A Framework For Evaluating Performance Of Software Testing Tools

Pramod Mathew Jacob, Prasanna Mani

Abstract: Software plays a pivotal role in this technology era. Due to its increased applicable domains, quality of the software being developed is to be monitored and controlled. Software organization follows many testing methodologies to perform quality management. Testing methodologies can be either manual or automated. Automated testing tools got massive acceptability among software professionals due to its enhanced features and functionalities than that of manual testing. There are hundreds of test automation tools available, among which some perform exceptionally well. Due to the availability of large set of tools, it is a herculean task for the project manager to choose the appropriate automation tool, which is suitable for their project domain. In this paper, we derive a software testing tool selection model which evaluates the performance aspects of a script-based testing tool. Experimental evaluation proves that, it can be used to compare and evaluate various performance characteristics of commercially accepted test automation tools based on user experience as well as system performance.

Index Terms: Automated testing, Software testing, Test script, Testing Tool, Test bed, Verification and Validation.

1 INTRODUCTION

SOFTWARE is advanced in recent days by enhancing its applicable domains. Software is embedded in almost all electronic gadgets and systems. In this scenario the quality of the software plays a significant role. The customer or end – user should be satisfied which is primarily depended on quality and capability of the software being developed. In order to ensure quality of the software, software organizations perform testing. Software Testing [1] is a quality control activity which involves defect detection and correction. Software Testing is a part of SDLC process to validate and verify the working of a developed product or application [2]. 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. When coding is completed, we perform Integration testing for finding out the bugs in the software application. It helps in preventing the software failure. The ultimate purpose of software testing is to satisfy the stakeholders as well as ensuring the quality of the application. Software industry usually follows automated testing than that of manual testing [3]. In manual testing, testers evaluate the software manually for the faults. Tester behaves like an end user and evaluates all the possible features and functionalities of the developed software for ensuring its behavior and quality. The tester manually prepares test plan [4] and suitable test cases which is tested over the application to verify the behavior of Graphical User Interface (GUI) [5], functional and non – functional requirements. 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. In automated testing, execution of test cases is supported by automation tools. Automated testing is quite beneficial in case of large projects.

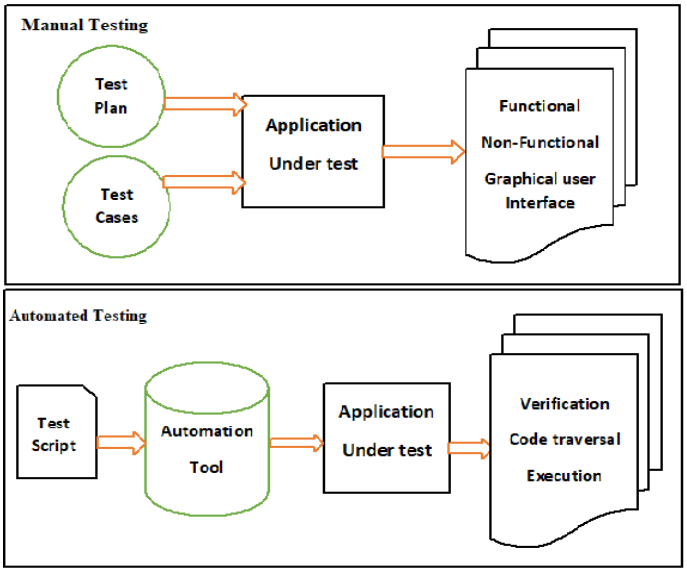


***Fig. 1**. Processes involved in manual and automated testing*

Here automation tools perform tests that repeat predefined actions, matches the developed program's probable and real results [6]. 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 behaviour by executing test cases. If the expected behaviour matches with the test results, the automation tool confirms the quality of the product. The Figure 1 and Figure 2 shows the process involved in both manual and 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. To develop high-quality software, it is essential to use software testing methods and tools (STMTs) effectively and efficiently [7]. In recent times, software organizations started supporting automated testing [8]. They use a wide variety of automation tools with versatile features and functionalities. The widely used automation testing tools includes Selenium [9], HPE UFT [10], TestComplete [11], Watir [12], Sahi [13] etc. The testing

