International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 9 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v9i2.5455
***Article Received**: 09 December 2020 **Revised**: 22 December 2020 **Accepted**: 27 January 2021 **Publication**: 28 February 2021*

Software Testing Methodologies: A Information Review

Prof. Sapana Desai
Masters of Computer Applications Department
Sarvajanik College of Engineering and Technology
Surat, India
sapana.desai@scet.ac.in

Prof. Prashant Keswani
Masters of Computer Applications Department
Sarvajanik College of Engineering and Technology
Surat, India
prashant.keswanii@scet.ac.in

Abstract—In today's Complex era, the need for simplest software application has increased massively. The quality of such a handy application along with adequate testing is the biggest challenge one can face. Software Testing is an integral part of any software development which has to be followed right from the sapling phase of development. This paper focuses on testing methodologies which are used prior along with testing techniques for quality assurance and best of the quality.

Keywords-Quality, Software Testing, Software Development, Methodologies, Techniques, Optimization of Testing.

I. INTRODUCTION

Software testing is an activity to identify the discremination between system actual behaviour vs expected behaviour. It mainly focuses on number of verification & validation methodologies which helps to identify the functional as well as non functional deviation of system behaviour from its expected behaviour. Thereafter, the result shows the difference between what the system is performing and what is ideal behaviour of the system. Testing of any software includes identifying bugs, faults, errors or any missing functionalities which are part of the system being developed. The goal of the testing process is to deliver bug free products along with measurable quality. Software Testing can also be considered as a risk-based activity. Main objective of testing is to minimize the pool of input data being tested, identifying test cases, and visualizing risks involved. [1].

Figure 1 shows the crucial relationship between the cost of testing with errors which are being found. It clearly states that cost will rise substantially in either of functionality testing or non functionality testing. The decision of what to test and what to reduce is crucial as it may lead to a missing number of bugs which are important in terms of software testing aspect. The objective of testing is to perform an ideal amount of testing to cut down the extra testing efforts.[1]. As per figure 1, software testing is the main component of quality assurance. Testing importance is well understood by example of software being used in different ranges including flight control[2] to locate and launch rockets.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 9 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v9i2.5455
***Article Received**: 09 December 2020 **Revised**: 22 December 2020 **Accepted**: 27 January 2021 **Publication**: 28 February 2021*

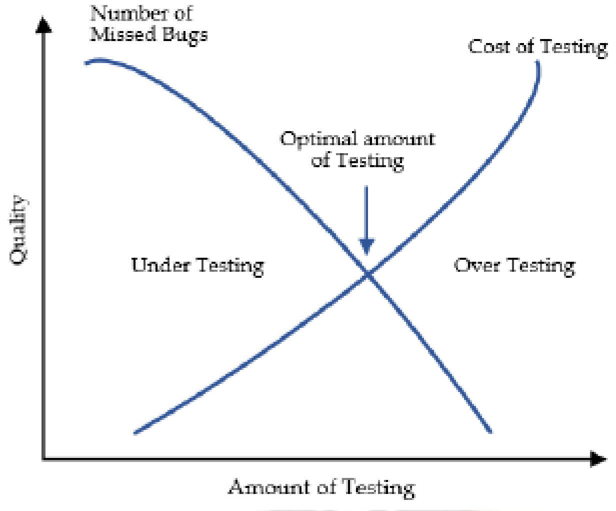

Figure 1. Software testing efforts (Courtesy [1])

Testing levels & steps vary from person to person & product to product.The three basic steps in the software testing are Unit testing, Integration testing and System testing which is being tested either by software developer or software tester. The testing mentioned above steps are inclusive in the Software Development Lifecycle (SDLC). The primary role of diving the entire software into modules is to make it testable. This is termed as Unit Testing. Integration testing is the second step in software testing. AFter Unit, all the modules are combined to test how well the modules are working once it is being combined. Lastly, the entire system is being tested to identify & locate errors if any. Apart from this, testing makes sures that integrated units do not affect or disturb the working of other modules. However, a large complex software application testing is highly time consuming as it involves more modules and deliverables. The difficult part is to test and check every scenario with all the possible combinations, which leads to an alarming need for an optimized software process.[6]. Testing cycle is mainly composed of several phases, from Test Planning to the analysis of Test Results. Test Plan is the first phase in testing which plans all the activities along with types of testing involved which are to be conducted in the whole testing process. The second phase of testing is development of phases where tests are being developed followed by testing life cycle which combines test cases & test scenarios. The outcome of this is termed as Test resorts which is being analysed in the last stage of testing termed as Test result analysis. In the last phase defect analysis is done by a developer who has developed the software or system ,this step can also be handled along with the client as it helps not only in the better understanding of which tests can be ignored but also it will help in identifying what to fix and how to simplify the product [7].

