A Comparative Analysis On Software Testing Tools And Strategies

Pramod Mathew Jacob, Priyadarsini S, Renju Rachel Varghese, Sumisha Samuel, Prasanna Mani

Abstract: Current software industry's prime focus is on developing quality application programs. The basic business management principle 'Customer acceptability is directly proportional to the quality of the product' is admissible for software products too. Thus, testing phase has a vital role in improving customer satisfaction of a software application. The various research analytics claim that nearly 30% effort of entire software development is used for performing testing activities. Every software firm or application developers follow a typical custom set of testing strategies and uses some standard testing tools for quality assurance. The project manager has to decide the testing strategy between manual and automated testing. In automated testing, there are many tools available with different capabilities and performance characteristics. This review analyzes the performance metrics of various testing tools and testing strategies used for enriching the quality of the application being developed. The review result may guide the project manager to make the trade-off decisions for choosing the appropriate testing tools and testing strategies applicable for their project domain.

Index Terms: Software testing, Test script, Test cases, Automated testing tool, Manual testing, System testing.

1 INTRODUCTION

Software testing Software testing [1, 2] is a quality control activity which involves defect detection and correction. Testing can be performed at various stages of software development process depending upon the methodology and tools being used and usually begins after the requirement confirmation phase. The initial phase is at unit level where it mainly focuses on coding. After coding all software units, we perform Integration testing for finding out the bugs in the software application. The ultimate purpose of software testing is to prevent software from failure, ensuring the quality and satisfying the stakeholders. Software testing can also be mentioned as the process of performing verification and validation to a software or an application that meets the business oriented and technical-oriented functional requirements that guided in its design phase and development. Validating and Verifying (V&V) [3] is the process of ensuring that a software meets the requirements mentioned in Software Requirement Specification (SRS) document and that it fulfils its intended functionality. It can be considered as a methodology to ensure software quality. The terms can be defined as follows:

Verification: It is the process of ensuring whether the software product is built in the right manner.

Validation: It is the process of ensuring whether the developed software product is as expected. Testing usually consist of three levels called Unit testing, Integration testing and System testing each of them has many sub levels of testing strategies. These testing can be performed either using human resources or using automation tools.

- *Pramod Mathew Jacob is currently working as Assistant Professor in Providence College of Engineering Chengannur, Kerala, India, E-mail: pramod3mj@gmail.com*
- *Priyadarsini S is currently working as Assistant Professor in Providence College of Engineering Chengannur, Kerala, India.*
- *Renju Rachel Varghese is currently working as Assistant Professor in Providence College of Engineering Chengannur, Kerala, India.*
- *Sumisha Samuel is currently working as Assistant Professor in Providence College of Engineering Chengannur, Kerala, India.*
- *Dr. Prasanna Mani is currently working as Associate Professor in Vellore Institute of Technology, Vellore, Tamil Nadu, India.*

There is a wide set of testing automation tools available. Our work evaluates the performance and characteristics of most widely used test automation tools. This review work also focuses on analyzing the efficiency and applicability of various testing strategies that can be followed in a software industry.

2 RELATED WORKS

K. M. Mustafa et al has classified the software testing tools based on software testing methods [4]. They have evaluated nearly 135 testing tools and categorized them over three types of software products (web application, application software, network protocol). Their analysis also points out which testing method has limited automation tools. T. E. J. Vos, et al has proposed a framework for evaluating software testing tools and techniques [5] which is successfully used for some case studies in EvoTest and FITTEST. But this framework fails to evaluate the performance constraints of automated tools like test script generation time, test script evaluation time and much more. James B. Michael et al derived object-oriented metrics for measuring the effectiveness of the software testing tools and is validated using three testbeds [6]. The proposed computational metrics can be used in evaluating the effectiveness, quality, accuracy, performance and much more for a tool. Insha Altaf, J. A. Dar et al has performed a survey on Selenium tool in software testing [7]. They have discussed about the basic features and functionalities of the Selenium testing tool along with mentioning some of its application domains. R. Angmo et al has evaluated the performance of web-based automation testing tools [8]. They have analyzed web testing tools: Selenium and Watir. Their evaluation results claim that Selenium web driver is much better than Watir web driver, if suitable Selenium browser plugins are added. In most of the related works, the authors mainly focusing on a particular framework or testing aspects. Most of the tools used are outdated now. So, our review focuses on the most widely used testing tools like Selenium and HPE UFT against some basic standard performance metrics.