- *Pramod Mathew Jacob, Research Scholar, SITE,VIT Vellore and Assistant Professor at Providence College of Engineering, Chengannur, email:pramod3mj@gmail.com*
- *Dr. Prasanna M, Associate Professor, SITE, VIT Vellore, email: prasanna.m@vit.ac.in*

personnel should choose the best one that suits their business and project domain perspectives. Performance metrics has some role to perform in this scenario. Though there are generic performance metrics available for software testing, there is no such performance evaluation model available for automation testing tools. The various existing metrics and works done in this area is discussed in the next session.

2 RELATED WORKS

We have conducted a literature survey on the works related to the performance metrics of software testing tools. Only a very few works have been done in this area so far and are summarized below. James B. Michael et al [14] analysed the existing testing metrics and has derived a metric suite for evaluating and choosing appropriate software testing tool. They considered parameters like ease of use, tool maturity and tool management, need of human intervention and control, test case generation capability and many more. They also derived some object oriented parameters like maximum allowable classes in a tool, maximum parameters allowable, Tool response time and Feature support level. These parameters are validated using LRDA Testbed, Parasoft Testbed, Telelogic Testbed and the results were analysed. Pawan Singh et al [15] has derived some metrics for quantifying the efficiency of testing tools. They considered the operational metrics like Toughness of user interface, customer satisfaction and tool age, user inconvenience, support for documentation, functional metrics, tool correctness and coverage etc. They have evaluated their metrics by using the popular testing tools like Quick Test Professional and WinRunner. The drawback of their work is, it requires some experience as well as needed some back history details of project being previously tested to estimate the values more accurately. Rigzin Angmo and Monika Sharma [16] has evaluated the open source web testing tools like Selenium and Watir. They evaluated the tool performance by considering Execution speed, Easiness of record and playback feature, required steps for testing and browser compatibility. They considered qualitative parameters rather than powerful quantitative metrics which is major drawback of their work. Khaled M. Mustafa et al [17] has classified the various available testing automation tools based on testing methodologies being used. They have evaluated nearly 135 testing tools and categorised 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. But though there are many tools available for testing the same domain, this work fails to provide some evaluation criteria for choosing one among them. T. E. J. Vos et al [18] has proposed a framework for evaluating software testing tools and techniques [5] which is successfully used for some case studies in EvoTest and FITTEST. They derived metrics for evaluating Generated test cases count, Task applicability, Tool reliability, Test case coverage and many more. But this framework fails to evaluate the performance constraints of automated testing tools like test script generation time, test script evaluation time and much more.

3 PROPOSED METRICS MODEL

Testing process is not always predefined. Different organizations follow different test methodology for different projects. Record and play back tools [21], Tools which uses test scripts and other CASE (Computer Aided Software Engineering) [22] tools are used for testing applications. Our aim is to derive test performance metrics for evaluating the automation testing tools which supports scripting. We propose a hybrid testing tool metrics to evaluate the testing tool performance and efficiency in which test scripts are considered as the basic parameter. Our proposed algorithm for estimating script-based metrics is illustrated below. It can be applied on a selected module of a software project. A software module is a set of functions or methods desirably possessing high cohesiveness. Each module may perform a set of functionalities and test cases should be derived for validating functional behaviours.

3.1 Algorithm for estimating the script-based metrics

1. Choose a test automation tool to be evaluated.
2. Choose the module or product to be tested.
3. Identify the number of functionalities or features to be evaluated.
4. Create minimal test suite for each feature or functionality of selected module.
5. Create test script for each test case and note the start time and finish time for each test case.
6. Compute the script creation time for each test script and estimate the total script creation time.
7. Execute the test scripts and note down script execution start time and finish time.
8. Compute total script execution time.
9. Estimate the tool reliability rate and productivity rate.
10. Plot the graph for all calculated parameters.
11. Repeat steps 4 to 11 for various tools to be evaluated
12. Analyse graph and compare parameters to derive conclusions on performance.

