ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 4, Issue 1, July 2014

Quantitative Analysis of Automation and Manual Testing

R. M. Sharma
Assistant Professor
Department of Computer Application, M.C.N.U.J.C. Bhopal M.P. India

Abstract*— Testing is a major activity in software development process to find the defect in the software. Testing can be conducted manually as well as automated. Various types of metrics are collected during software development process and software testing process. These metrics provide quantitative approach to analyze any process of software engineering. This paper presents the concept of automation and manual testing and problem with manual testing and benefit of automatic testing. This paper also represents the analysis of automation and manual testing based on some testing metrics. The main objective of this research paper is to focus on, effectiveness and importance of automation testing.

***Index Terms*—Manual Testing, Automation Testing, Metrics, Test suits.**

I. INTRODUCTION

In recent years, software testing is becoming more popular and important in the software development industry. Testing is the critical element of software quality assurance and represents the ultimate review of specification, design, and coding. Software can be tested either manually or automatically.

To create test cases manually and execute them without any tool support is known as manual testing [6]. Manual software testing is performed by a human sitting in front of a computer carefully going through application screens, trying various usage and input combinations, comparing the results to the expected behavior and recording their observations [5].

Automation Testing means using an automation tool to execute test suite. Goal of automation is to reduce number of test cases to be run manually and not eliminate manual testing all together [5]. Some automation tools are: Winrunner, Loadrunner, JUnit, Silktest [10] etc.

II. SOFTWARE TESTING

Software testing involves experimentally and systematically checking the correctness of Software. Software testing is a process that ensures the quality of the product to its stakeholders with information about the quality of the product or service under test [5]. Software testing not only ensures that the software product functions properly under all conditions but also ensures that the product does not function properly under specific conditions. Different type of Software has different types of requirement. For example, video game software is completely different from banking software. The requirement of game user is different from banking user. In such situations when an organization develops or invests in a software product it needs to ensure that the software product will be acceptable to its end users, its target audience, its purchasers, and other stakeholders. Software testing is the process of attempting to make this evaluation.

A. Mode of Testing

Based on test execution, testing can be classified into two categories.

a. Manual Testing

b. Automation testing

1. Manual Testing

Manual testing is a testing technique where the test engineer prepare test case manually and execute them to identify defect in the software. It is most rigorous and old method of software testing [3]. Manual testing is a laborious activity that requires the tester to possess a certain set of qualities; to be patient, observant, speculative, creative, innovative, open-minded, and skillful. Repetitive manual testing can be difficult to perform on large software applications or applications having very large dataset coverage.

Problems with manual testing

***Time consuming and tedious*:** Since test cases are executed by human resources so it is very slow and tedious.

Huge investment in human resources: As test cases need to be executed manually so more testers are required in manual testing.

Less reliable: Manual testing is less reliable as tests may not be performed with precision each time because of human errors.

***Non-programmable*:** No programming can be done to write sophisticated tests which fetch hidden information.

Manual Testing can become boring and hence error prone.

2. Automation Testing

Automating software testing involves developing test scripts using scripting languages such as Python, JavaScript or Tcl (Tool Command Language), so that test cases can be executed by computers with minimal human intervention and attention. Test design and development together can be automated to reduce human effort and save cost [9]. The automation software can also enter test data into the system under test, compare expected and actual results and generate detailed test reports [3]. Test Automation demands

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 4, Issue 1, July 2014

considerable investments of money and resources. Successive development cycles will require execution of same test suite repeatedly. Using a test automation tool it's possible to record this test suite and re-play it as required. Once the test suite is automated, no human intervention is required [8]. Goal of automation is to reduce number of test cases to be run manually and not eliminate manual testing all together.

Benefits of automation testing

Fast: It is faster than the manual testing.

***Cost Effective*:** Test cases are executed by using automation tool so less tester are required in automation testing

Repeatable: The same test case (record and replay) can be re-executed using testing tools [9].

Reusable: Test suits can be re-used on different versions of the software.

Programmable: Testers can program sophisticated tests that bring hidden information.

Comprehensive: Testers can build test suites of tests that cover every feature in software application.

More reliable: Automation tests perform precisely same operation each time they are run.

***Test Coverage*:** Wider test coverage of application features.

III. TEST CASE

The test case ideally addresses one test requirement. A Test case consists of inputs given to program and its expected outputs and Precondition(s), if any identified by some testing methods. Expected output may contain post-condition(s), if any, and output which may come as a result when selected inputs are given to the software. Every test case will have a unique identification number. When we do testing, we set desired pre-condition, if any, given selected inputs to program and note observed output(s). We compare the observed output(s) with expected output and if they are the same, the test case is successful. If they are different, that is failure condition with selected input(s) and this should be recorded properly in order to find the cause of failure. The set of test cases is called a test suit [1]. A test suite contains detailed guidelines or objectives for each collection of test cases.