3 TERMINOLOGY

In order to analyze various testing techniques, it is better to recollect the basic terminologies used in software testing:

Test case: A test case is a set of test data that is to be inputted, along with expected results and resultant conditions, designed for a particular testing module in order to ensure system functionalities [9, 10]. Consider the example test cases for the ATM Personal Identification Number (PIN) field validation in a web page. Assume that the PIN should be a four-digit integer number. Then the test case should be as follows:

Valid test case: PIN = 1234 // 4 –digit integer
Invalid Test cases:
PIN = 123 // non 4 –digit integer
PIN = abcd // not an integer
PIN = 12a4 // combination of character and integer
PIN = Blank // No Input data

In this case test cases can be defined using the Equivalence class partitioning method [11]. There is one valid and two invalid classes.
Valid class: (1000 ≤ PIN ≤ 9999) // 4-digit numbers.
Invalid classes are: (PIN < 1000) and (PIN > 9999)

Test suite: It is mathematically a set which consist of all the test cases as set elements. A software can have infinite number of test cases. A good test suite should have minimal test cases which covers most errors. Test suite to validate the ATM PIN field is:
{2000, 999, 10000, 12a4, abcd}

Error: Error is the degree of mismatch from actual result and expected result. It represents mistake made by code developers [12]. Though error and bug sounds synonyms, they are not exactly the same. Error is the mistake found by tester. When developer accepts this mistake, then it is called as a bug.

Fault: Fault is an incorrect code statement, function or data interpretation in a computer program which makes the program or software system to generate an unexpected result. Fault leads to error.

Bug: Bug is a fault in the code block which generates an unintended result. It is normally the mistake accepted by the developer.

Failure: It is the inability of a software system to perform its expected functional and non-functional requirements. Execution of a fault leads to failure.

Defect: A defect is a mistake committed by programmer in coding or logic that causes a program to generate incorrect or deviated results. Normally a defect is detected when a failure occurs.

Test script: It can be defined as a collection of instructions applied on the system under testing, to ensure that the system performs its specified functionalities and behaves as expected.

4 SOFTWARE TESTING STRATEGIES

Software industry usually follows two testing approaches: Manual testing and Automated testing. In manual testing, testers evaluate the software manually for the faults. Here the tester plays the role of an end user and evaluates all the possible features and functionalities of the developed software to ensure its behaviour and quality. The tester manually prepares test plan and suitable test cases which is tested over the application to verify the behaviour of Graphical User Interface (GUI), functional and non – functional requirements as illustrated in Figure 1. But performing manual testing requires a large amount of human intervention and the presence of an experienced - skilled person to design an appropriate test suite.

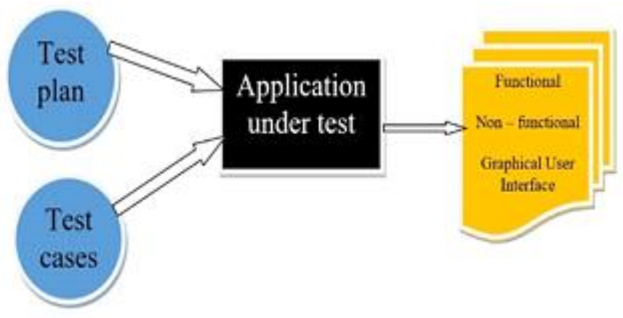


Figure 1: *Process involved in manual testing*

In automated testing, execution of test cases is supported by automation tools. Automated testing is quite beneficial in case of large projects. Here automation tools perform tests that repeat predefined actions, matches the developed program's probable and real results [18].