We need to derive some script-based performance metrics for evaluating the tool performance, though the model focuses on script based testing tools. Some basic metric can be time taken for generating test scripts, time for executing test scripts etc. Based on these estimated basic parameters we can derive metrics for tool productivity, tool reliability etc. Our model becomes hybrid by estimating metrics for tool learning time. The algorithm for estimating tool learning time is illustrated below. Tool learning time or tool learning time is a tool metric which has much impact in organizational overall productivity.

3.2 Algorithm for estimating the tool learning metrics

1. Choose a person to be trained. Categorize the person to any of the knowledge levels (Naïve, Moderate, Experienced) based on his previous experiences and achievements.
2. Note the starting time and finishing time for tool learning process.
3. Compute the overall tool learning time and stabilize it.
4. Repeat steps 1 to 3 for as many persons (N) you required to train.
5. Compute mean tool learning time.
6. Plot the values in graphical or tabular format.
7. Repeat steps 1 to 6 for various tools to be evaluated.
8. Analyse the graphs to tables to derive the conclusions based on performance metrics.

The above said algorithms are just generic algorithms that can

be applied in evaluating any automated testing tools that uses script. Based on the above discussed algorithms, we derived the following metrics for evaluating the performance.

4 DERIVED METRICS FOR EVALUATION

Software metrics is a mathematical value which conveys the degree of capability of some particular aspects of software [23]. Software metrics are categorized into process metrics, product metrics and project metrics. Testing tool metrics can be considered as a project metric though the project managers evaluate tool performance and efficiency to choose the best testing tool appropriate for their project domain. The various testing tool metrics used in our metric model are classified into three categories as shown in Figure 2.

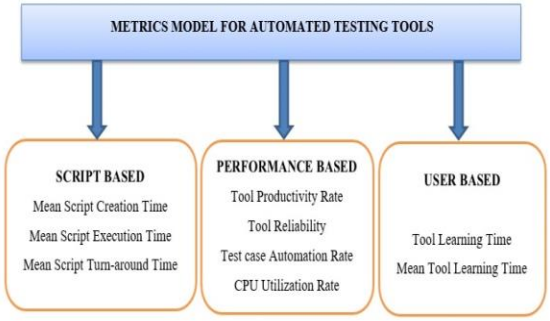


***Fig. 2**. Proposed metrics model*

The model consists of three categories as follows:

i. Script based: Metrics based on test scripts.
ii. User based: Metrics focuses on user experience.
iii. Performance based: Metrics based on performance and efficiency of the system.

The various metrics in each category are discussed below:

