Software Testing Techniques: A Literature Review

Neha Sharma[1], Dr. Shilpi Singh[2]
[1]*Research Scholar, Dept. of Computer Science & Engineering, National Institute of Technology Patna*
[2]*Asst. Prof., Dept. of Computer Science & Engineering, Amity University Patna*

Abstract **- With the growing complexity of today's software applications injunction with the increasing competitive pressure has pushed the quality assurance of developed software towards new heights. Software testing is an inevitable part of the Software Development Lifecycle and keeping in line with its criticality in the pre and post development process makes it something that should be catered with enhanced and efficient methodologies and techniques. This paper aims to discuss the existing as well as improved testing techniques for the better-quality assurance purposes.**

Index Terms **- Testing Methodologies, Software Testing Life Cycle, Testing Frameworks, Automation Testing, Test Driven Development, Test optimisation, Quality Metrics**

I.INTRODUCTION

Testing is characterized as a cycle of assessment that either the framework meets its initially indicated necessities or not. It is principally a cycle incorporating approval and check measure that whether the created framework meets the prerequisites characterized by client. Consequently, this action brings about a distinction among genuine and anticipated outcome. Programming Testing alludes to discovering bugs, mistakes or missing necessities in the created framework or programming. Thus, this is an examination that furnishes the partners with the specific information about the nature of the item.

Programming Testing can likewise be considered as a danger-based action. The significant thing during testing measure the product analysers must comprehend that how to limit countless tests into sensible tests set, and settle on insightful choices about the dangers that are critical to test or what are not [1].

Figure 1 shows the testing cost and blunders found a relationship. The Figure 1 obviously shows that cost goes up significantly in testing the two kinds for example utilitarian and non-functional. The dynamic for what to test or decrease tests then it can cause to miss numerous bugs. The successful testing objective is to do that ideal measure of tests so additional testing exertion can be limited [1].

As per Figure 1, Software testing is a significant segment of programming quality confirmation. The significance of testing can be considered from life-basic programming (e.g., flight control) testing which can be exceptionally costly due to chance with respect to plan delays, cost invades, or out and out scratch-off [2], and more about this [3][4].

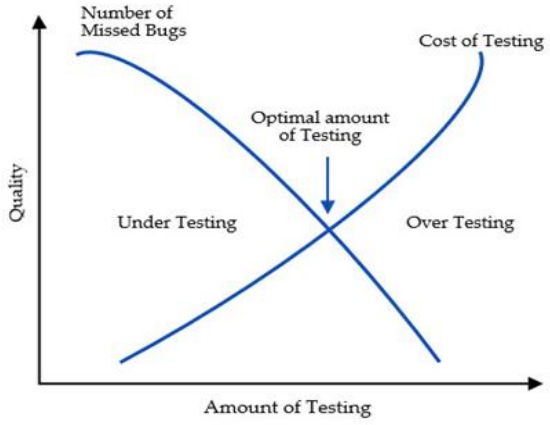

Figure 1: Every Software Project has optimal test effort (Courtesy [1])

Testing has certain levels and steps as indicated by which the individual who does the testing contrasts from level to level. The three essential strides in the product testing are Unit trying, Integration testing and System testing. Every one of these means is either tried by the product designer or the quality confirmation engineer who is otherwise called a product analyzer [5]. The testing referenced above advances is comprehensive in the Software Development Lifecycle (SDLC). It is basic to break the product advancement into a lot of modules where every module allotted to an alternate group or diverse person. After the finish of every module or unit, it is tried by the designer just to check whether the created module is working by the desire or not, this is named as Unit Testing. The second step of testing inside the

SDLC is Integration Testing. When the modules of a solitary programming framework have been grown freely, they are incorporated together and regularly blunders emerge in the construct once the coordination has been finished. The last testing step in the SDLC is System Testing, which is trying of the entire programming from every single viewpoint. Likewise, programming testing guarantees that the coordinated units don't meddle or upset the programming of some other module. Notwithstanding, testing of a huge or strongly complex frameworks may be a very tedious and extensive methodology as the more segments inside the application, the more troublesome it gets the chance to test every blend and situation, subsequently driving towards a desperate requirement for upgraded programming testing measure for premium streamlining [6].