II. Current Testing Methodologies

Generating test cases is the primary process of testing life cycle. For test case generation, many techniques are used majorly falling in either black box or white box category. Effective software testing method is white box as it tests the functional part of the product along with internal building of any application. To design test cases, using white box testing, programming knowledge is primitive. White box testing many times refers to glass box or structural box testing. White box testing can be used in all levels of unit, integration & system testing. One other types of testing is security testing which deals with data protection & maintains intended working of any software or application. White box testing is capable of checking & testing internal logic of application, all the independent paths, logical decisions. Every declared loop is checked at boundaries and data structures which are internal are also examined. Moreover, white box testing is a complex testing as it requires programming knowledge for testing any particular module.[9][10]
Balck box testing involves checking the functionality of an application without worrying about internal working of the application. This technique can be applicable at all levels of testing in the software development life cycle. The main focus of black box testing is to run the test cases in a such a way that covers all the functional deviations. Black box test cases are focused & run with all the possible minimum , maximum and null values. Black box testing strategies are highly appreciated all around the globe and used widely for testing purposes.

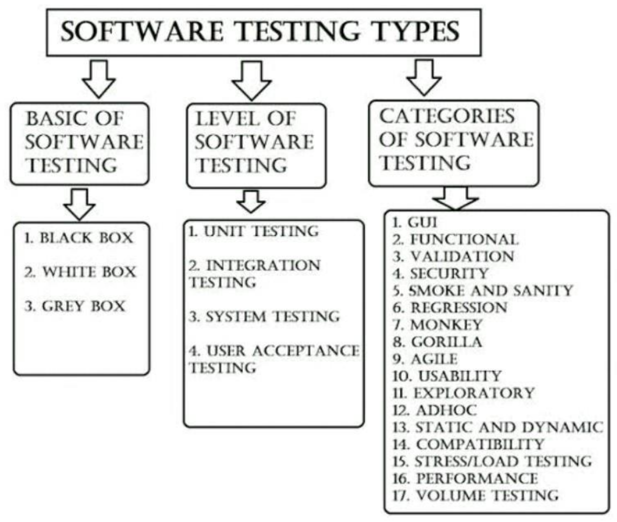

Figure 2:Software Testing Techniques [8]

The combination of White box testing and Black box testing is known as Grey Box Testing. Grey box testing has an advantage of both the testing techniques. This testing technique serves the need of knowing the internal structure of the application by the tester, and makes the testing more efficient. This idea is further extended in Figure 2 ,referenced from author J. Irena [8].

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 9 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v9i2.5455
***Article Received**: 09 December 2020 **Revised**: 22 December 2020 **Accepted**: 27 January 2021 **Publication**: 28 February 2021*

A. Software Testing Life Cycle (STLC)

In Figure 3, the STLC steps, stages and phases a software undergoes during the testing process has been shown. . Although, no fixed standard of STLC is followed. It varies from application to application[11].

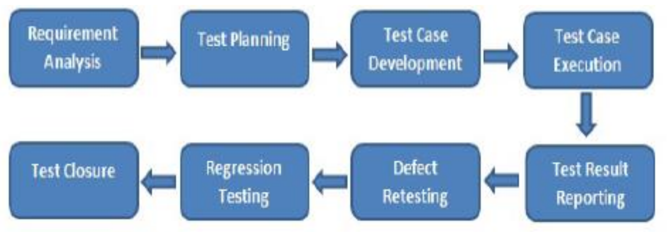

Figure 3: Software Testing Life Cycle [12]

In the first phase of STLC, the software requirements have been reviewed by the Quality Assurance team which helps in understanding the basic requirements which are being constructed. In case of conflict, the assurance team coordinates with the development team to understand the mismatch to resolve the conflict. Most important phase of STLC is Test Planning where the testing strategy is being defined. This phase works with plan preparation whose deliverable is known as Test Plan. Test plan is a required document which is being biased to the functional testing of application[11].

