2016 6th International Conference on Information and Communication Technology for The Muslim World

Software Testing Techniques: A Literature Review

Muhammad Abid Jamil[1,2], Muhammad Arif[2], Normi Sham Awang Abubakar [1], Akhlaq Ahmad[1, 3]
[1]KICT, International Islamic University, Kuala Lumpur, Malaysia
[2]College of Computer and Information System, Umm Al Qura University, Saudi Arabia,
[3]College of Engineering and Islamic Architecture, Umm Al Qura University, Saudi Arabia,
{majamil, hahamid, aajee}@uqu.edu.sa, nsham@iium.edu.my

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.**

Keywords— Testing Methodologies, Software Testing Life Cycle, Testing Frameworks, Automation Testing, Test Driven Development, Test optimisation, Quality Metrics

I. Introduction

Testing is defined as a process of evaluation that either the specific system meets its originally specified requirements or not. It is mainly a process encompassing validation and verification process that whether the developed system meets the requirements defined by user. Therefore, this activity results in a difference between actual and expected result. Software Testing refers to finding bugs, errors or missing requirements in the developed system or software. So, this is an investigation that provides the stakeholders with the exact knowledge about the quality of the product.

Software Testing can also be considered as a risk-based activity. The important thing during testing process the software testers must understand that how to minimise a large number of tests into manageable tests set, and make wise decisions about the risks that are important to test or what are not [1].

Figure 1 shows the testing cost and errors found a relationship. The Figure 1 clearly shows that cost goes up dramatically in testing both types i.e. functional and non-functional. The decision making for what to test or reduce tests then it can cause to miss many bugs. The effective testing goal is to do that optimal amount of tests so that extra testing effort can be minimised [1].

According to Figure 1, Software testing is an important component of software quality assurance. The importance of testing can be considered from life-critical software (e.g., flight control) testing which can be highly expensive because of risk regarding schedule delays, cost overruns, or outright cancellation [2], and more about this [3][4].

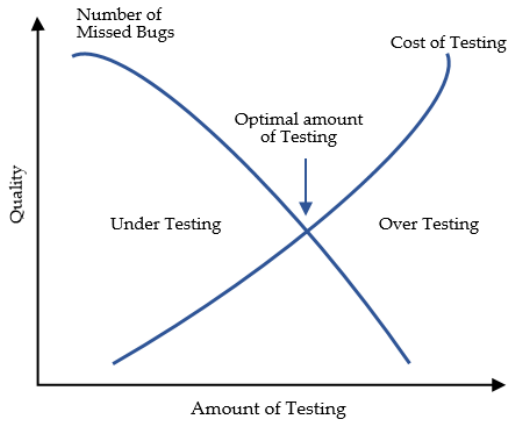

Figure 1: Every Software Project has optimal test effort (Courtesy [1]).

Testing has certain levels and steps according to which the person who does the testing differs from level to level. The three basic steps in the software testing are Unit testing, Integration testing and System testing. Each of these steps is either tested by the software developer or the quality assurance engineer who is also known as a software tester [5]. The testing mentioned above steps is inclusive in the Software Development Lifecycle (SDLC). It is essential to break the software development into a set of modules where each module assigned to a different team or different individual. After the completion of each module or unit, it is tested by the developer just to check whether the developed module is working by the expectation or not, this is termed as Unit Testing. The second step of testing within the SDLC is Integration Testing. Once the modules of a single software system have been developed independently, they are integrated together and often errors arise in the build once the integration has been done. The final testing step in the SDLC is System Testing, which is testing of the whole software from each and every perspective. Also, software testing ensures that the integrated units do not interfere or disturb the programming of any other module. However, testing of a large or intensely complex systems might be an extremely time-consuming and lengthy procedure as the more components within the application, the more difficult it gets to test each combination

978-1-5090-4521-1/16 $31.00 © 2016 IEEE
DOI 10.1109/ICT4M.2016.40

and scenario, consequently leading towards a dire need for enhanced software testing process for premium optimisation [6].

Testing cycle is mainly composed of several phases, from Test Planning to the analysis of Test Results. Test Planning being the first phase is mainly the plan of all the test activities that are to be conducted in the whole testing process. Test Development is the second phase of the testing life cycle, where the test cases that are to be used in the testing process are developed. Test execution is the next phase of the Testing cycle that encompasses the execution of the tests cases, and the relevant bugs are reported in the next phase that is the Test Reporting phase. Test Result Analysis is the last stage of the testing process in which the defect analysis is done by the developer who developed the system or the software, this step can also be handled along with the client as it will help in the better understanding of what to ignore and what exactly to fix or enhance or simply modify [7].