1. Script Creation Time (SCT): It is the time taken to generate the script of an individual test case. It depends upon the complexity of the module or test case being tested, number of input fields being tested etc. SCT increases as either of the above said parameters increases.
   SCT = SFT – SST (1)
   SFT: Script_creation Finish Time
   SST: Script_creation Start Time
2. 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. The tool with lesser MSCT value can be considered as an efficient tool.
   $MSCT = \frac{\sum_{i=1}^{n} SFTi - SSTi}{n}$ seconds/script (2)
   SSTi : Script_creation Start Time of i^{th} test script
   SFTi : Script_creation Finish Time of i^{th} test script
   n: Total number of scripts being created
3. Script Execution Time (SET): It is the time taken to execute a generated test script. SET value should be minimal for a good test automation tool.
   SET = EFT – EST (3)
   EST: Execution Start time
   EFT: Execution Finish time
4. 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. Tool with lesser MSET value is considered best.
   $MSET = \frac{\sum_{i=1}^{n} EFTi - ESTi}{n}$ seconds/script (4)
   ESTi : Execution_script Start Time of i^{th} test script
   EFTi : Execution_script Finish Time of i^{th} test script
   n: Total number of scripts being executed
5. Mean Script Turn-around Time (MSTT): It is the mean time required for a tool to create and execute a test script for a particular test case. Tool with minimal MSTT value is preferably good.
   $MSTT = \frac{\sum_{i=1}^{n} SCTi + SETi}{n}$ seconds/test case (5)
6. Tool Productivity Rate (TPR): It can be defined as the number of test cases executed per hour. For a good testing tool, TPR value should be higher.
   TPR =3600 / MSTT test cases/hour (6)
7. Total Learning Time (TLT): It is the time taken by an individual user to learn the testing tool to operate. For a preferred test automation tool, TLT value should be minimal.
   TLT = TFT – TST (7)
   TST: Training Start Time of a user
   TFT: Training Finish Time of a user
8. Mean Tool Learning Time (MTLT): It is the average time required for a user to learn the software automation tool to use. It can be also defined as the average time taken for training personnel to familiar with the software tool. Learning level and skill of a user is unquantifiable, though it depends upon the knowledge level of the user in using similar tools. In order to stabilize that, we use a metric variable called Knowledge Factor (KF). Knowledge Factor (KF) can have values 0.1, 0.2 and 0.3. For a naïve user who is completely unaware about automation tools, KF value will be 0.3. For a user with moderate knowledge in testing tools usage will have a KF value of 0.2. An experienced person who is familiar with similar tools or previous versions of this tool will have a KF value of 0.1. Tool with minimum MTLT is preferred by the project managers so that they can train their employees faster.
   $MTLT = \frac{\sum_{i=1}^{n} TLTi(1+KFi)}{n}$ hours/person (8)
   TLTi: Total Learning Time of i^{th} person
   KFi: Knowledge Factor of i^{th} person
   n: Total number of persons being trained
9. CPU Utilization rate: Percentage of CPU capacity being used for running the software tool.
10. Tool Reliability (TR): It is the probability of failure-free operation of the testing tool for a specific time period in a particular environment.
11. Test case Automation Rate (TAR): It is the capability of a testing tool to automate the test cases. The tool with high TAR value for a particular project is considered as best.
    $TAR = \frac{Number\ of\ test\ cases\ being\ automated}{Total\ number\ of\ test\ cases} * 100$ (9)

5 EXPERIMENTAL ANALYSIS AND RESULTS

We have applied our testing tool metrics on two software testing tools which supports script generation. We refer the tools as TOOL ABC and TOOL XYZ respectively. We have generated a test suite with five test cases. We have noted down the script creation start time, script creation finish time, script execution start time and script execution finish time. We entered these values in tables (Refer Table 1 and Table 2) to derive Script Creation Time (SCT), Mean Script Creation Time (MSCT), Script Execution Time (SET) and Mean Script Execution Time (MSET). The time shown in tables is in 24-hour format.

Based on the input data set shown in Table 1 and Table 2, we have derived the metric values for TOOL ABC and TOOL XYZ as shown in Table 3.

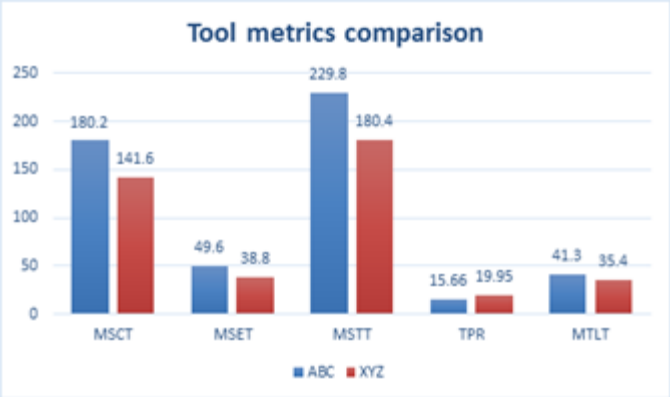
For evaluating the Mean Tool Learning Time (MTLT), we selected 10 persons each for training the tool ABC and tool XYZ. The persons we selected are from different user categories like naïve user, moderate user and experienced user. We assigned the KF value as 3, 2 and 1 for these user categories. The collected data values are displayed in Table 4.