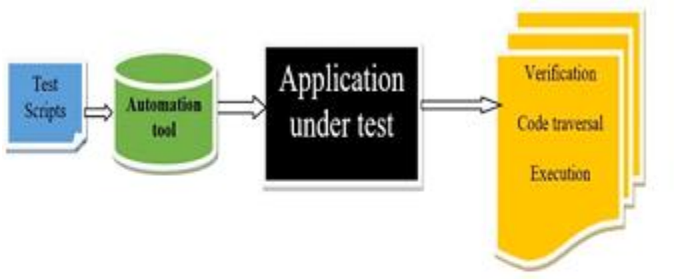


Figure 2*: Processes involved in automated testing*

Test scripts are used by automated tools to perform software testing. The automated tool traverses through the entire code blocks for bugs, verify each system behavior by executing test cases. If the expected behavior matches with the test results, the automation tool confirms the quality of the product. The Figure 2 shows the process involved in automation testing. It is not practically feasible to automate all test cases which involves some constraints regarding aesthetics and appearance. There are pros and cons for both testing strategies. Choosing which one to use among the two depends upon factors like size of the project, time, availability of resources and much more. The performance analytics [19, 20] of automated and manual testing are plotted in Figure 3.

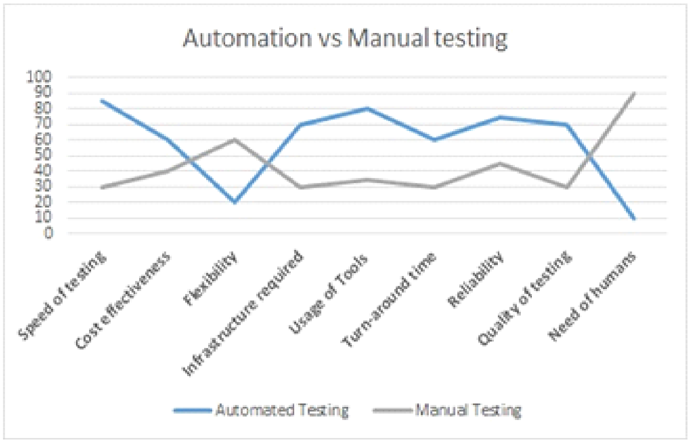


Figure 3: *Performance analytics of automated versus manual testing*

If the project consists of too many test modules, then it is better to prefer automated testing [27,28,29] than manual testing. Because it is not feasible to derive all the combinations of test cases for a project with too many coupled modules. If a project got more time available for testing, then do it manually, where the possibility of finding real time errors is comparatively high than that of automated testing tools. The prime constraint in performing manual testing is the need of experienced testers. The manual testing [30] efficiency depends on qualitative factors like experience and knowledge level[31,32] of tester, nature of the software module under test, category of the software etc. The analytical comparison of both testing strategies is shown in Table 1. From the plotted graph, it is clear that automated testing performs far better than that of manual testing in most evaluation parameters. It is recommended to use automated software testing tools in certain testing scenarios. As the software industry[33] is too competitive, there is a wide room for software quality and reliability[34] in the scenario. Deadlines and product release dates are more essential in this competitive market. In this scenario an intelligent software company will choose some good testing tools which may increase the efficiency of testing as well as to get the product deadlines met.