For example consider a program that is written for sum of two numbers. The program must accept two numerical inputs, perform the addition and display the output. But certain conditions exist, which can slow down the functioning of the program. Like if one input is zero. The program should correctly display the output, which is the number itself. What if one number is negative? Then the program should perform subtraction and correctly assign a positive or negative sign to the answer. The characteristics of a test case are that there is a known input and an expected output, which is worked out before the test. The known input should test a pre-condition and the expected output should test a post-condition. For Above example the following table (1) shows some test cases.

Table 1. Test Case

Test Case	Input Variable1	Input Variable2	Expected Output
1	5	6	11
2	0.8	0.8	1.6
3	-6	8	2
4	0	5	5
5	16.89	12.56	29.45

IV. TEST METRICS

Software Metric used to describe a measurement of a particular attribute of a software project. The Software Metrics that the Quality Assurance team produces are concerned with the test activities that are part of the test phase and so are formally known as software testing metrics [4]. Test metric enable quantitative insight into the effectiveness of the software testing process and provide feedback as to how to improve the testing process. Test metrics provide means for managing, tracking and controlling the status of testing [11]. There are several testing metrics that are in common use that can provide valuable insights of testing process.

A. Manual Test Metrics

1. Test Case Productivity metrics

Test case productivity metric used to provide the information about test progress or indication of test progress. This metric represent the test case design productivity (how many test case designed per hour) [13]. Table (2) shows the test cases name and number of test case to calculate test case productivity.

Table 2 Test Cases Productivity

Test Case Name	Test case
ABC_1	35
ABC_2	40
ABC_3	35
ABC_4	34
ABC_5	34
ABC_6	35
Total Test case	205

Example

Test case productivity = Total Test Cases/effort (hour)

Efforts: Suppose any tester create 205 test cases in 10 hours.

Then the test case productivity will be

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 4, Issue 1, July 2014

Test case productivity = 205/10=20.5 Test Cases/hour

We can compare the Test case productivity with the previous release(s) and draw the most effective conclusion about test progress. As shown in Figure (1) Y axis represents the test case productivity and X axis represent trends of every month release(s). The data is assumed for previous release(s) to draw the graph.

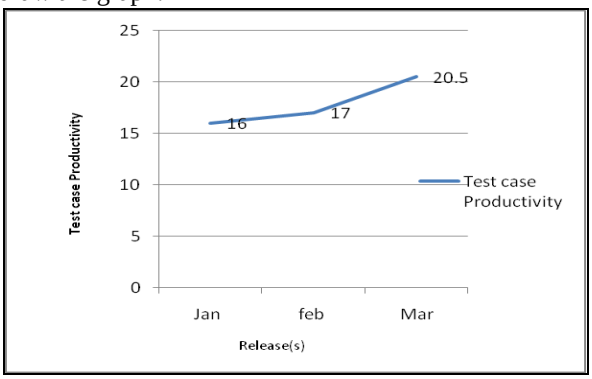

Fig 1. Test Case Productivity

B. Defect Acceptance Metrics

The test engineer determines number of valid defect during test execution. To get better picture we can compare the previous release(s) values. Figure 3 represent comparison of defect acceptance values of previous release. Table (3) shows the defect acceptance for various release(s) [12].

Defect Acceptance = (No. of Valid defect/Total No. of defect)*100

Example

Table 3. Defect Acceptance of Various Release(s)

Release(s)	Valid defect	Total defect	Defect acceptance %
1st	45	80	56
2nd	46	84	55
3rd	40	60	67

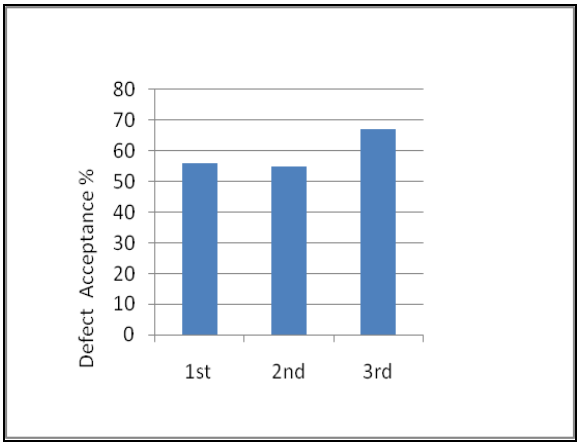

Fig 2 Defect Acceptance

C. Test Execution metrics

Test execution metric used to represent the status of test cases. It provide the status like (success, failure, or unable to run) of test cases. It gives the statically view of the release.