Mean Tool Learning Time (MTLT) = $\frac{\sum_{i=1}^{n}(TLTi)*KFi}{n}$ hours/person

MTLT of TOOL ABC: = 413.2 / 10 = 41.3 hours /person
MTLT of TOOL XYZ: = 354.5 / 10 = 35.4 hours/person

The proposed metrics model is applied to the testing tools to derive the following tool performance metrics. The graphical analysis of various performance metrics is portrayed in Figure 3.

***Fig. 3**. Performance evaluation of TOOL ABC and TOOL XYZ*



From the evaluation graph it is easy for the project manager to trade-off between various testing tools that can be applicable for the project domains. If they need a testing tool with lesser training time, then the project manager can choose a test automation tool with minimum MTLT value. If they require a tool with better productivity, then project manager can choose a tool with high TPR value. So, this performance evaluation model can be used as a reference model for project managers or test engineers to choose a suitable automation tool.

6 CONCLUSIONS

Software industry is now moving towards automation of SDLC phases. Test automation is done using various tools like Selenium, HPE UFT, Winrunner, Junit and many more. Due to the availability of a wide range of automated testing tools, it is a hectic task for the project manager to choose the tools that is best fit for the project. The selection criteria is based on parameters like performance, efficiency, test automation productivity, tool learning time etc. Our model evaluates the above listed parameters by following a sequential procedure and derives the performance and efficiency metric values for script based testing tools. Our experiment results show that it is easier to analyse the execution speed and performance of test tools.

***Table 1:** Script Creation Time of TOOL ABC & TOOL XYZ*

Tool id	TOOL ABC			TOOL XYZ		
Test case_id	SST HH:MM:SS	SFT HH:MM:SS	SCT (In Seconds)	SST HH:MM:SS	SFT HH:MM:SS	SCT (In Seconds)
1	11:40:12	11:42:53	161	02:13:10	02:15:03	113
2	12:21:13	12:25:32	319	02:17:12	02:20:55	223
3	10:22:15	10:24:16	121	03:10:05	03:11:08	63
4	09:00:01	09:00:59	58	11:12:45	11:14:51	126
5	09:50:13	09:54:15	242	12:59:32	13:03:35	183
	Total Script Creation Time of ABC		901	Total Script Creation Time of XYZ		708

Table 2 *Script Execution Time of TOOL ABC & TOOL XYZ*

Tool id	TOOL ABC			TOOL XYZ		
Test case_id	EST HH:MM:SS	EFT HH:MM:SS	SET (In Seconds)	EST HH:MM:SS	EFT HH:MM:SS	SET (In Seconds)
1	12:40:20	12:41:10	50	02:33:18	02:34:03	45
2	13:15:18	13:17:01	103	12:32:12	12:33:21	69
3	14:22:10	14:22:23	13	13:19:45	13:20:08	23
4	09:30:11	09:30:58	47	04:18:45	04:19:11	26
5	09:50:28	09:51:03	35	14:59:32	15:00:03	31
	Total Script Execution Time of ABC		248	Total Script Execution Time of XYZ		194

Table 3 *Derived metric values for TOOL ABC and TOOL XYZ*

Metrics	Unit	TOOL ABC	TOOL XYZ
Mean Script Creation Time (MSCT)	seconds / script	180.2	141.6
Mean Script Execution Time(MSET)	seconds / script	49.6	38.8
Mean Script Turn-around Time (MSTT)	seconds / test case	229.8	180.4
Tool Productivity Rate (TPR)	test-cases / hour	15.66	19.95

Table 4 *Script Execution Time of TOOL ABC & TOOL XYZ*

TOOL ABC				TOOL XYZ			
User id	TLT (In hours)	KF of user	TLT(1+KF)	User id	TLT (In hours)	KF of user	TLT(1+KF)
1	30	0.1	33	A	25	0.3	32.5
2	35	0.2	42	B	30	0.3	39
3	38	0.2	45.6	C	36	0.1	39.6
4	40	0.3	52	D	40	0.2	48
5	45	0.3	58.5	E	40	0.2	48
6	42	0.3	54.6	F	32	0.2	38.4
7	25	0.1	27.5	G	18	0.1	19.8
8	35	0.2	42	H	26	0.1	28.6
9	20	0.1	22	I	18	0.2	21.6
10	30	0.2	36	J	30	0.3	39

ACKNOWLEDGMENT

The authors wish to thank VIT, Vellore and Providence College of Engineering, Chengannur for the technical support provided in executing the test cases.