TABLE 1*. AUTOMATED TESTING VS MANUAL TESTING*

Criteria	Automated testing	Manual testing
Speed of testing	Faster	Slower
Cost effectiveness	More cost effective	Slightly less
Flexibility	Not flexible	Too flexible
Reusability	Easily reusable	Not practically feasible
Infrastructure required	Comparatively high	Less
Need of Training	Highly recommended	Not required
Usage of Tools	Too high	Slightly less
Turn-around time	High	Less
Reliability	Comparatively more	Less reliable
Need of programming	Highly needed	Not needed
Quality of testing	High	Low
Need of human resources	Very few	Huge
Mode of performing tests	Concurrently in different systems	Sequentially
Testing GUI & Aesthetics	Not possible	Can be performed
Performing Regression tests	Can be easily performed	Very difficult and boring task
Execution of Build Verification Test (BVT)	Easier to perform	Too difficult to execute BVT
Applicable projects	Complicated and large projects	Small projects with limited functionalities

5 SOFTWARE TESTING TOOLS PERFORMANCE EVALUATION

There are lots of software testing tools available in the market. The most successful ones include Selenium, HPE Unified Functional Testing (UFT), IBM Rational functional tester, Silk Test, LoadRunner, WinRunner and Test complete. The various software testing tools can be classified based on their functional features and their role in testing process as shown in Table 3. There is still research undergoing in developing testing tools to address other aspects of testing and validation. During the STLC automation phase, the project manager should choose an automation tool which is applicable to the project being tested. The choice depends upon factors like tool functionalities and features, testing team, budget of the project being tested, performance aspects of the tools and much more. There are testing tools which is customized for a particular programming language like JUnit for Java application and PyUnit for Python. Some tools like web application tool, desktop application tool and mobile application tool are domain specific. Our comparative analysis mainly evaluates two widely accepted software testing tools: Selenium and HPE Unified Functional Testing (UFT). HPE Unified Functional Testing (UFT) [21] which was earlier known as HP Quick Test Professional (QTP) provides features for test automation and functional testing for various software and environments. It is widely used for enterprise quality assurance. It provides a GUI and features for keyword and scripting interfaces. It uses VBScript to write the test scripts. HP UFT is a single console for verifying the interface, database and service layers of a software or application. Selenium [22] is an open source, portable testing framework mainly focusing on web applications. Selenium has a record-playback tool for authorizing tests without learning to develop test scripts. The prime components of Selenium framework include Selenium IDE, Selenese, Selenium Remote control, Selenium Client API, Selenium web driver and Selenium Grid. For our evaluation we selected Windows (Ver. 10) as operating system platform. We have installed the Selenium IDE [23] and web driver add-ons in Mozilla Firefox browser. HPE UFT [24] desktop application is installed with required updates. We evaluated the performance of two testing tools by testing the login pages of our VIT University [25], Rediffmail, Facebook and Gmail. These login pages read username, password and a CAPTCHA from user and redirects them to homepage after successful validation. The test cases matrix used for validating a login page is shown in Table 4.

TABLE 2*. TEST CASE MATRIX FOR LOGIN PAGE VALIDATION*

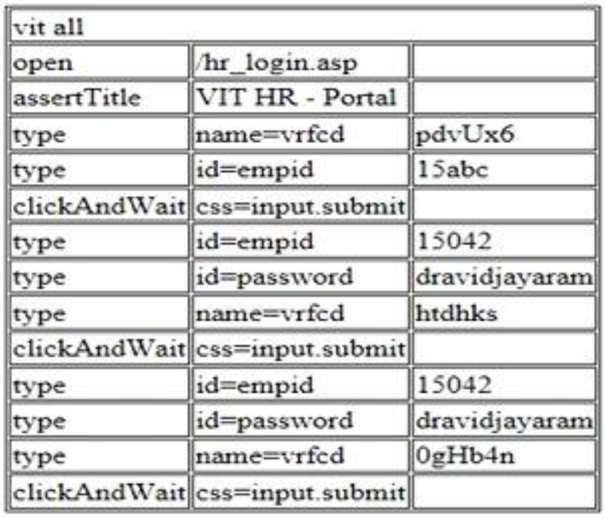
Test case ID	Username	Password	CAPTC-HA	Expected outcome message
1	Valid	Valid	Valid	Login success
2	Invalid	Valid	Valid	Incorrect Emp.

