IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 3, January 2022
Impact Factor: 6.252

A Review of the Literature on Software Testing Techniques

Khusbu Ruparel[1] and Anish Kamble[2]
Assistant Professor, BSC IT, Suman Education Society's LN College, Borivali East, Mumbai, India[1]
Student, BSC IT, Suman Education Society's LN College, Borivali East, Mumbai, India[2]

Abstract: *The rising complexity of today's software applications, along with increased competitive pressure, has pushed quality assurance of produced software to new heights. Software testing is an unavoidable aspect of the Software Development Lifecycle, and its importance in the pre and post development processes necessitates the use of improved and efficient procedures and techniques. The goal of this study is to describe existing as well as updated testing approaches for better quality assurance.*

Keywords: Testing Methodologies, Software Testing Life Cycle, Testing Frameworks.

I. INTRODUCTION

Testing is described as the process of determining whether or not a certain system satisfies its originally established criteria. It is mostly a validation and verification process. The process of determining if the produced system fits the user's needs. As a result of this action, there is a discrepancy between the actual and predicted results. Finding defects, faults, or missing requirements in a developed system or programme is referred to as software testing. As a result, this is an inquiry that provides stakeholders with precise information on the product's quality. Software testing may also be thought of as a risk-taking activity. The main thing for software testers to learn during the testing process is how to reduce a huge number of tests into manageable test sets and make intelligent judgements about them. what dangers should be tested and which should not [1]. the discovered association between testing costs and mistakes. Figure 1 clearly indicates that the expense of testing both categories, functional and non-functional, increases considerably. When deciding what to test or how many tests to run, many bugs might be missed. The effective testing aim is to do the least number of tests possible so that extra testing work is minimised. software testing is an important component of software quality assurance. The relevance of testing may be seen in life-critical software (for example, flight simulators). control) testing, which can be costly due to the danger of schedule delays, cost overruns, or outright cancellation [2], and more on this Testing involves several degrees and processes, and the individual who performs the testing varies from level to level. Unit testing, integration testing, and system testing are the three primary processes in software testing. Each of these procedures is checked by either the software developer or the quality assurance engineer, commonly known as a software tester.

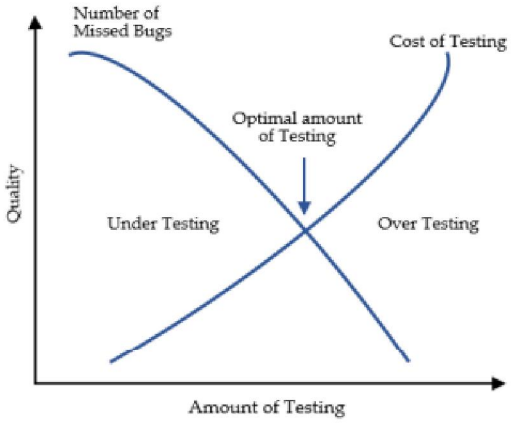

Figure 1: Every Software Project has optimal test effort

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 3, January 2022
Impact Factor: 6.252

The above-mentioned testing steps are all part of the Software Development Lifecycle (SDLC). It is critical to divide software development into phases. A collection of modules, each of which is allocated to a distinct team or individual. Following the completion of each module or unit, the developer tests it to see if the produced module works as expected; this is known as Unit Testing. Integration testing is the second phase in the SDLC testing process. Once the modules of a single software system have been built individually, they are merged together, and problems in the build frequently occur once the integration has been completed. The SDLC's last testing stage is System Evaluating, which involves testing the entire software from every angle. Furthermore, software testing guarantees that the integrated components do not interact with or disrupt any other module's programming. However, testing large or very complex systems may be a time-consuming and lengthy process since the more components in the programme, the more difficult it is to test each combination and scenario, resulting in a desperate need for improved software testing processes for premium optimization. The testing cycle is divided into numerous parts, ranging from test planning through test result analysis. The first step, test planning, is primarily concerned with the organisation of all test operations. should be carried out throughout the testing procedure. The second step of the testing life cycle is test development, which is where the test cases that will be utilised in the testing process are created. The following part of the Testing cycle is test execution, which includes the execution of test cases, and the applicable issues are reported in the following phase, which is test reporting.