REFERENCES

[1] Pramod and Prasanna, “A comparative analysis on Black box testing strategies,” in ICIS - 2016, IEEE Conference Proceedings, Kochi India, 2016.

[2] G. J. Myers, The Art of Software Testing, Wiley, 2011.

[3] R. M. Sharma, “Quantitative Analysis of Automation and Manual testing,” International Journal of Engineering and Innovative Technology, vol. 4, no. 1, pp. 252-257, 2014.

[4] D. B. V. R. Murthy, “Importance’s of Test Plan & Testing Procedure vs Verification & Validation – Researcher View,” International Journal of Scientific Engineering and Applied Science (IJSEAS), vol. 1, no. 9, pp. 408-412, 2015.

[5] M. L.-V. a. D. P. K. Moran, “Automated GUI Testing of Android Apps: From Research to Practice,” in IEEE International Conference on Software Maintenance and Evolution (ICSME), Raleigh, NC, 2016.

[6] “Software testing class,” [Online]. Available: http://www.softwaretestingclass.com/automation-testing-vs-manual-testing/. [Accessed 12 11 2016].

[7] S. K. D. L. J. Lee, “Survey on software testing practices,” IET Software, vol. 3, no. 6, pp. 275-282, 2012.

[8] R. Khalid, “Towards an automated tool for software testing and analysis,” in 14th International Bhurban Conference on Applied Sciences and Technology (IBCAST), Islamabad, Pakistan, 2017.

[9] “SeleniumHQ browser automation,” [Online]. Available: http://www.seleniumhq.org/. [Accessed 15 December 2016].

[10] “Unified Functional Testing (UFT),” Hewlett Packard Enterprise, [Online]. Available: https://saas.hpe.com/en-us/software/uft. [Accessed 12 January 2017].

[11] “TestComplete,” Smartbear, [Online]. Available: https://smartbear.com/product/testcomplete/overview/. [Accessed 14 December 2016].

[12] “Watir,” [Online]. Available: https://watir.com/. [Accessed 17 February 2017].

[13] “Sahi- The tester's web automation tool,” [Online]. Available: http://sahipro.com/. [Accessed 26 November 2016].

[14] J. B. Michael, Bernard, J. Bossuyt, Synder and B. B, “Metrics for measuring the effectiveness of software-testing tools,” 13th International Symposium on Software Reliability Engineering, 2002. Proceedings, pp. 117-128, 2002.

[15] P. Singh and Mulualem Wordofa Regassa, “Metrics for Quantification of the Software testing tools effectiveness,” American Journal of Software Engineering and Applications, vol. 4, no. 1, pp. 15-22, 2015.

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

[17] K. M. Mustafa, R. E. Al-Qutaish and M. I. Muhairat, "Classification of Software Testing Tools Based on the Software Testing Methods," 2009 Second International Conference on Computer and Electrical Engineering, Dubai, 2009, pp. 229-233..

[18] [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.20.

[19] G. R. SriivasanDesikan, Software Testing: Principles and Practices, Pearson.

[20] R. Mall, Fundamentals of Software Engineering, PHI.

[21] L. N. a. K. Doorgah, “Improving test data management in record and playback testing tools,” in International Conference on Computer & Information Science (ICCIS), Kuala Lumpeu, 2012.

[22] Y. L. a. C. Ye, “The analysis of CASE tools to the efficiency of software development,” in International Conference on Electronic & Mechanical Engineering and Information Technology, Harbin, Heilongjiang, China, 2011.

[23] [Online]. Available: http://sunnyday.mit.edu/16.355/metrics.pdf.