				Code
3	Valid	Invalid	Valid	Incorrect password
4	Valid	Valid	Invalid	Incorrect Captcha

We have evaluated the tools using the following performance metrics [26].

1. Mean Script Creation Time (MSCT): It can be defined as the mean time to create test scripts. It is the average time taken to generate one test script.
2. Mean Script Execution Time (MSET): It can be defined as the mean time to execute test scripts. It is the average time taken to execute a single test script.
3. Tool Learning Time (TLT): It is the average time required for a single naïve user to learn the software tool to use. Of course, learning level is unquantifiable. It can be redefined as the average time taken for training naïve personnel to familiar with the software tool.
4. CPU Utilization rate: Percentage of CPU capacity being used for running the software tool. 5. Tool Reliability (TR): It is the probability of failure-free operation of the testing tool for a specific time period in a particular environment. Its value usually ranges between 0 and 1.

We evaluated the test cases in Table 2 using the Record-Play back feature of Selenium. The test script generated by Selenium is shown in Figure. 5.



vit all		
open	/hr_login.asp	
assertTitle	VIT HR - Portal	
type	name=vrfcd	pdvUx6
type	id=empid	15abc
clickAndWait	css=input.submit	
type	id=empid	15042
type	id=password	dravidjayaram
type	name=vrfcd	htdhks
clickAndWait	css=input.submit	
type	id=empid	15042
type	id=password	dravidjayaram
type	name=vrfcd	0gHb4n
clickAndWait	css=input.submit	

***Figure 4**: Test data output*

After saving the suitable test cases, we monitored the time for creating each scripts. We have summed up all these to estimate the Mean Script Creation Time (MSCT). It is found that the MSCT depends on the number of actions or activities performed during the recording time. Script creation time is directly proportional to the count of activities performed during recording. The phase test execution takes place afterwards, where we loaded the test suite script. We have noted the initial time and final time for test execution to estimate the Mean Script Execution Time (MSET). Then we repeated the same test suite and testing procedures to evaluate HPE UFT. But in HPE UFT it is not possible to directly write the script to verify the CAPTCHAs [27] without having access to the CAPTCHA database which contain all CAPTCHAS values present in the system. If we are permitted to access CAPTCHA database, then it is possible to store the values for respective images in an array and check the image. If image match, use that value. This is not a perfect solution though, access to CAPTCHA database is not granted always. So, in our evaluation we validated only the username and password fields.

6 RESULTS

We have analyzed the above-mentioned performance parameters of Selenium and HPE UFT tool. The results that we obtained is plotted below in Figure 5.