II. EXISTING TESTING METHODS

For the commencement of the Testing process, the first step is to generate test cases. The test cases are developed using various testing techniques, for the effective and accurate testing. The major testing techniques are Black box testing, White Box testing and Grey Box testing [8].

White Box testing is significantly effective as it is the method of testing that not only tests the functionality of the software but also tests the internal structure of the application. While designing the test cases to conduct white box testing, programming skills are requisite to design the test cases. White box testing is also called clear box or glass box testing. This kind of testing can be applied to all levels including unit, integration or system testing. This type of testing is also called Security Testing that is it fulfils the need to determine whether the information systems protect data and maintains the intended functionality. As this kind of testing process makes use of the internal logical arrangement of the software hence it is capable enough of testing all the independent paths of a module, every logical decision is exercised, all loops are checked at each boundary level, and internal data structures are also exercised. However, white box testing serves a purpose for being a complex testing process due to the inclusion of programming skills in the testing process [9][10].
Black Box testing is a testing technique that essentially tests the functionality of the application without going into its implementation level detail. This technique can be applied to every level of testing within the SDLC. It mainly executes the testing in such a way that it covers each and every functionality of the application to determine whether it meets the initially specified requirements of the user or not. It is capable of finding incorrect functionalities by testing their functionality at each minimum, maximum and base case value. It is the most simple and widespread testing process used worldwide [9] [10].

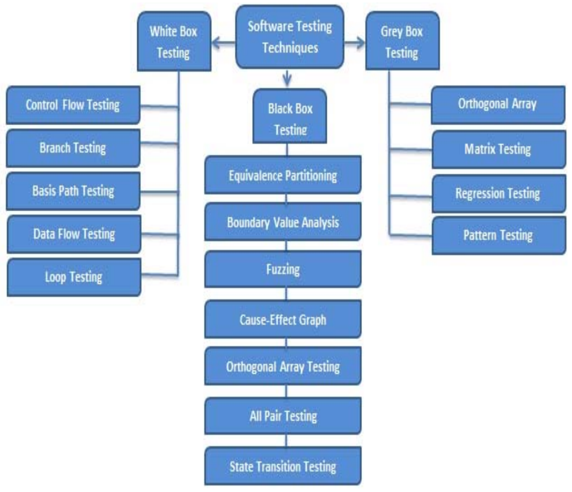

Figure 2: Software Testing Techniques [8].

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].

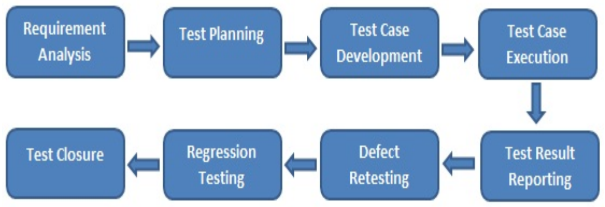

Figure 3: Software Testing Life Cycle [12]

During the first phase of the STLC, the review of the software requirements takes place by the Quality Assurance team in which they understand the core requirements according to which the test will be conducted. If in the case of any conflict arises, the team must coordinate with the development team to better understand and resolve the conflict. Test planning is the second and most important phase of the STLC, as this the step where all the testing strategy is defined. This phase deals with the preparation of the test plan, which will be the ultimate deliverable of this phase. Test Plan is a mandatory document biased towards the functional testing of the application, without which the testing process is not possible [11].

Test designing phase is the phase where the test case is developed, and the test planning activity is ceased. Appropriate Test cases are written by the QA team manually or in some cases, automated test cases are generated. Test case specifies a set of test inputs or data, execution conditions, and expected results. The specified set of test data should be chosen such that it produces expected result as well as intentionally erroneous data that will produce an error during the test. This is usually done to check what conditions the application ceases to perform [11].

Test Execution Phase is comprised of execution of the test cases based on the test plan that was produced before the execution phase. If the functionality passes the execution phase without any bug reportage, the test is said to be cleared or passed, and every failed test case will be associated with the found bug or error. The deliverable of such activity is defect or Bug report.