Testing cycle is basically made out of a few stages, from Test Planning to the examination of Test Results. Test Planning being the principal stage is chiefly the arrangement of all the test exercises that are to be led in the entire testing measure. Test Development is the second period of the testing life cycle, where the experiments that are to be utilized in the testing cycle are created. Test execution is the following period of the Testing cycle that includes the execution of the tests cases, and the applicable bugs are accounted for in the following stage that is the Test Reporting stage. Test outcome Analysis is the last phase of the testing cycle in which the deformity examination is finished by the designer who built up the framework or the product, this progression can likewise be taken care of alongside the customer as it will help in the better comprehension of what to overlook and what precisely to fix or upgrade or just change [7].

II. EXISTING TESTING METHODS

For the initiation of the Testing cycle, the initial step is to create experiments. The experiments are created utilizing different testing strategies, for the successful and exact testing. The significant testing procedures are Black box testing, White Box testing and Gray Box testing [8].

White Box testing is altogether powerful as it is the technique for testing that tests the usefulness of the product as well as tests the inward structure of the application. While planning the experiments to lead white box testing, programming aptitudes are essential to plan the experiments. White box testing is likewise called clear box or glass box testing. This sort of testing can be applied to all levels including unit, coordination or framework testing. This sort of testing is additionally considered Security Testing that is it satisfies the need to decide if the data frameworks ensure information and keeps up the expected usefulness. As this sort of testing measure utilizes the inward coherent course of action of the product subsequently it is fit enough of testing all the autonomous ways of a module, each intelligent choice is worked out, all circles are checked at every limit level, and interior information structures are additionally worked out. Be that as it may, white box testing fills a need for being a perplexing testing measure because of the incorporation of programming aptitudes in the testing cycle [9][10].

Discovery testing is a trying method that basically tests the usefulness of the application without going into its usage level detail. This method can be applied to each degree of testing inside the SDLC. It predominantly executes the testing so that it covers every single usefulness of the application to decide if it meets the at first determined prerequisites of the client or not. It is fit for finding inaccurate functionalities by testing their usefulness at every base, greatest and base case esteem. It is the most straightforward and far and wide testing measure utilized overall [9] [10].

Figure 2: Software Testing Techniques [8]

Grey Box Testing is the combination of the White Box and Black Box Testing Technique serving the advantages of both. The need for such kind of testing

aroused because in this type of testing the tester is aware of the internal structure of the application, hence testing the functionality in a better way taking the internal structure of the application into consideration. Figure 2 is referenced from author J. Irena [8] and further extended here in our research paper.

A. Software Testing Life Cycle (STLC)

Figure 3 discusses the STLC steps, stages and phases a software undergo during the testing process. Though, there is no fixed standard of the software or application undergoing STLC, and it varies from region to region throughout the world [11].

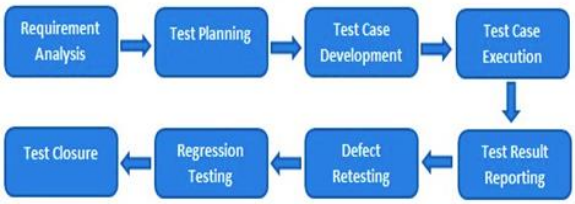

Figure 3: Software Testing Life Cycle [12]

During the main period of the STLC, the survey of the product prerequisites happens by the Quality Assurance group in which they comprehend the center necessities as indicated by which the test will be directed. On the off chance that on account of any contention emerges, the group must organize with the advancement group to all the more likely comprehend and resolve the contention. Test arranging is the second and most significant period of the STLC, as this the progression where all the testing system is characterized. This stage manages the arrangement of the test plan, which will be a definitive deliverable of this stage. Test Plan is an obligatory record one-sided towards the utilitarian testing of the application, without which the testing cycle is beyond the realm of imagination [11].

Test planning stage is where the experiment is created, and the test arranging action is stopped. Suitable Test cases are composed by the QA group physically or sometimes, mechanized experiments are created. Experiment indicates a lot of test information sources or information, execution conditions, and anticipated outcomes. The predefined set of test information ought to be picked with the end goal that it produces expected outcome just as purposefully incorrect information that will create a blunder during the test. This is normally done to check what conditions the application stops to perform [11].

Test Execution Phase is involved execution of the experiment's dependent on the test plan that was delivered before the execution stage. In the event that the usefulness passes the execution stage with no bug reportage, the test is supposed to be cleared or finished, and each bombed experiment will be related with the discovered bug or blunder. The deliverable of such movement is imperfection or Bug report.

Test Reporting is the revealing of the produced outcomes after the execution of the experiments which additionally includes bug detailing which at that point sent to the advancement group, so it tends to be fixed [11].