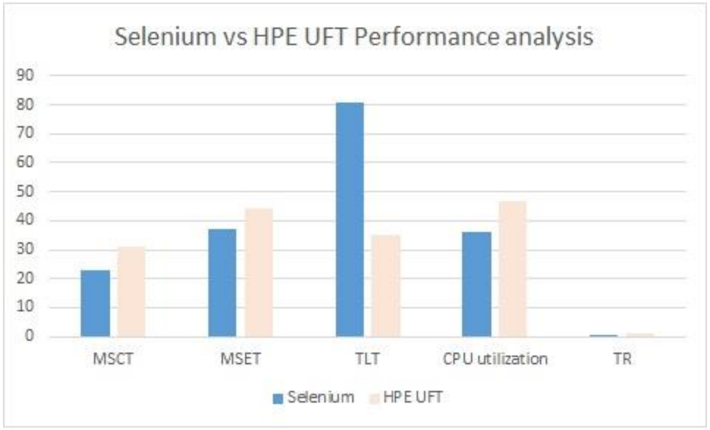


***Figure 5:** Performance evaluation of Selenium & HPE UFT*

The results that we mentioned is an approximate average value, though these performance parameters depends upon various qualitative parameters like tester's knowledge level and experience, software and hardware support, level of testing, complexity of the project being tested, language used for scripting and much more. But our results can be used as a reference model for software testers to choose the testing tool which is appropriate for their domain. Though Selenium uses Record and playback feature, Test script generation time can be consumed. We have noted the time for generating scripts for multiple test cases and then estimated the average number of scripts that can be generated in an hour. Mean script execution time is also estimated in the same manner by noting the time taken to execute each test cases. MSET also depends upon the complexity of the test module. For estimating tool learning time, we choose around 10 students with basic programming knowledge. We trained them till they are able to generate and execute test scripts. The mean time taken for each individual is measured to approximate up the TLT value. TLT is not purely quantitative though it depends upon the skill of the persons being trained. CPU utilization is an average over a period of time. At any point in time, if the CPU is either busy (100 percent) or idle (0 percent). Though HPE UFT is a desktop application, it consumes much CPU power than Selenium which runs in a web browser. The comparative analysis of both functional and performance aspects of Selenium and HPE UFT are summarized in Table 3.

7 CONCLUSIONS

Software testing plays a vital role in software quality management. Manual testing and automated testing are the main strategies followed. Automation testing is always preferred by a software industry to improve productivity and efficiency. Though manual testing is time consuming and costly than automated testing, it will figure out some errors which can't be found through test automation. Test automation tools are widely used in software industry to increase the productivity. But there is no tool which is perfectly automates software testing. There are many software tools useful for various programming domains and software applications. But Selenium and HPE UFT are widely used tools where the former is open-source and later is licensed. HPE UFT can be used for both webpage and desktop applications whereas Selenium restricted only for web applications. Though Selenium is freeware, we cannot chose it always, because of the need of an experienced test professional. HPE UFT is comparatively easier to use and develops test scripts in less time. Root cause analysis and recovery strategies are much better in HPE UFT than that of Selenium. Thus our evaluation results may guide a project manager to choose the best tool applicable for their project domain.

8 ACKNOWLEDGMENT

We acknowledge Providence College of Engineering, Chengannur for providing all the support for implementing this work.

REFERENCES

[1]. B. Beizer, "Software Testing Techniques". London: International Thompson Computer Press, 1990.

[2]. Glenford J. Myers, "The Art of Software testing", Second Edition, Wiley India Edition.

[3]. B. Beizer, "Black Box Testing", New York: John Wiley & Sons, Inc., 1995.

[4]. K. M. Mustafa, R. E. Al-Qutaish and M. I. Muhairat, "Classification of Software Testing Tools Based on the Software Testing Methods," Computer and Electrical Engineering, 2009. ICCEE '09. Second International Conference on, Dubai, 2009, pp. 229-233. doi: 10.1109/ICCEE.2009.9

[5]. T. E. J. Vos, B. Marín, M. J. Escalona and A. Marchetto, "A Methodological Framework for Evaluating Software Testing Techniques and Tools," 2012 12th International Conference on Quality Software, Xi'an, Shaanxi, 2012, pp. 230-239. doi: 10.1109/QSIC.2012.16

[6]. J. B. Michael, B. J. Bossuyt and B. B. Snyder, "Metrics for measuring the effectiveness of software-testing tools," Software Reliability Engineering, 2002. ISSRE 2003. Proceedings. 13th International Symposium on, 2002, pp. 117-128. doi: 10.1109/ISSRE.2002.1173225

[7]. Altaf, J. A. Dar, F. u. Rashid and M. Rafiq, "Survey on selenium tool in software testing," Green Computing and Internet of Things (ICGCIoT), 2015 International Conference on, Noida, 2015, pp. 1378-1383. doi: 10.1109/ICGCIoT.2015.7380682

[8]. R. Angmo and M. Sharma, "Performance evaluation of web based automation testing tools," Confluence The Next Generation Information Technology Summit (Confluence), 2014 5th International Conference -, Noida, 2014, pp. 731-735. doi: 10.1109/CONFLUENCE.2014.6949287

[9]. Van Vleck, T., "Three Questions About Each Bug You Find", ACM Software Engineering Notes, vol. 14, no. 5, July 1989.

