International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 9 Issue: 8
DOI: https://doi.org/10.17762/ijritcc.v9i8.5480
***Article Received**: 03 July 2021 **Revised**: 20 July 2021 **Accepted**: 12 August 2021 **Publication**: 31 August 2021*

Software Testing: Data Kart and Integrated Pipeline Approach

Md Rehan Faisal
Department of Computer Science, SKITM, MDU Haryana
rehan2502@gmail.com

Abstract--With the evolution of Digital Era and growing demand of automation from manual effort, demand of Software and automated application increases and it will keep increasing day by day. With increase in application and software the quality of product and quality assurance become a vital role in any software life cycle. To maintain the quality and final release cycle of software, Software Testing become a key challenges one can face. Software testing play important role from the beginning till the release of application. Current paper focus on tradition testing phase along with enhanced data driven and pipeline integrated techniques to maintain best quality of software.

Keywords—Data Driven Approach, Pipeline, Testing Methodologies, Automation Testing, Software Testing, Testing Frameworks, Test Driven Development, Test optimization, Quality Metrics.

I. INTRODUCTION

Software testing is one of the important and crucial steps in software development life cycle. It provides us the quality of software as well as insight view of the application. Software testing provides us discrimination between system actual behaviour vs expected behaviour. The goal of software testing process is to provide high quality product which is bug free and also improve quality by identifying the defects. Evaluate work products which may be requirement documents, Design, all the Customer's requirement, find defect and prevent defect in software system. Testers test the product, create requirement traceability matrix for mapping of all the test cases to requirement, create test cases and ensure that all functionality all verified for every executed test case. Before system go live, system need to test on all parameter so that it will run smoothly and bug free for better user experience and user ease. So software testing phase is important area for the testing point of view as well as research point of view.

II. CURRENT TESTING PROCESS

In current testing methodologies creating test case and Test set is first steps. For test case generation many different approaches are used and it also depend on the types of category either black box testing or white box testing. If we look into other perspective for testing software it may also be classified into Functional testing, Non-Functional Testing and Maintenance Testing. If tester have to test some functionality then will go for functional testing which comprises of Unit testing, Integration testing, Smoke, UAT, Localization and Globalization. If tester need to test other than functionality then will opt for Non-Functional testing which includes Performance Testing, Endurance Testing, Load Testing, Volume Testing, Scalability Testing and Usability.

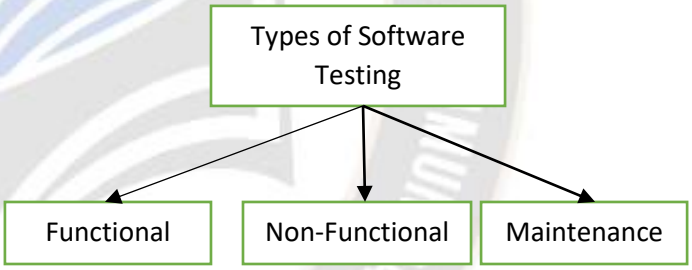

Figure 1: Software Testing Types on Functionality Basis

A. Manual Software Testing

Manual Testing is the approach in which software is tested manually, there is no use of automated tools in this process. Whole process depend on tester who individually identify the test area and results of testing. Everything is carried out in manual way from error to the report of testing.

B. Automated Software Testing

Automation Testing is the approach in which software is not tested manually, it is also called as Test Automation and the tester who perform this activity is called as Test Automation Engineer. During this testing process Test Automation Engineer writes automated scripts to test the product and find out any error in system. This process is quick, fast and automated as compared to manual process.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 9 Issue: 8
DOI: https://doi.org/10.17762/ijritcc.v9i8.5480
***Article Received**: 03 July 2021 **Revised**: 20 July 2021 **Accepted**: 12 August 2021 **Publication**: 31 August 2021*

III. METHODS OF SOFTWARE TESTING

In Software Testing Field there are lot of methods and Approach followed by company and organization to carry out this testing. Some of them are below:

A. Black-Box Testing Approach

Black-Box Testing is the approach in which software is tested without knowing the internal working of application. Tester is unaware of what logic and algorithm written inside the application to perform the desired task. During the testing process tester is unware of internal structure and codes, tester just provide input value corresponding to the filed parameter and validate the result as a parameter without going into deep of hoe system calculate and evaluate all these things.

B. White-Box Testing

In this Testing approach whole internal structure of code is involved, where tester need to know all internal code and logic of the system to perform testing activity. Testing is done on internal structures or working of application. White box tester need programming skills because in this approach internal perspective of system as well as programming skills are used to design test case and to carry out the testing activity.