II. CURRENT TESTING METHODS

The initial step in starting the testing process is to build test cases. For effective and accurate testing, test cases are created utilising a variety of testing approaches. Black box testing, white box testing, and grey box testing are the three primary testing approaches [8]. White Box testing is extremely successful since it checks not only the functionality of the programme but also the internal structure of the application. Programming abilities are required for developing test cases to do white box testing. White box testing is also known as transparent box testing or glass box testing. This type of testing can be used at any level, including unit, integration, and system testing. This sort of testing is also known as security testing since it determines whether the information systems secure data and preserve the desired functionality. Because this type of testing procedure makes advantage of the software's internal logical layout, it is capable of testing all of a module's independent routes, every logical choice is exercised, all loops are verified at each boundary level, and internal data structures are also exercised. White box testing, on the other hand, serves a function in that it is a sophisticated testing procedure owing to the involvement of programming skills in the testing process. Black Box testing is a testing approach that focuses on the functionality of an application without delving into its technical details. This approach is applicable to Every stage of the SDLC's testing. It primarily conducts testing in such a way that it covers every component of the programme in order to assess whether or not it fits the user's initially set criteria. It is capable of detecting improper functionality by evaluating it at each minimum, maximum, and base case value. It is the most basic and widely used testing method in the world.

Figure 2: Software Testing Techniques

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 3, January 2022
Impact Factor: 6.252

Grey Box Testing is a hybrid of the White Box and Black Box Testing Techniques, combining the benefits of each. The demand for such testing arose because in this sort of Because the tester is aware of the internal structure of the programme, he or she can test the functionality more effectively by taking the internal structure of the application into account. Figure 2 is cited by author J. Irena [8] and expanded upon in our study work.

2.1 Software Testing Life Cycle (STLC)

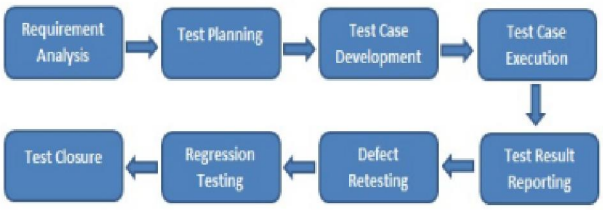

Figure 3: Software Testing Life Cycle [12]

We go through the STLC procedures, stages, and phases that software goes through during the testing process. STLC is a defined standard for software or applications that differs from area to region throughout the world. During the first step of the STLC, the Quality Assurance team reviews the software requirements to ensure that they understand the basic criteria against which the test will be run. If a disagreement emerges, the team must work with the development team to better understand and address the issue. Test planning is the second and most significant part of the STLC since it defines the whole testing approach. This phase is concerned with the creation of the test plan, which will be the phase's final output. The Test Plan is a necessary document that is geared at functional testing of the application. which the testing method is impractical [11]. The test designing phase is when the test case is created and the test planning activity is completed. The QA team writes manual test cases or, in certain situations, generates automated test cases. A test case defines a collection of test inputs or data, execution circumstances, and expected outcomes. The stated set of test data should be chosen in such a way that it delivers both anticipated results and purposely erroneous data that causes an error throughout the test.

The Test Execution Phase consists of running the test cases according to the test plan that was created prior to the execution phase. If the functionality successfully completes the execution phase the test is stated to be cleared or succeeded if no bugs are reported, and each failed test case is connected with the detected issue or error. The output of such an action is a defect or bug report. Test reporting is the reporting of created outcomes following the execution of test cases, which also includes problem reporting, which is subsequently given to the development team to be repaired [11].

2.2 Software Release Life Cycle

This life cycle follows the STLC and includes additional testing processes such as Alpha and Beta testing. Alpha testing, where Alpha refers to the first step of application testing at the developer's end, can be done using either the white box approach or the grey box technique. An alpha release is defined as testing at either the integration or system level utilising a black box approach. The alpha testing concludes with a feature freeze, which usually indicates that no additional features will be introduced to either enhance the functionality or for any other reason. The Beta Testing step follows Alpha Testing and may be called formal acceptance testing because it is performed by the user following the Alpha release. The programme or software is for testing purposes, it was made available to a certain group of users. Before a programme is formally published, a beta version is usually made accessible to the targeted audience for input. The intended audience is frequently referred to as Beta Testers, and the application may be referred to as a prototype version of the programme designed primarily for demonstration reasons. As a result, the final version of the programme is published following Beta Testing.

III. ENHANCEMENT IN TESTING PROCESSES

By using Combinational Criteria, Test Suite Prioritisation improves the testing process. The primary mechanism underlying such test case prioritisation is the conversion of the Weblogs into the applicable test suites for the user session, and then writes it down in an XML format. The coverage based on combinatorial test suites should

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 3, January 2022
Impact Factor: 6.252

appropriately prioritise the Algorithm utilised for this method. Furthermore, empirical studies should be conducted to assess the efficiency of the given application and its associated test suites. C-PUT is a tool used in this respect that simply transforms the logs of web applications into test suites that are formatted in XML; it is then used to provide functionality for the prioritisation of these tests. There is ongoing research to see whether these test suite prioritising strategies may be utilised to improve the defect detection ratio [18] [19]. The application of genetic algorithms (GAs) for the purpose of automated test data generation for testing the application is yet another improvement in the testing process, as previously the dynamic means of test data generation remained a major issue in the software testing process, so the use of Genetic Algorithm based testing is an effective of the test data generation, and it is also capable of handling the data generation in accordance with the complexity of the programme.