Test Reporting is the reporting of the generated results after the execution of the test cases which also involves bug reporting which then forwarded to the development team so that it can 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 Prioritisation does enhancement in the testing process by Combinational Criteria. The major methodology behind such test case prioritising is the conversion of the weblogs into the test suites relevant with the user session, and further writing it down into an XML format. The Algorithm used for this approach should be accurately prioritised by the coverage based on combinatorial test suites. Moreover, empirical studies should be carried out for analysing the effectiveness of the specific application and its relevant test suites [17]. A tool used in this regard is known as C-PUT which essentially formats the logs of the web applications into test suites which are formatted in XML; it is then used for the provision of the functionality for the prioritisation of these tests. There is an ongoing research about if these test suite prioritisation techniques can be used to enhance the fault detection ratio or not [18] [19]. The usage of genetic algorithms (GAs) for the purpose of automated test data generation for testing the application is yet another enhancement in the testing process, as previously the dynamic means of test data generation remained a big issue in the software testing process, so the usage of Genetic Algorithm based testing is an effective of the test data generation, it also capable of handling the data generation keeping in line with the complexity of program [20].

A. *Test Automation*

The major enhancement in the testing process leads the testing process towards the Test Automation, which is the use of particular software to carry out the testing process as well as it makes the comparison of actual results with the expected results. Test Automation technique is time effective, as it saves the time of manual testing which can be quite laborious.

In SDLC, Test Automation occurs during the implementation as well as the testing phase. Throughout the world, Test Automation is being practised instead of manual testing as it saves a great amount of time accomplishing the testing processes in shorter time span. Test automation has taken over the manual testing process by reducing its need as well as by exposing the amount of errors, shortfalls that cannot be acknowledged via the manual testing process.

Regression Testing being one of the major testing types requires much time when done manually. It typically tests whether the software or the application works properly after the fixation of any bugs or errors. Because sometimes after the error fixation, the code or application's error or bug ratio gets even higher. So, for the avoidance of the time taken for regression testing; a set of automated test suites is made to form a regression test suite for such purpose. Test Automation also helps in finding the problem at the much earlier stage, saving heaps of modification cost and energy at later stages [21].

The environment which caters a term typically knows the automation testing execution called Testing Framework. The testing framework is mainly responsible for executing the tests, as well as defining the format in which to express expectations and for the reporting of the results. The standout feature of Testing Framework that makes it widely applicable in various domains worldwide is its application independency [21].Testing Frameworks are of certain kinds, including Modular, Data Driven, Keyword Driven and Hybrid. The Modular Testing Framework is based on the principle of abstraction which involves the creation of different scripts for different modules of the software or application that is to be tested, thus abstracting each and every component from another level. This Modular division leads to the scalability as well as easier maintenance of the automated test suites. Also, once the functionality is available in the library, the creation of different driver scripts for different types of tests becomes easy and fast. The major con of such type of framework is to embed data within them, so when the modification or up gradation is requisite in the test data, the whole code of the test script needs to get modified. It was the major cause that served as a purpose for the invention of the Data Driven Testing Framework. In this type of Framework the test data and the expected results are ideally stored within different files, helping in the execution of single driver script being able to execute all the test cases with multiple sets of data. This kind of Framework reduces the number of test scripts as well as minimises the amount of code desired for the generation of test cases, gives more flexibility in fixation of errors or bugs.

Keyword driven testing Framework utilises self-explanatory keywords which are termed as Directives. Such type of framework is used to explain the actions that are expected to be performed by the software or application that is to be tested. This kind of testing is a basically extension of Data Driven Testing as the data as well as the directives are kept in separate data files. It encompasses all advantages of the data-driven testing framework. Also, reusability of the keywords is another major advantage. The ill factor of this kind of testing framework is that due to the usage of keywords, it adds complexity to the framework making test cases longer and more complex. Hence, to combine the strengths of all frameworks mitigating the ill factors being possessed by them. A hybrid approach is considered best for the usage as it is mainly a combination of all the three approaches and this combination integrates the advantages of all the testing frameworks, making it the most efficient one.

B. *Testing Frameworks in the Agile*

The agile lifecycle is another innovation in software testing as it encompasses short and speedy test cycles with frequently modifying requirements. Thus, the agile environment can encompass any testing framework, but due to the frequent iterations and rapid change in specified requirements, it maintenance of test automation suite becomes quite difficult. Though testing frameworks remains a bad fit for the agile environment because achieving maximum code and functionality coverage remains difficult in it.

C. *Test Driven Development (TDD)*

It is a technique that makes use of automated unit tests for the purpose of driving the design of software and forcing the decoupling process of the dependencies. With the usual testing process, tester often finds one or more defects or errors, but TDD gives a crystal clear measure of success when the test no longer fails, enhancing the confidence level about the system meeting its core specifications. Using the TDD approach a great amount of time can be save that might get wasted over the debugging process [21].

