”, Software Testing, Verification and Reliability, Vol. 15, No. 1, March 200
10. B. Agarwal et al., “Software engineering and testing”. Jones & Bartlett Learning, 2010.
11. K. Bogdan. “Automated software test data generation”. Software Engineering, IEEE Transactions on 16.8 (1990): 870-879.
12. Jacobson et al. The unified software development process. Vol. 1. Reading: Addison-Wesley, 1999.

13. Everett et al., “Software testing: testing across the entire software development life cycle”. John Wiley & Sons, 2007.
14. Guide to the Software Engineering Body of Knowledge, Swebok, A project of the IEEE Computer Society Professional Practices Committee, 2004.
15. E. F. Miller, “Introduction to Software Testing Technology”, Software Testing & Validation Techniques, IEEE, 1981, pp. 4-16.[11]M. Shaw, “Prospects for an engineering discipline of software,” IEEE Software, November 1990, pp.15-24.
16. D. Nicola et al. "A grey-box approach to the functional testing of complex automatic train protection systems." Dependable Computing-EDCC 5. Springer Berlin Heidelberg, 2005. 305-317
17. J. A. Whittaker, “What is Software Testing? And Why Is It So Hard?” IEEE Software, 2000, pp. 70-79.
18. N. Jenkins, “A Software Testing Primer”, 2008, pp.3-15.
19. Luo, Lu, and Carnegie, "Software Testing Techniques Technology Maturation and Research Strategies’, Institute for Software Research International-Carnegie Mellon University, Pittsburgh, Technical Report, 2010.
20. M. S. Sharmila and E. Ramadevi. "Analysis of performance testing on web application." International Journal of Advanced Research in Computer and Communication Engineering, 2014
21. J. A. Whittaker, “What is Software Testing? And Why Is It So Hard?” IEEE Software, 2000, pp. 70- 79.
22. N. Jenkins, “A Software Testing Primer”, 2008, pp.3-15.
23. Luo, Lu, and Carnegie, "Software Testing Techniques Technology Maturation and Research Strategies’, Institute for Software Research International-Carnegie Mellon University, Pittsburgh, Technical Report, 2010.
24. M. S. Sharmila and E. Ramadevi. "Analysis of performance testing on web application." International Journal of Advanced Research in Computer and Communication Engineering, 2014.
25. M. Shaw, “Prospects for an engineering discipline of software,” IEEE Software, November 1990, pp.15-24
26. D. Nicola et al. "A grey-box approach to the functional testing of complex automatic train protection systems." Dependable ComputingEDCC 5. Springer Berlin Heidelberg, 2005. 305- 317
27. Jamil, M. A., Arif, M., Abubakar, N. S. A., & Ahmad, A. Software testing techniques: A literature review. In 2016 6th international conference on information and communication technology for the Muslim world (ICT4M) (pp. 177-182). IEEE. 2016
28. A. Memon, “A Uniform Representation of Hybrid Criteria for Regression Testing”, Transactions on Software Engineering (TSE), 2013.
29. Infosys, “Metric model”, white paper, 2012. Data retrieved from (http://www.infosys.com/ engineering -services /whitepapers/Documents /comprehensive-metrics-model.pdf)
30. B. Boehm, “Some Future Trends and Implications for Systems and Software Engineering Processes”, 2005, pp.1-11.
31. R. Ramler, S. Biffl, and P. Grünbacher, "Valuebased management of software testing," in ValueBased Software Engineering. Springer Science Business Media, 2006, pp. 225– 244.
32. D. Graham, "Requirements and testing: Seven missing-link myths," Software, IEEE, vol. 19, 2002, pp. 15-17