3.1 Automated Testing

The biggest advancement in the testing process leads to Test Automation, which is the use of specific software to carry out the testing process as well as the comparison of actual results with predicted outcomes. The test automation approach saves time by eliminating the need for manual testing, which can be time-consuming. Test Automation happens throughout both the implementation and testing phases of the SDLC. Test Automation is increasingly being used instead of manual testing throughout the world since it saves time by completing testing operations in less time. Test automation has replaced manual testing by decreasing the need for it and revealing the number of faults. When done manually, one of the key testing kinds, regression testing, takes a long time. It often examines whether the programme or application functions correctly after the installation. Bugs and mistakes must be fixed. Because the error or bug ratio in the code or application may increase after the error is fixed. So, in order to reduce the time required for regression testing, a collection of automated test suites is assembled to build a regression test suite. Test automation also aids in the early detection of problems, saving a significant amount of money and energy at a later stage The environment that serves a word is commonly known as the Automation Testing Execution Framework. The testing framework is primarily in charge of carrying out the tests. as well as providing the structure for expressing expectations and reporting results. The application independence of the Testing Framework is a distinguishing quality that makes it extensively usable in numerous sectors throughout the world [21]. Modular, Data Driven, Keyword Driven, and Hybrid testing frameworks are examples. The Modular Testing Framework is founded on the notion of abstraction, which entails writing multiple scripts for different modules of the programme or application to be tested, thereby abstracting each component from another level. This modular separation facilitates scalability and easy management of automated test suites. Furthermore, after the feature is implemented, With the provided driver scripts in the library, creating new driver scripts for different sorts of testing becomes simple and quick. The main disadvantage of such frameworks is that they embed data within them, thus when changes or upgrades are required in the test data, the entire code of the test script must be rewritten. It was the primary reason for the development of the Data Driven Testing Framework. In this sort of Framework, the test data and expected outcomes are ideally saved in separate files, facilitating the execution of a single driver script. This type of Framework minimises the number of test scripts as well as the amount of code required for the development of test cases, allowing for greater flexibility in the correction of faults or problems. Testing based on keywords Framework employs self-explanatory terms known as Directives. A framework of this sort is used to explain the activities that are expected to be done by the programme or application being tested. Because the data and the directives are stored in distinct data files, this type of testing is really an extension of Data Driven Testing. It incorporates all of the benefits of the data-driven testing paradigm. Another significant advantage is the reusability of the keywords. The disadvantage of this type of testing framework is that it adds complexity to the testing process owing to the use of keywords. The framework lengthens and complicates test cases. As a result, it is necessary to combine the strengths of all frameworks in order to mitigate the negative characteristics that they contain. A hybrid method is preferred for use since it is primarily a combination of all three approaches, and this combination merges the benefits of all testing frameworks, making it the most efficient.

3.2 Testing Frameworks in the Agile

Another breakthrough in software testing is the agile lifecycle, which includes short and quick test cycles with rapidly changing requirements. As a result, the agile environment might include any testing framework, however owing to

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 3, January 2022
Impact Factor: 6.252

frequent iterations and fast changes in defined requirements, maintaining a test automation suite becomes tough. Though testing frameworks are still a poor fit for the agile environment due to the difficulty in getting maximum code and functionality coverage.

3.3 Test Driven Development (TDD)

It is a strategy that use automated unit tests for the goal of driving software design and pushing the decoupling of dependencies. During the traditional testing method, testers frequently discover one or more faults or mistakes, but TDD provides a crystal-clear measure of success when the test no longer fails, increasing confidence in the system's essential requirements. TDD can save a significant amount of time that would otherwise be lost during the debugging process [21]. BDD (Behaviour Driven Development) is primarily an extension of Test-driven Development that focuses on the system's behavioural elements rather than the implementation level aspects. As a result, the testing process is more efficient since there is a clear knowledge of what the system is meant to perform. As a result, BDD is mostly test-driven. Development is combined with acceptance testing, which often refers to performing a test to evaluate whether or not the given requirements of the product or programme are satisfied. When it is carried out by the intended customer or user, it is referred to as User Acceptance Testing. [22]

