Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X
IMPACT FACTOR: 5.258

IJCSMC, Vol. 5, Issue. 6, June 2016, pg.211 – 217

TO IMPROVE CORRECTNESS, QUALITY AND REDUCING TESTING TIME

Rajiv Sharma, Mohit Sangwan
Assistant Professor, Department of CSE, Shri Baba Mastnath Engineering College
M.Tech student, Department of CSE, Shri Baba Mastnath Engineering College
mohitcse9@gmail.com, sonurajiv_007@rediffmail.com

Abstract- Software testing is an essential activity in software engineering, it is a discipline as well as an iterative process, which consists of Tests Designing, Tests Execution, Problems Identifying and Problem Fixing, for validating functionality and as well as for attempting the software break. Testing built the confidence of user/ developer about the software but it can't guarantee the production of high quality software. We do Software testing to find problems and to fix them for improving software quality. Software testing may consume 35%-40% of a software development budget. Manual and Automated Testing methods seems complementary to each other. Purpose of testing is to find out Defects and causes and fixed them as early as possible. Testing requires more project effort and time than any other software development activity; therefore it needs a suitable strategy to make testing successful. Testing is an activity to find the bugs in software that may perform by tester or by applying strategies like white box or black box. So, the activities involved in the testing should be in planned way.
Keywords---BVA, UAT, V&V, MCQRTT, HTTP.

I. INTRODUCTION

Software testing is focused on finding defects in the final software before give it to the user. So it is the responsibility of the developer and the tester that he/she will examine all core functionality and the components associated with the software.

- **Verification** – Are we building the product right? It refers to the correctness of the function specifications.
- **Validation** – Are we building the right product? It refers to the user expectation whether the product developed meets the user requirement or not.

Verification focuses on the system (software, hardware, documentation, and personnel) that complies with an organization's standards and processes, relying on review or non-executable methods. While Validation physically focuses on that the system operates according to plan by executing the system functions through a series of tests that can be observed and evaluated. Verification answers the question, "Did we build the right system?" while validations

addresses, "Did we build the system right?" Determining when to perform verification and validation relates to the development, acquisition, and maintenance of software.

The rest of this paper is organised as follows:section II summarizes related researches.Section III gives a brief introduction of Testing conditions .Section IV describes methods used in testing phase.Section V describes Experimental results and analysis.In section VI,we draw conclusion and give future work.

II. RELATED WORK

Gregory M. Kapfhammer "Software Testing" 2008.Testing [3] is an important technique for the improvement and measurement of a software system's quality. Any approach to testing software faces essential and accidental difficulties. Indeed, as noted by Edsger Dijkstra the construction of the needed test programs is a "major intellectual effort". While software testing is not a "silver bullet" that can guarantee the production of high quality applications, theoretical and empirical investigations have shown that the rigorous, consistent, and intelligent application of testing techniques can improve software quality. Hyunsook Do, Siavash Mirarab, Ladan Tahvildari "An Empirical Study of the Effect of Time Constraints on the Cost-Benefits of Regression Testing" 2008.
Regression testing [5] is an expensive process used to validate modified software. Test case prioritization techniques improve the cost effectiveness of regression testing by ordering test cases such that those that are more important are run earlier in the testing process. Many prioritization techniques have been proposed and evidence shows that they can be beneficial. It has been suggested, however, that the time constraints that can be imposed on regression testing by various software development processes can strongly affect the behavior of prioritization techniques. Therefore, we conducted an experiment to assess the effects of time constraints on the costs and benefits of prioritization techniques. Goutam Kumar Saha "Understanding Software Testing Concepts" 2008.Software testing [6] concepts have been briefly described in this paper. It easier to understand fundamental concepts of software testing by going through a concept map thereof. Software testing is itself a discipline as well as a process. Software development is nothing but a process of coding functionality in order to meet the defined end-user requirements. We can think of software testing as an iterative process, which consists of Tests Designing, Tests Execution, Problems Identifying and Problem Fixing, for validating functionality and as well as for attempting the software break. Software testing aims to find problems and to fix them for improving software quality. Software testing may represent 40% of a software development budget. Basic methods of performing software testing include Manual Testing and Automated Testing. User acceptance testing is not an "accept or reject" proposition any more. UAT is rather more about finding gaps between "how the completed system works" and "how business operational processes are performed". Also, UAT is generally considered to be a "validation process" rather than a "verification process". "Validation" determines if something works as intended in the user's environment and meets their needs. "Verification" determines if something has been built according to specifications. It is a common belief that well-documented Business Requirements are the basis for UAT and exhaustive requirements specification would be of great help during UAT. a well-defined UAT process together with a feasible amount of test automation would yield productive and effective acceptance test results.