B. Software Release Life Cycle

This life cycle emerges after the STLC and it encompasses further testing process in which Alpha and Beta testing are inclusive.

Alpha Testing, in which Alpha refers to the first stage testing of the application at the developer's end, can be done via white box technique or grey box technique. The testing at either integration or system level testing could be done using black box approach, which is termed as an alpha release. The alpha testing ceases with a feature freeze, which typically means no more feature will be added to either extend the functionality or for any other purpose [13][14].

Beta Testing phase comes after Alpha testing and can be considered as a formal acceptance testing as it is done by the user, after the Alpha release. The software or the application is released to a certain intended group of users for the testing purpose. Usually, the beta version of the applications is made available to the targeted audience for feedback before it gets officially released. The targeted audience is often called Beta Testers, and the application may be termed as a prototype version of the software mainly for demonstration purposes. Hence, the final version of the software gets released after the Beta Testing [15] [16].

III. ENHANCEMENT IN TESTING PROCESSES

Test Suite Prioritization does upgrade in the testing cycle by Combinational Criteria. The significant procedure behind such experiment organizing is the transformation of the weblogs into the test suites important with the client meeting, and further recording it into a XML design. The Algorithm

utilized for this methodology ought to be precisely organized by the inclusion dependent on combinatorial test suites. Also, observational investigations ought to be completed for examining the adequacy of the particular application and its pertinent test suites [17]. A device utilized in such manner is known as C-PUT which basically arranges the logs of the web applications into test suites which are designed in XML; it is then utilized for the arrangement of the usefulness for the prioritization of these tests. There is a continuous exploration about if these test suite prioritization strategies can be utilized to upgrade the shortcoming discovery proportion or not [18] [19]. The use of hereditary calculations (GAs) with the end goal of robotized test information age for testing the application is one more upgrade in the testing cycle, as already the dynamic methods for test information age stayed a major issue in the product testing measure, so the use of Genetic Algorithm based testing is a compelling of the test information age, it likewise fit for taking care of the information age keeping in accordance with the intricacy of program [20].

A. Test Automation

The significant improvement in the testing cycle drives the testing cycle towards the Test Automation, which is the utilization of specific programming to do the testing cycle just as it makes the examination of real outcomes with the normal outcomes. Test Automation strategy is time successful, as it spares the hour of manual testing which can be very relentless.

In SDLC, Test Automation happens during the usage just as the testing stage. All through the world, Test Automation is being drilled rather than manual testing as it spares a lot of time achieving the testing measures in shorter time length. Test robotization has assumed control over the manual testing measure by diminishing its need just as by uncovering the measure of mistakes, shortages that cannot be recognized through the manual testing measure.

Relapse Testing being one of the significant testing types requires a lot of time when done physically. It normally tests whether the product or the application works appropriately after the obsession of any bugs or blunders. Since some of the time after the blunder obsession, the code or application's mistake or bug proportion gets much higher. Thus, for the evasion of the time taken for relapse testing; a lot of robotized test suites is made to frame a relapse test suite for such reason. Test Automation additionally helps in finding the issue at a lot prior stage, sparing loads of change cost and vitality at later stages [21].

Nature which provides food a term commonly realizes the mechanization testing execution called Testing Framework. The testing structure is fundamentally answerable for executing the tests, just as characterizing the arrangement in which to communicate desires and for the detailing of the outcomes. The champion component of Testing Framework that makes it broadly relevant in different areas overall is its application independency [21]. Testing Frameworks are of specific sorts, including Modular, Data Driven, Keyword Driven and Hybrid. The Modular Testing Framework depends on the rule of deliberation which includes the formation of various contents for various modules of the product or application that will be tried, in this way abstracting every single part from another level. This Modular division prompts the adaptability just as simpler support of the mechanized test suites. Likewise, when the usefulness is accessible in the library, the formation of various driver contents for various sorts of tests turns out to be simple and quick. The significant con of such kind of system is to implant information inside them, so when the adjustment or up degree is essential in the test information, the entire code of the test content needs to get changed. It was the significant reason that filled in as a reason for the creation of the Data Driven Testing Framework. In this sort of Framework, the test information and the normal outcomes are unmistakably put away inside various records, helping in the execution of single driver content having the option to execute all the experiments with numerous arrangements of information. This sort of Framework lessens the quantity of test contents just as limits the measure of code wanted for the age of experiments, gives greater adaptability in obsession of blunders or bugs.