IV. METRICS OF TESTING

4.1 Prioritization Metrics

The use of Test Metrics is critical since they may significantly improve the efficacy of the testing process. They are a key measure of efficiency and effectiveness. accuracy and analysis of stated metrics. They may also assist in identifying areas that demand improvement, as well as the subsequent action or step that must be made to eradicate it. Test Metrics are more than just a single stage in the STLC; they serve as an umbrella for the continuous development of the whole testing process. Software Testing Metrics concentrate on quality aspects important to the process and product and are divided into Process Quality Metrics and Product Quality Metrics, both of which strive to improve not just the testing process but also the product. However, there is a major issue with the current testing process: aligning the testing technique with the application being created. Not every testing strategy can be incorporated in every application to be built. For example, testing a network protocol software vs testing a specific e-commerce application would be considerably different, with entirely different test case complexity, highlighting the importance of human engagement in the testing process rather than relying just on existing test cases. Prioritisation The duration of the test based on certain HTTP requests within a test case is one of the metrics. Frequency based prioritising improves the testing process by selecting test cases that include the most frequently used pages for execution before those that do not.

4.2 Metrics for Process Quality

A process is the most important component since it is capable of providing a high-quality result in the shortest amount of time and at the lowest possible cost. This is the ultimate reason why organisations all over the world have focused on improving process performance, and this is precisely where the demand for metrics developed, since it is essential to evaluate the process from numerous aspects efficiently. The fundamental metric of process quality is measuring process efficiency, which includes measures of parameters like as the test progress curve represents the planned progress of the Testing Phase as defined by the test plan. The cost of testing is the metric's next important step, both phase and component wise. The main goal of which is to assist in identifying the portions that require further testing. and the expense that they will incur as a result of it. Average Defect Turnaround Time is another indicator that displays the average time taken by the testing team to verify faults. The average defect response time is a measure that indicates operational efficiency. It is a measure of the average time it takes the team to respond to mistakes. Metrics for Process Effectiveness guarantee that the finished application or product is of high quality. Test coverage, defect removal efficiency, and compliance Its key sections are the Volatility Index, failed and completed test cases. guaranteeing an overall better Testing Process. Furthermore, the adoption of RTM (Need Traceability Matrix) can result in a better Testing Process since it links each test case to a specific requirement, making testing more accurate.

IJARSCT
ISSN (Online) 2581-9429
International Journal of Advanced Research in Science, Communication and Technology (IJARSCT)
Volume 2, Issue 3, January 2022
Impact Factor: 6.252

V. CONCLUSION AND FUTURE WORK

Testing is the most important phase of the Software Development Lifecycle since it affects the ultimate delivery of the product. It is a time-consuming and demanding procedure, thus improved approaches and creative methodologies are required. This allows for the deployment of Automated Testing and other other Test Metrics both before and during the testing process. It has the potential to improve existing testing procedures in terms of both time effectiveness and efficient and dependable final products that not only fulfil the set specifications but also deliver optimum operational efficiency. The platform on which software development and testing take place is constantly evolving and remains extremely important. However, something as important and necessary as testing occurs fairly late in the Software Development process. There should be as much contact as possible between specification authors and testers for better understanding and early review, which may resolve ambiguity issues and save money on later software fixes. After clarifying the criteria and needs, testers should send over a lightweight test model to developers so that they may ensure the primary specifications are satisfied before processing the project for formal testing. The use of simulation tools may greatly assist testers in simulating a comparable environment in which the product is intended to function, as well as particular exception testing and techniques for the product. The optimum way to handle exceptions may be established. While testing the product in the same testing environment that the product is intended for, this may be readily accomplished by including simulation into the testing process. As a result, future work in the testing process will be much more technology reliant, leveraging the simulation and automated testing model-based approach, not only speeding the testing life cycle but also delivering optimal bug avoidance and efficient quality assurance.

REFERENCES

[1]. P. Ron. Software testing. Vol. 2. Indianapolis: Sam's, 2001.
[2]. S. Amland, "Risk-based testing:" Journal of Systems and Software, vol. 53, no. 3, pp. 287–295, Sep. 2000.
[3]. Redmill and Felix, "Theory and Practice of Risk-based Testing", Software Testing, Verification and Reliability, Vol. 15, No. 1, March 2005.
[4]. B. Agarwal et al., "Software engineering and testing". Jones & Bartlett Learning, 2010.
[5]. K. Bogdan. "Automated software test data generation". Software Engineering, IEEE Transactions on 16.8 (1990): 870-879.
[6]. Jacobson et al. The unified software development process. Vol. 1. Reading: Addison-Wesley, 1999.
[7]. Everett et al., "Software testing: testing across the entire software development life cycle". John Wiley & Sons, 2007.
[8]. J.Irena. "Software Testing Methods and Techniques", 2008, pp. 30-35.
[9]. Guide to the Software Engineering Body of Knowledge, Swebok, A project of the IEEE Computer Society Professional Practices Committee, 2004.
[10]. 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
[12]. 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.