Test cases are being developed in the Test designing phase. Suitable test cases are being created by the QA team either manually or automated. Test case takes inputs as a set of data, conditions to execute along with expected results. The defined set of test data needs to be picked in such a way that it generates results which is an expected output along with incorrect data that should produce errors. Usually this is done to verify conditions that are performed by applications[11].

Test Execution phase incorporates test case execution which is based on a test plan that was generated prior to test execution phase. If the functionality successfully passes the execution phase without any bug report, the test is said to be passed or else it is failed. The outcome of this activity is Bug Report or Defect Report.

After the execution of test cases generated results are known as Test Report. If found any bug report which is being forwarded to the development team for correction & modification[11].

B. Software Release Life Cycle

Software testing life cycle is being followed by Software Release Life Cycle. This includes Alpha & Beta testing combinely known as User Acceptance Testing.

In Alpha Testing, application is being tested at developers end and can be done via white box technique or grey box technique. Alpha release is either at integration or system level testing which is being followed in black box testing. The alpha testing concludes with a feature which basically means no more features are being added or modified [13][14].

Beta testing is done after Alpha testing where testing is carried out at users side along with user and developer. Beta testing is also termed as Formal Acceptance Testing. The software or application is The application or software is released to a specific interested group of users for the testing purpose. Generally beta versions of the applications are given to authorised audiences for further suggestions and feedback for improvements. This beta version audiences are termed as Beta Testers and application under testing is known as Prototype of application. Hence, the final version of the software gets released after the Beta Testing [15] [16].

A. III. ENHANCEMENT IN TESTING PROCESSES

Combinational criteria is used in Test Suite priority. Aim behind prioritising test cases is to converter logs with their sessions followed by XML file format. This approach uses algorithms that are helpful in identifying test cases. In addition to this, empirical studies should be used to analyse effectiveness of particular applications[17]. C-PUT tool is used to format web application logs into test suites which are further developed to XML, which is then used for functional priority of tests.

An continuous research is going on how to prioritise techniques which can be used to find the fault detection ratio[18][19]. The GA (genetic algorithms) are used for automation of test cases & generating test reports. This dynamic approach is necessary for future advancements. Testing based on Genetic Algorithm, is useful for test data generation which is capable of managing data along with inlined focus on program complexity[20].

B. Test Automation

a) The main objective of test automation is to intensify the testing process, which matches the actual results with expected results and checks for mismatch. Automation of the testing process is less time consuming as compared to time invested in manual testing. In SDLC, test automation is being inlined right from the requirement gathering phase till testing phase. Automation has reduced the time required for manual testing. It has marginally cut down the efforts, shortcomings. One major type of testing is Regression testing, which will consume a lot of time if done manually. Regression testing works for checking of features which are being fixed after reported with bugs. This type of testing is necessary as many a time newly developed features have implication on previously made features and leads to bugs. For such a purpose, automation testing makes it faster and effective. Automation also helps in identifying issues in primary stages, which save hops of modification in energy & cost invested during development stages.

Testing framework sets the environment automation testing. This framework helps in execution of tests, identifying format

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 9 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v9i2.5455
***Article Received**: 09 December 2020 **Revised**: 22 December 2020 **Accepted**: 27 January 2021 **Publication**: 28 February 2021*

for expectations for result reports. The remarkable testing framework feature is, framework is application independent[21]. Test Framework includes Hybrid, Modular, Keyword & Data Driven. The modular testing framework works on the principle of abstraction which creates different scripts for different modules of software application. Division based on Modular makes it scalable and easily maintainable for test suites. Availability of any function in the library makes ease of driver or stub creation. The major disadvantage of this framework is that it has embedded data , modification in one data set leads to modification in the entire test suite. Data-driven is a test automation framework which stores test data in a table or spread spreadsheet format. Input data can be stored in a single or multiple data sources like XML, csv, xls, xlsx and databases in a data driven test automation framework. 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, giving more flexibility in fixation of errors or bugs.