C. Grey-Box Testing

Grey-Box Testing is a software testing technique which is a combination of White-Box and Black-Box Testing approach. In this approach tester need to know all about software working as well as internal structure like coding, Algorithm and Steps involved in processing.

Black-Box focus only on input and output value without knowing the internal structure and coding, due to this tester not able to prepare a better test case. So to overcome this activity Grey-Box testing is best approach for testing.

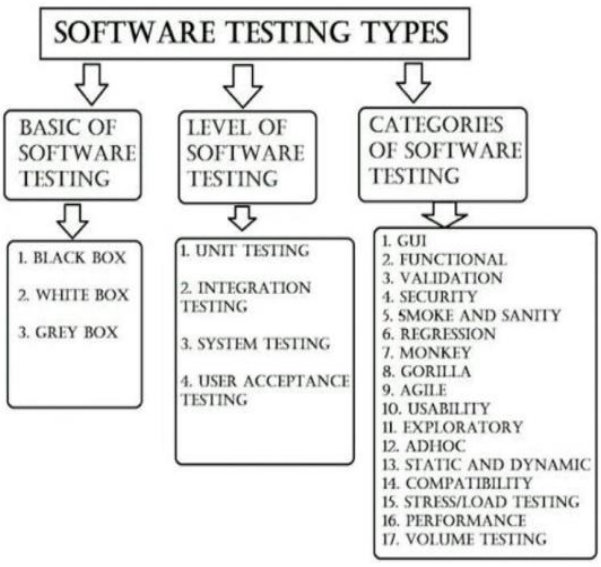

Figure 2: Software Testing Types

IV. ADVANCE TESTING PROCESS

This is advanced or enhancement in testing process, which is very much handy and automated as compared to existing steps. Advance steps not only save money, time and resources but also offer user to schedule and let machine process all steps. Enhanced steps consists of Automating screens, Data form UAT server to process and validate, automated test pipeline with dev pipeline and on demand advance reporting.

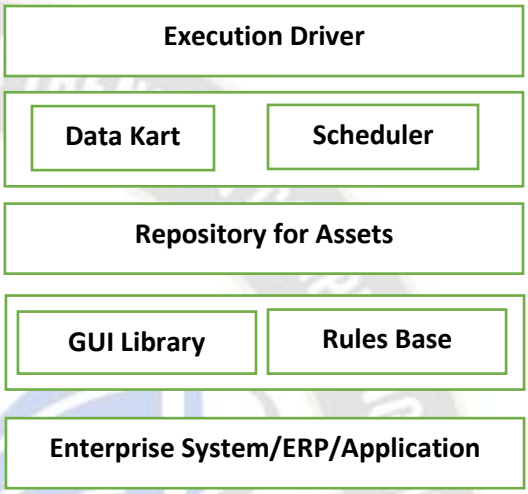

Figure 3: Advance Testing Architecture

A. Automating Screens for Test

The main objective of this is to find relational matrix between the expected value and actual value and then checks the mismatch. Automation of screen using test automation is easy and less time consuming as compared to the manual testing process. In software development life cycle test automation is start from the Requirement phase to till the testing phase. Test automation involve form capturing the screen and then adding these screen to make a test case. Once the test case is ready we combine these test case into test set which is generally a group of test set.

Regression testing consume lot of time, whenever new bug is fixed regression testing is required, which works for checking of features which are being fixed after bugs. This type of testing is recursive in nature and required whenever a new bug is fixed, so creating a test automation with flow and set of flows and running after each new released will save lot of time and manual effort. Automation also helps in identifying issues in early stages, which save lot of resources & cost invested during early development stages itself. There are different testing framework which is used to set the environment automation testing. This framework helps in execution of tests and test sets, identifying exact format for expectations for result reports.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 9 Issue: 8
DOI: https://doi.org/10.17762/ijritcc.v9i8.5480
***Article Received**: 03 July 2021 **Revised**: 20 July 2021 **Accepted**: 12 August 2021 **Publication**: 31 August 2021*

B. Data Kart

Data Kart is concept for providing real test data of their field type when a test starts running. Data Kart table is associated with the test when a test is created, once it is associated with their column and data type it fetches real time test data from underlying cache table and execute the test to find a relational matrix between expected and actual results. Data Kart provide concept of supplying data to test automation flow, main advantages of this Data Kart is that it integrate with Enterprise UAT Data Base and work with actual test data. User have option to change the corresponding data during execution by applying a constant label in test on that particular filed of screen. Associated mapping is done between parameter and underlying Data Kart cache table column.