Figure 2, represent the test execution status, the pie graph show that the 60% of the test case execute successfully where 15% of test cases fail , and 12% test cases are unable to run (execute) for some reason like setup issue, out of scope or any other reason[11].

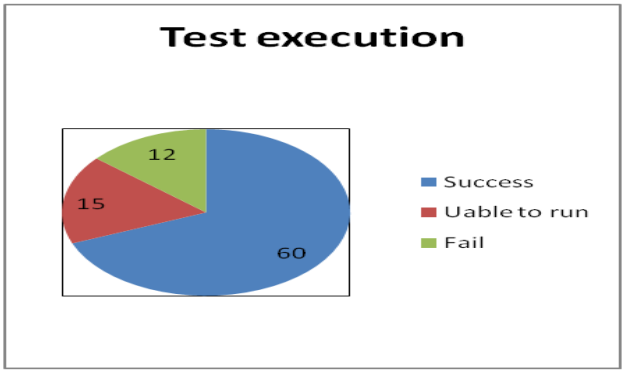

Fig 3. Test Execution Status

D. Defect Rejection Metrics

This metric determine the number of defects rejected during execution.

Defect Rejection = (No. of defect Rejected/Total No. of defect)*100 Table 4. Defect Rejection

Figure 4 represent the percentage of the invalid defect the testing team has opened and one can control, whenever required. Table (4) shows Defect Rejection of various Release(s).

Example

Release	Invalid Defect	Total Defect	Defect Rejection %
1st	12	60	20
2nd	13	75	17
3rd	14	70	20
4th	12	50	24

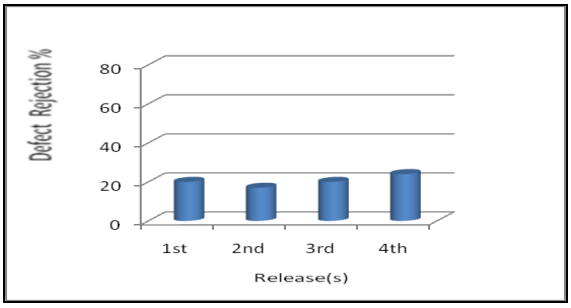

Fig 4 Defect Rejection

E. Bad Fix Defect Metrics

When test engineer fix some defect the new defect arises known as new defect(s) or bad fix defect. This metric used to determine the effectiveness of defect resolution process.

Bad fix Defect= (No. of Bad fix Defect(s)/Total No. of valid Defects)*100

It is used for defect controlling and monitoring the test

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 4, Issue 1, July 2014

process [13].

F. Test Execution Productivity Metrics

Test execution productivity metrics gives the average test execution per-day. It is used for performance evolution and time estimation.

Test Execution Productivity= (Total No. of Test Case Executed/Execution effort (hour))*8

The Test execution productivity is execution(s) per day.

Example

Table 5. Test Execution Status

Test case Name	Base Run effort (hr)	Re-run 1 status	Re-run1 effort (hr)	Re-run2 status	Re-run 2 effort (hr)	Re-run3 status	Re-run3 effort (hr)
ABC-1	1.2	T(0.66)	1	T(0.33)	0.34	T(1)	1
ABC-2	1.5	T(0.33)	0.4	T(1)	2		
ABC-3	2	T(1)	1.2				
ABC-4	1	T(0.66)	2				
ABC-5	2	T(1)	2.3				
ABC-6	2.3						

Where Test Execution is calculated as,

Total no. of test case execution=BTC+ (T (1)*1) +T (0.33)* 0.33) + (T (0.66)* 0.66))

Where,

(BTC) Base Test Case represents number of test case executed at least once. T (1) represent Number of test case retested with 71% to 100% of total test case steps and T (0.66) represent number of test case retested with 41% to 70% of total test Case steps where as T (0.33) represent the number of test case retested with 1% to 40% of total Test case steps

Total No. of Test Case execution = 6 + ((1*4) + (2*0.33) +(2*0.66)) = 11.98

Test Execution Productivity = (11.98/19.24) * 8 = 4.98 Execution(s)/day. The graph in figure (4) represents the comparison of previous release(s)(assumed data) and can have an effective conclusion.