BDD (Behaviour Driven Development) is mainly an extension of Test-driven Development focusing on the behavioural aspects of the system rather than the implementation level aspects. Hence, giving a clear understanding of what exactly the system is supposed to do giving more efficiency to the testing process. Thus, BDD is mainly Test-driven Development incorporated with Acceptance testing, which typically refers to conducting a test to determine if the specified requirement of the product or software is met or not. If it is performed by the intended customer or user, then it is termed as User Acceptance Testing [22].

IV. TESTING METRICS

A. *Prioritization Metrics*

The usage of Test Metrics has prime significance as they can enormously enhance the effectiveness of the testing process. They serve as an important indicator of the efficiency and correctness and analysis of defined metrics. They can also help in the identification of the areas which require improvement along with subsequent action or step that needs to be taken to eliminate it. Test Metrics are not just a single step in STLC but acts as an umbrella for the constant improvement of the whole testing process itself [23] [24].Software Testing Metrics focus on the quality facets relevant to the process and product and are categorised into Process Quality Metrics, and Product Quality Metrics both of whims aim to provide enhancements in not only the testing process but also in the product quality.

However, there lays a critical issue faced by the existing testing process which is matching of the testing approach with the application being developed. Not every testing approach can be implemented in every application to be developed. For example testing of a network protocol software as compared with the testing of certain e-commerce application will be quite different with completely different test cases complexity, and that outlines the criticality of human involvement within the testing process and not just mere reliance on the existing test cases.Prioritisation Metrics include the length of the test based on some HTTP requests within a test case. Frequency based prioritisation enhances the testing process such that the test cases that encompasses most used pages are, selected for execution before those test cases that utilise less frequent ones [25][26].

B. *Process Quality Metrics*

A process is the most eminent part as it is capable of producing a quality outcome within the least time in the most cost effective manner. This is the ultimate reason that why organisations throughout the world have put their focus on the enhancement of the processes performance, and this exactly where the need for the metrics emerged, as it is required to gauge the process from various dimensions efficiently. Measuring Efficiency of the process is the key metric of process quality which encompasses certain measurements of factors like Test progress Curve which depicts the planned progress of the Testing Phase by the test plan [27][28].

The cost of Testing is the next major step of the metric both phase wise and component wise. The major objective of which is to help in identifying the parts that require intensive testing and cost that they will bear according to it. Average Defect Turnaround Time is another metric which depicts average verification time by the testing team for the verification of the defects. Average Defect Response Time is the metric that is an indicator of the operational efficiency. It is the measure of average time taken by the team for responding to the errors. Metrics for Process Effectiveness ensures that the resulted application or products will yield a high-quality output. Test coverage, Defect Removal Efficiency, Requirement Volatility Index, failed and executed test cases being major categories of it ensuring an overall enhanced Testing Process.
Also, the use of RTM (Requirement Traceability Matrix) can result in improved Testing Process, as it maps each test case with specifies requirement, making the testing more accurate [23] [24].

V. CONCLUSION AND FUTURE WORK

Testing is the most critical part of the Software Development Lifecycle, as it is something upon which the final delivery of the product is dependent. It is time consuming and an intensive process, therefore, enhanced techniques and innovative methodologies are requisite. This makes Automated Testing and other various Test Metrics implementation before and during the testing process. It can enhance the existing testing methods, both for time effectiveness as well as for efficient and reliable final product which not only meets the specified requirements but also provides with maximum operational efficiency.

The platform over which the software development and testing reside continues to evolve and remains exceedingly eminent. However, something so crucial and critical like Testing comes often quite late in the process of Software Development. There should be a maximum interaction between specification writers and Testers for better understanding and early review, which may fix ambiguity problems and consequently result in saving the cost of later fixing of the software. Testers after being clear about the specifications and requirements should hand over developers a certain lightweight test model, so they make sure the primary specification are met before handling the project for official testing. Use of simulation tools can immensely help the testers in creating the similar environment in which the product is destined to run, certain exception testing and methods for the exception handling can be best determined. While testing the product in the similar testing environment for which the product is meant for, and that can be easily done by integrating the simulation within the Testing process. Hence, the future work in relevance with the testing process will be much more technology dependent harnessing the simulation and automated testing model based approach, not only expediting the testing life cycle but also providing optimum bug prevention and efficient quality assurance.

REFERENCES

[1] 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/Testing05.pdf)

[23] Infosys, “Metric model”, white paper, 2012. Data retrieved from (http://www.infosys.com/engineering-services/white-papers/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/Vol55No3/10Vol55No3.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