Directives are keyword driven framework utilities. This type of framework can be used to elaborate expectations functions by software applications. This is an extended version of Data driven testing where directives are kept in separate files. It encapsulated all the functionalities of data driven testing. Main advantage is reusability. The disadvantage of data driven testing is complexity that makes test cases lengthy and complex. So, a hybrid test framework is a better approach which combines the advantages of the rest of the framework.

a. B. *Testing Frameworks in the Agile*

The lifecycle which is agile is a revolution in software testing as it adds speed to test cycles by allowing modifications on iteration basis. At the end , an agile environment which allows rapid changes in requirements followed by a testing framework makes maintenance of automated test cases little hasty. In Agile, testing frameworks doesn't fit so well, as it adds difficulties in achieving maximum code coverage & Functionality coverage.

b. *C. Test Driven Development (TDD)*

This technique uses automated unit tests which are used for the delivering design of software & minimized the dependencies & coupling. When testers test with conventional methods to find bugs, there are chances to find one or many, but with TDD most of the uncovered errors are answered which adds up system stability. This is a time saving approach which was earlier being wasted in debugging[21].

Behaviour Driven Development (BDD) is an enhancement of Test-Driven Development which focuses on the behaviour pattern of system over implementation aspects. So, it gives you a better vision of what system should do for better efficiency. Thus, BDD follows Test Driven Development which includes Acceptance Testing. User Acceptance Testing is also a synonym for this type of testing is performed by a special group of users or customers[22].

This lines up a question of matching the testing approach with application under development. Not all testing methodologies are being used or suitable for all the applications being developed. For instance, testing e-commerce applications is different with testing any network measuring application. Prioritization metrics adds the tests based on HTTP requests within the 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].

C. IV. TESTING METRICS

B. *Prioritization Metrics*

Test Metrics is important as it enhances the effectiveness of the testing process. It indicates the correctness & efficiency of metrics. It also highlights the areas where improvisation is needed. Test Metrics serves as an umbrella activity which suggests constant improvements throughout the STLC[23] [24]. Testing metrics pins the quality areas which are primary for process improvement and are bifurcated in Process Quality Metrics & Product Quality.

This lines up a question of matching the testing approach with application under development. Not all testing methodologies are being used or suitable for all the applications being developed. For instance, testing e-commerce applications is different with testing any network measuring application. Prioritization metrics adds the tests based on HTTP requests within the 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].

C. *Process Quality Metrics*

The most important part is the process which has a capability to produce an outcome which has a high quality with a cost effective manner. This is the prime reason for organizations to have a focal point on the process improvement where metrics turn up. Process efficiency is used to measure process quality which includes measuring the test process curve which shows progress of Testing Phase by test plan[27][28].

In testing, cost is the main concern in phase & component wise. The objective behind testing metrics is to recognize parts which are important for intensive testing. Average Defect Response time metric is one more metric which shows average verification time by the testing team. Average Defect Response Time indicates operational efficiency. It measures average time taken by the testing team for identifying & responding time. Defect Removal Efficiency, Requirement Volatility Index,Test coverage failed and executed test cases being major categories of it ensuring an overall enhanced Testing Process.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 9 Issue: 2
DOI: https://doi.org/10.17762/ijritcc.v9i2.5455
***Article Received**: 09 December 2020 **Revised**: 22 December 2020 **Accepted**: 27 January 2021 **Publication**: 28 February 2021*

Requirement Traceability Matrix is useful for mapping every test case with its specific requirement which enhances the testing results and helps in achieving the better quality. [23][24].

D. V. CONCLUSION

In the software development lifecycle, software testing is crucial as it defines the final delivery status of the product.
If it's delayed at the SDLC, it becomes time consuming and more costly in terms of bugs. Testing methodologies can be manual or automated depending on the nature & work of the software application being developed. Test Metrics can be added and used throughout the testing process. It accelerates the testing process in terms of time and final product delivery.
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.

ACKNOWLEDGMENT

Being a faculty of Computer applications and pursuing research work, through this testing methodologies comparisons & need for the testing being used right from the first phase of Software Development Life Cycle. Testing is the future and making it dynamic and less time consuming keeping quality in mind is a bigger challenge and which will be our next goal to address.

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 Costbased 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/Vol55No3/10Vol55No3.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