Table (6) Types of Test Case and Values

Base Test Case	6
T(1)	4
T(0.66)	2
T(0.33)	2
Total Efforts(hr)	19.24

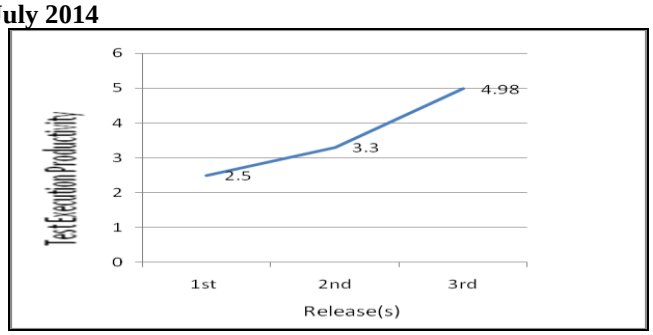

Fig 5 Test Execution productivity

G. Test Execution Efficiency metrics

Some time it is very necessary to evaluate the performance of testing team. This metric is used to determine the efficiency of the testing team in identifying the defects. This metric also used to find out the defect missed out during testing process [13].

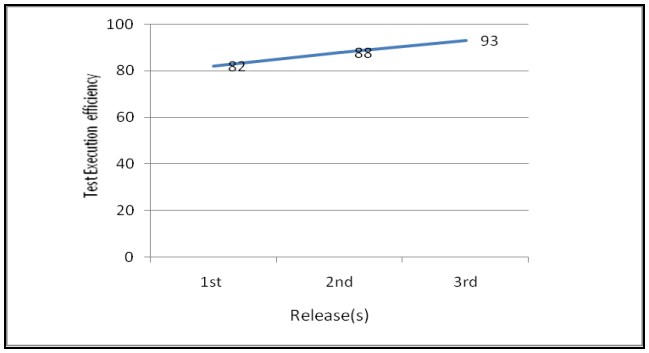

Fig 6 Test Execution Efficiency

Test Efficiency = (No. of valid defect during testing process / (No. of valid defect during testing process + No. of valid defect identified after release))*100

Testing team identified some valid defect during testing process but after release of software user also identified some valid defect so this metric is very important to calculate the efficiency of testing team. The graph in figure (5) represents the comparison of test efficiency of (assumed data for efficiency) previous release(s).

V. AUTOMATION TESTING METRICS

A. Automation Scripting Productivity

In automation testing the testing tools are used to test the software [13]. In this type of testing, test team create the test script using automation tools to test software; this metric gives the scripting productivity for automation test script. This metric used for progress evaluation of automated testing [12].

Automation Scripting Productivity= (Total Operation Performed /Total effort (hr))

Where Operations performed is: No. of click, No. of input parameter, check point etc.

Example

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 4, Issue 1, July 2014

Table 7 Types of Operations and No. of Operation Performed

Operation Performed	Total
Number of click	21
No. of input parameter	10
No. of check Point	7
Total Operation	38

Efforts = Scripting in 12 hours.

Automation Scripting Productivity =38/12=3.17 Operation(s)/hour

The graph in figure (6) shows the comparison of previous release(s), the data is assumed for previous release(s).

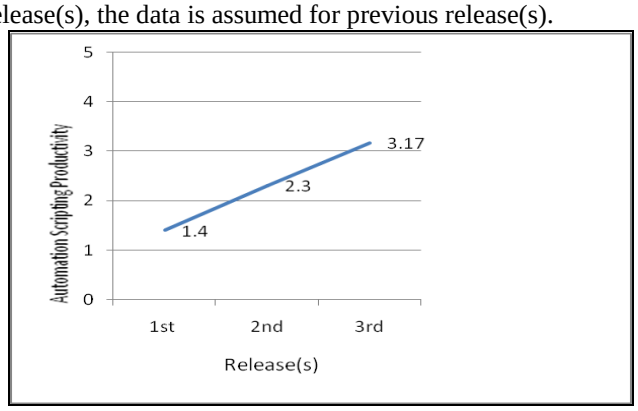

Fig 7 Automation Scripting Productivity

B. Automation Test Execution Productivity

This is same as manual test execution productivity it give the result in test execution per-day.

C. Automation Coverage metrics

In automation testing we use testing tools to test the software; it is required to calculate test coverage. Test coverage means how much we cover with testing tools or how much we automate manual process. This metric used to determine how much automation actually achieved using particular test tool. This metric gives the percentage of manual test cases automated [2].

Automation Coverage= (Total No. of Test Case Automated/ Total No. of Manual Test Case)*100

Example

If there are 100 Manual test cases and one has automated 80 test cases then Automation Coverage = 80%

D. Cost Comparison Metrics

The testing is major cost factor in software development. The test automation has been proposed as one solution to reduce the testing cost [12]. Test automation tools provide the facility to run repetitive test case to speedup testing process but they require proper training to run test cases in particular test tool. The following metrics gives the cost comparison between manual testing and automation testing [13]. This metrics is used analysis of ROI (return on investment).