Data Kart also provide facility to analyze the data first by applying different level of filter to identify the potential of data that will be going to use in test execution. Applying sort filter will provide how unique data exist in Data Kart and what will be possibility factor on this data when a test will be execute. Key Club is concept of clubbing two or more table data on the basis of primary and foreign key relation in table.

C. Integrated Test Pipelines

Now a days most development company follow continuous integration and continuous delivery/continuous deployment process for code merge and build deployment process. It is a modern software development practice in which incremental code changes are made frequently and reliably. Automated build-and-test steps triggered by CI ensure that code changes being merged into the repository are reliable. The code is then delivered quickly and seamlessly as a part of the CD process. In the software world, the CI/CD pipeline refers to the automation that enables incremental code changes from developers' desktops to be delivered quickly and reliably to production.

Along with this CI/CD pipeline a test pipeline is also added whenever developer pushes code it trigger the test pipeline and generate a test report corresponding to the bug fixed for details analysis. Adding test pipeline helps lot and also save manual effort for Software testing phase in SDLC. As soon as test pipeline executes corresponding to the requirement and test the targeted module it will generates log and report file with all details analysis which will be further used as expected and actual result analysis of targeted module.

D. On Demand Reporting

Report help in analyzing data, logs and results in more precise and accurate way. Report help user, tester and admin to look into the final status of test execution, input output value and expected and actual result set of test without going deep into the system and without monitoring the ongoing execution. When a test and test set is executed a details report is generated corresponding to that execution ID with all steps in details. Expected and Actual value can be identified from report itself by analyzing the report. Report also provide insight view of which screen open, what data was entered inside different UI fields, which button was clicked on UI and also attach screen shot of that executed steps. User have option to schedule a report on their email, just after execution end user will get a details report dashboard inside inbox along with all steps and business process details in pdf format. Report also populate suggested filed from machine learning approach for those steps which get failed or skipped during execution process.

E. Scheduler Based Recursive Testing

Scheduler plays an important role for such activity which is recursive in nature and need scheduling activity. Scheduling is the action of assigning resources to perform some task by system itself in automated manner. Scheduler can be applied to any resources such as process, network and any kind of custom activity. Scheduler based recursive testing is required where we need test to be executed on recursive interval. For such test we need to schedule the test using scheduler and it will automatically run the test on schedule time, generate the log and report along with all pass and failed details. Scheduler helps in executing recursive test without any resource hence it will save manpower and cost as well for the organization.

V. CONCLUSION

In any application development life cycle testing is crucial as it decide the final end product. Software Testing provides in-depth view about the quality of the product or application which is used by the many end users. Software testing also gives independent view of software in the different aspect to understand the risk of software implementation and potential attack. If any delay in development phase it delay all the process and resources of company. Testing process may vary organisation to organisation and type of software developed, it can be automatic or manual. To accelerate the testing process in terms of time and final product delivery test matrices can be added from start till end process. Hence the future point to more automated testing model based approach and real data driven based testing.

REFRENCES

[1] S. Amland, "Risk-based testing:" Journal of Systems and Software, vol. 53, no. 3, pp. 287–295, Sep. 2000.
[2] Redmill and Felix, "Theory and Practice of Risk-based Testing", Software Testing, Verification and Reliability, Vol. 15, No. 1, March 2005.
[3] B. Agarwal et al., "Software engineering and testing". Jones & Bartlett Learning, 2010.
[4] K. Bogdan. "Automated software test data generation". Software Engineering, IEEE Transactions on 16.8 (1990): 870-879.
[5] Jacobson et al. The unified software development process.Vol. 1. Reading: Addison-Wesley, 1999.

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 9 Issue: 8
DOI: https://doi.org/10.17762/ijritcc.v9i8.5480
***Article Received**: 03 July 2021 **Revised**: 20 July 2021 **Accepted**: 12 August 2021 **Publication**: 31 August 2021*

[6] Everett et al., "Software testing: testing across the entire software development life cycle". John Wiley & Sons, 2007.
[7] B. Boehm, "Some Future Trends and Implications for Systems and Software Engineering Processes", 2005, pp.1- 11.
[8] R. Bryce, "Test Suite Prioritisation and Reduction by Combinational based Criteria", IEEE Computer Society", 2014, pp.21-22.
[9] 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)
[10] 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 missinglink myths," Software, IEEE, vol. 19, 2002, pp. 15-
[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 TechniquesTechnology 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 Usersession based Testing of Web Applications", IEEE Transactions on Software Engineering, Vol. 33, No. 10, 2007, pp. 643 - 658