Catchphrase driven testing Framework uses obvious watchwords which are named as Directives. Such sort of structure is utilized to clarify the activities that are relied upon to be performed by the product or application that will be tried. This sort of testing is a fundamentally expansion of Data Driven Testing as the information just as the orders are kept in independent information documents. It envelops all favorable circumstances of the information driven

testing system. Additionally, reusability of the catchphrases is another significant preferred position. The evil factor of this sort of testing structure is that because of the use of catchphrases, it adds unpredictability to the system making experiments longer and more intricate. Subsequently, to consolidate the qualities of all structures alleviating the evil components being controlled by them. A cross breed approach is viewed as best for the use as it is fundamentally a blend of the apparent multitude of three methodologies and this mix coordinates the upsides of all the testing systems, making it the most proficient one.

B. Testing Frameworks in the Agile

The coordinated lifecycle is another advancement in programming testing as it envelops short and quick test cycles with much of the time changing necessities. In this manner, the lithe condition can include any testing structure, yet because of the continuous cycles and quick change in determined prerequisites, it support of test computerization suite turns out to be very troublesome. Despite the fact that testing structures stays a terrible fit for the dexterous condition in light of the fact that accomplishing most extreme code and usefulness inclusion stays troublesome in it.

C. Test Driven Development (TDD)

It is a procedure that utilizes mechanized unit tests to drive the plan of programming and compelling the decoupling cycle of the conditions. With the standard testing measure, analyzer regularly discovers at least one imperfections or blunders, yet TDD gives a completely clear proportion of achievement when the test does not bomb anymore, upgrading the certainty level about the framework meeting its center particulars. Utilizing the TDD approach a lot of time can be spare that may get squandered over the troubleshooting cycle [21].

BDD (Behavior Driven Development) is essentially an augmentation of Test-driven Development zeroing in on the social parts of the framework as opposed to the usage level viewpoints. Thus, giving an away from of what precisely the framework should do giving more productivity to the testing cycle. Consequently, BDD is for the most part Test-driven Development consolidated with Acceptance testing, which commonly alludes to directing a test to decide whether the predefined necessity of the item or programming is met or not. On the off chance that it is performed by the expected client or client, at that point it is named as User Acceptance Testing [22].

IV. TESTING METRICS

A. Prioritization Metrics

The use of Test Metrics has prime noteworthiness as they can gigantically improve the adequacy of the testing cycle. They fill in as a significant pointer of the proficiency and accuracy and investigation of characterized measurements. They can likewise help in the recognizable proof of the zones which require improvement alongside ensuing activity or step that should be taken to dispense with it. Test Metrics are a solitary advance in STLC as well as goes about as an umbrella for the consistent improvement of the entire testing measure itself [23] [24].Software Testing Metrics center around the quality aspects pertinent to the cycle and item and are arranged into Process Quality Metrics, and Product Quality Metrics both of impulses plan to give upgrades in the testing cycle as well as in the item quality.

In any case, there lays a basic issue looked by the current testing measure which is coordinating of the testing approach with the application being created. Only one out of every odd testing approach can be actualized in each application to be created. For instance testing of an organization convention programming as contrasted and the testing of certain web based business application will be very extraordinary with totally unique experiments unpredictability, and that plots the criticality of human association inside the testing cycle and not simply simple dependence on the current test cases. Prioritisation Metrics incorporate the length of the test dependent on some HTTP demands inside an experiment. Recurrence based prioritization upgrades the testing cycle with the end goal that the experiments that incorporates most utilized pages are, chose for execution before those experiments that use less continuous ones [25][26].

B. Cycle Quality Metrics

A cycle is the most prominent part as it is equipped for creating a quality result inside minimal time in the most savvy way. This is a definitive explanation that why associations all through the world have put their

emphasis on the improvement of the cycles execution, and this precisely where the requirement for the measurements developed, as it is needed to check the cycle from different measurements productively. Estimating Efficiency of the cycle is the key measurement of cycle quality which includes certain estimations of components like Test progress Curve which portrays the arranged advancement of the Testing Phase by the test plan [27][28].

The expense of Testing is the following significant advance of the metric both stage insightful and segment savvy. The significant target of which is to help in recognizing the parts that require escalated testing and cost that they will bear as per it. Normal Defect Turnaround Time is another metric which portrays normal confirmation time by the testing group for the check of the imperfections. Normal Defect Response Time is the metric that is a pointer of the operational proficiency. It is the proportion of normal time taken by the group for reacting to the mistakes. Measurements for Process Effectiveness guarantees that the came about application or items will yield a great yield. Test inclusion, Defect Removal Efficiency, Requirement Volatility Index, fizzled and executed experiments being significant classes of it guaranteeing a general upgraded Testing Process.