Manual Cost is evaluated as: -

Cost (Manual) = Execution Efforts (hours) * Billing Rate

Automation cost is evaluated as: -

Cost (Automation) = Tool Purchased Cost (One time investment) + Maintenance Cost + Script Development Cost + (Execution Efforts (hrs) * Billing Rate)

In automation testing the Script is re-used so script development cost will be the script update cost.

VI. CONCLUSION

Manual testing is time consuming, tedious and requires heavy investment in human resources. Automation tools enable us to record the test suite and re-play it if required. Once the test suite is automated, no human intervention is required. In automation testing the initial investments are bigger than manual testing and you cannot automate everything but automatable test cases, determine which ones (manual or automated) would provide the biggest return on investment. Metrics are an important to analyze the quality, and progress of an automated software testing and manual testing effort. The test metrics provides the visibility into the readiness of the product and give clear measurement of the quality and completeness of the product. This paper discussed some manual testing metrics and automation metrics like test execution, test case productivity, defect rejection metrics test coverage metrics to evaluate the performance of both types of testing.

REFERENCES

[1] Yogesh Singh "Software Testing" Cambridge University Press ISBN 978-1-107-01269-7 First edition 2012.

[2] Dr.S.S.Riaz Ahamed "Studying The Feasibility and Importance of Software Testing: An Analysis" International Journal of Engineering Science and Technology Vol.1(3), 2009, ISSN: 0975-5462 pp.119-128.

[3] Prof. (Dr.) V. N. Maurya, Er. Rajender Kumar " Analytical Study on Manual vs. Automated Testing Using with Simplistic Cost Model" International Journal of Electronics and Electrical Engineering ISSN : 2277-7040 Volume 2 Issue 1 (January 2012)pp.24-35.

[4] Norman E Fenton , Martin Neil "Software Metrics: Roadmap" The Future of Software Engineering, Anthony Finkelstein (Ed.), ACM Press 2000 Order number is 592000-1, ISBN 1-58113-253-0.

[5] Abhijit A. Sawant, Pranit H. Bari and P. M. Chawan " Software Testing Techniques and Strategies" International Journal of Engineering Research and Applications (IJERA) Vol. 2, Issue 3, May-Jun 2012, pp.980-986.

[6] Shivkumar Hasmukhrai Trivedi "Software Testing Techniques" International Journal of Advanced Research in Computer Science and Software Engineering Volume 2, Issue 10, October 2012 ISSN: 2277 128X pp. 433-438.

[7] Vivek Kumar "Comparison of Manual and Automation Testing" International Journal of Research in Science and Technology (IJRST) 2012, Vol. No. 1, Issue No. V, Apr-Jun ISSN: 2249-0604.

[8] Rudolf Ramler and Klaus Wolfmaier "Economic Perspectives in Test Automation: Balancing Automated and Manual Testing with Opportunity Cost" AST'06, May 23, 2006, Shanghai, China. Copyright 2006 ACM 1-59593-085-X/06/0005 pp. 85-91.

ISSN: 2277-3754
ISO 9001:2008 Certified
International Journal of Engineering and Innovative Technology (IJEIT)
Volume 4, Issue 1, July 2014

[9] Vishawjyoti, Sachin Sharma " Study and Analysis of Automation Testing Techniques" Journal of Global Research in Computer Science Volume 3, No. 12, December 2012 pp. 36-43

[10] Saket Vihari, Arun Prakash Agrawal, "A System of Humanizing Test Automation Outlay Efficiency" International Journal of Emerging Science and Engineering (IJESE) ISSN: 2319–6378, Volume-1, Issue-7, May 2013pp 74-78

[11] Sheikh Umar Farooq, S. M. K. Quadri, Nesar Ahmad, "Software Measurements and Metrics: Role in Effective Software Testing" International Journal of Engineering Science and Technology (IJEST) ISSN: 0975-5462 Vol. 3 No. 1 Jan 2011 pp 671-680.

[12] Mr. Premal B. Nirpal, Mr. Premal B. Nirpal "A Brief Overview Of Software Testing Metrics" International Journal on Computer Science and Engineering (IJCSE) ISSN: 0975-3397 Vol. 3 No. 1 Jan 2011 pp. 204-211.

[13] www.mindlance.com/testing.

AUTHOR BIOGRAPHY/

R.M.Sharma: Working as a assistant professor in department of Computer Applications at MCRPV Bhopal. More than 13 year teaching in UG and PG classes in the field of software engineering, computer network and compiler design.