[10]. SriivasanDesikan, Gopalaswamy Ramesh, "Software Testing: Principles and Practices", Pearson.

[11]. Pramod Mathew Jacob and M. Prasanna, "A Comparative analysis on black box testing strategies," International Conference on Information Science – ICIS –'16, Kochi, India, 2016

[12]. Rajib Mall, "Fundamentals of Software Engineering",Third edition, PHI

[13]. http://www.tutorialspoint.com/software_testing/software_testing_quick_guide.htm

[14]. Ian Sommeriele, "Software Engineering", Addison Wesley.

[15]. Pressman, "Software Engineering –A Practitioner's Approach".

[16]. http://www.softwaretestingclass.com/software-testing-life-cycle-stlc/

[17]. http://istqbexamcertification.com/what-is-software-testing-life-cycle-stlc/

[18]. http://www.softwaretestingclass.com/automation-testing-vs-manual-testing/

[19]. https://www.infosys.com/it-services/validation-solutions/white papers/documents/choosing-right-automation-tool.pdf

[20]. http://www.softwaretestinghelp.com/software-test-metrics-and measurements/

[21]. https://en.wikipedia.org/wiki/HP_QuickTest_Professional

[22]. https://en.wikipedia.org/wiki/Selenium_(software)

[23]. http://www.seleniumhq.org/download/

[24]. https://saas.hpe.com/en-us/download/uft

[25]. https://peopleorbit.vit.ac.in/hr_login.asp

[26]. Harpeet Kaur, Gagan gupta, "Comparative study of automated testing tools: Selenium, QTP and Test Complete" Journal of Engineering Research and Applications www.ijera.com ISSN : 2248-9622, Vol. 3, Issue 5, Sep-Oct 2013, pp.1739-1743

[27]. P. M. Jacob and P. Mani, "A Reference Model for Testing Internet of Things based Applications", Journal of Engineering, Science and Technology (JESTEC), Vol. 13, No. 8 (2018) ,pp. 2504-2519.

[28]. 23. P. M. Jacob and P. Mani, "Software architecture pattern selection model for Internet of Things based systems," in IET Software, vol. 12, no. 5, pp. 390-396, 10 2018. doi: 10.1049/iet-sen.2017.0206.

[29]. 24. Pramod Mathew Jacob and M. Prasanna, "A Comparative analysis on black box testing strategies," International Conference on Information Science – ICIS –'16, Kochi, India, 2016

[30]. 25. V. Bose, R. Roy, M. Nadirsha, B. Raj, Ajesh M and P. M. Jacob, "Gesture based painting system," 2016 International Conference on Information Science (ICIS), Kochi, 2016, pp. 23-27.

[31]. 26. P. M. Jacob, Muhammed Ilyas H, J. Jose and J. Jose, "An Analytical approach on DFD to UML model transformation techniques," 2016 International Conference on Information Science (ICIS), Kochi, 2016, pp. 12-17.

[32]. 27. Pramod Mathew Jacob and Prasanna Mani, "A Performance Estimation Model for Software Testing tools", ," International Journal of Engineering and Advanced Technology (IJEAT), vol. 8. no. 4, pp. 248-253, 2019

[33]. Jisha Mariyam John and Hariharan R.L, "An Intelligent Rider Assistant System using Machine Learning for two wheel vehicles," International Journal of Engineering and Advanced Technology (IJEAT), vol. 8, no. 6, pp. 1361-1366, 2019.

[34]. Priyadarsini S, Junie Mariam Varghese, Aparna Mahesh and Talit Sara George, "Shopping Spree: A Location based Shopping Application", International Journal of Engineering and Advanced Technology (IJEAT), vol. 8, no. 6, pp. 1451-1455, 2019.