Likewise, the utilization of RTM (Requirement Traceability Matrix) can bring about improved Testing Process, as it maps each experiment with indicates prerequisite, making the testing more precise [23] [24].

V. CONCLUSION AND FUTURE WORK

Testing is the most basic aspect of the Software Development Lifecycle, as it is something whereupon the last conveyance of the item is needy. It is tedious and a serious cycle, subsequently, improved procedures and inventive techniques are essential. This makes Automated Testing and different Test Metrics execution previously and during the testing cycle. It can improve the current testing strategies, both for time adequacy just as for effective and dependable last item which meets the predefined prerequisites as well as furnishes with most extreme operational productivity.

The stage over which the product improvement and testing dwell keeps on advancing and remains incredibly famous. Notwithstanding, something so significant and basic like Testing comes frequently very late during the time spent Software Development. There ought to be a greatest collaboration between determination journalists and Testers for better understanding and early survey, which may fix equivocalness issues and thusly bring about sparing the expense of later fixing of the product. Analyzers after being clear about the determinations and prerequisites should hand over engineers a specific lightweight test model, so they ensure the essential particular are met before dealing with the venture for legitimate testing. Utilization of reenactment devices can gigantically help the analyzers in making the comparative condition where the item is bound to run, certain special case testing and strategies for the exemption taking care of can be best decided. While testing the item in the comparative testing condition for which the item is intended for, and that can be effectively done by incorporating the recreation inside the Testing cycle. Consequently, the future work in importance with the testing cycle will be substantially more innovation subordinate saddling the reenactment and computerized testing model based methodology, speeding up the testing life cycle as well as giving ideal bug counteraction and effective quality confirmation.

REFERENCE

[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.
[13] J. A. Whittaker, “What is Software Testing? And Why Is It So Hard?” IEEE Software, 2000, pp. 70-79.
[14] N. Jenkins, “A Software Testing Primer”, 2008, pp.3-15.
[15] Luo, Lu, and Carnegie, "Software Testing Techniques Technology Maturation and Research Strategies’, Institute for Software Research International-Carnegie Mellon University, Pittsburgh, Technical Report, 2010.
[16] M. S. Sharmila and E. Ramadevi. "Analysis of performance testing on web application." International Journal of Advanced Research in Computer and Communication Engineering, 2014.
[17] S. Sampath and R. Bryce, Improving the effectiveness of Test Suite Reduction for User-Session-Based Testing of Web Applications, Elsevier Information and Software Technology Journal, 2012.
[18] B. Pedersen and S. Manchester, Test Suite Prioritization by Cost based Combinatorial Interaction Coverage International Journal of Systems Assurance Engineering and Management, SPRINGER, 2011.
[19] S. Sprenkle et al., "Applying Concept Analysis to User-session based Testing of Web Applications", IEEE Transactions on Software Engineering, Vol. 33, No. 10, 2007, pp. 643 - 658
[20] C. Michael, “Generating software test data by evolution, Software Engineering”, IEEE Transaction, Volume: 27, 2001.
[21] A. Memon, “A Uniform Representation of Hybrid Criteria for Regression Testing”, Transactions on Software Engineering (TSE), 2013.
[22] R. W. Miller, “Acceptance testing”, 2001, Data retrieved from (http://www.dsc.ufcg.edu.br/~jacques/cursos/map/recursos/Testin g05.pdf)
[23] Infosys, “Metric model”, white paper, 2012. Data retrieved from (http://www.infosys.com/ engineering -services /whitepapers/Documents /comprehensive-metrics-model.pdf)
[24] B. Boehm, “Some Future Trends and Implications for Systems and Software Engineering Processes”, 2005, pp.1-11.
[25] R. Bryce, “Test Suite Prioritisation and Reduction by Combinational based Criteria”, IEEE Computer Society”, 2014, pp.21-22.
[26] M. I. Babar, “Software Quality Enhancement for value-based systems through Stakeholders Quantification”, 2005, pp.359-360. Data retrieved from(http://www.jatit.org/volumes/Vol55No3/10 Vol55No3.pdf)
[27] R. Ramler, S. Biffl, and P. Grünbacher, "Value-based management of software testing," in Value-Based Software Engineering. Springer Science Business Media, 2006, pp. 225– 244.
[28] D. Graham, "Requirements and testing: Seven missing-link myths," Software, IEEE, vol. 19, 2002, pp. 15-17