III. TESTING CONDITIONS

Software testing should develop a sufficient assessment of quality, at a reasonable cost and at timely decisions to be made concerning the software. Sufficient testing means when all the necessary required testing is to be done to check the functionality of software for all possible usage scenarios. The definition of "enough testing" is when all the testing necessary to prove that the end software is functional for all possible scenarios and over the full software lifecycle was concluded. The only case in which a tester can truthfully say that he or she has to run enough tests is if the software under test is: fairly simple and independent, and the expected environmental conditions, activation profiles and parametric changes are well defined and properly modeled.

Over the years tester stops testing when following 6 conditions are satisfied:

1. The first condition for testing is to have good quality specifications for each software lifecycle step. It means that we have to get involved earlier than usual in the software design and verification cycle, by starting to make our test cases even before the specifications are released. Using this approach, we can give the design engineers early feedback on the requirements quality and completeness.

2. The second condition for testing is for all the specifications for the different integration levels to be in synchronized way.

- To use the same signal names.
- Lower level condition to be built on the higher level specification functional definitions, rather than redefining them, etc.

3. The third condition for testing is for each of the specifications to contain the complete requirements set for generating tests at that particular integration level. The more source documents one has to use for test generation, the higher the probability that the tests will be written for the wrong objective.

4. The fourth condition for testing is to have additive, rather than repetitive tests at the different integration stages. In the traditional, repetitive approach to testing we are typically running the same set of tests at multiple integration levels. In the case of additive testing the higher integration level tests are built on top of the ones already executed. For example, in the case of a multiple board system, at full system integration level, we would concentrate on testing the board to board interfaces, building on the fact that all the constituent boards were completely verified at an earlier stage

5. The fifth condition for testing is to run each of the tests at the integration stage that either yields the maximum information about the product quality, or all things being equal, it is the cheapest to run. For the latter we have to keep in mind that the overall cost of test increases with around one order of scale each time we reach another integration level**.**

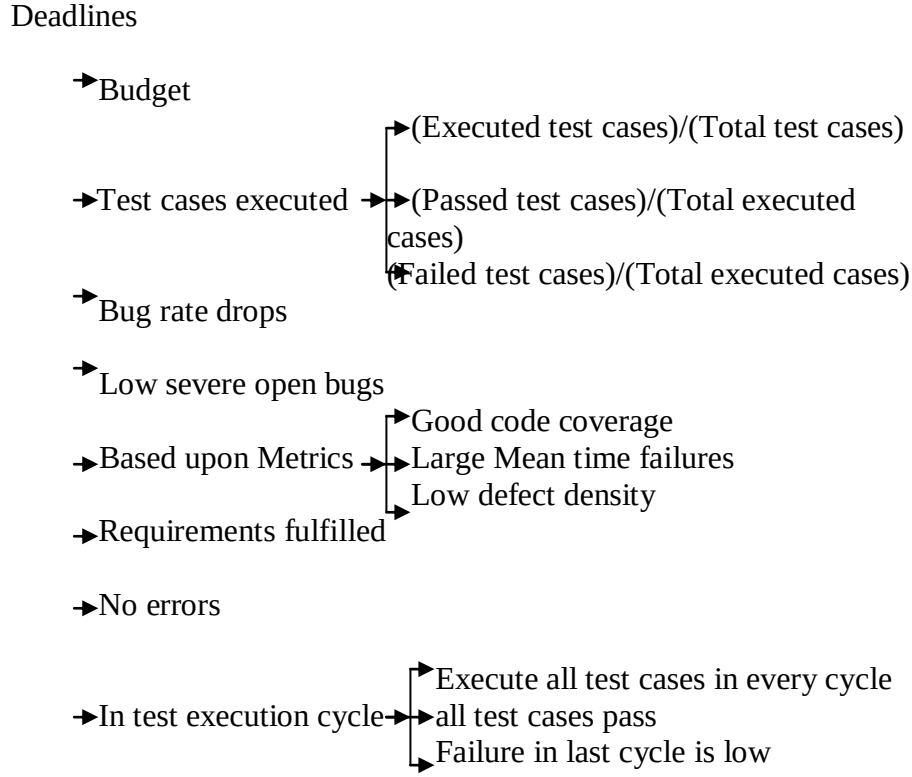
6. A sixth and most obvious prerequisite to software testing is to automate the test process as much as possible.

IV. METHODS USED IN TESTING PHASE

At some point, tester has to stop testing and ship the software. After observing all the details of Software testing we come to a conclusion that "testing can never be considered complete". Tester can never be proved theoretically or scientifically that the software is free from errors now.

Basically testers stop testing when:

- The planned testing deadlines are about to expire.
- Not able to detect any errors even after execution of all the planned test Cases.



(Fig:1) Method used in testing phase

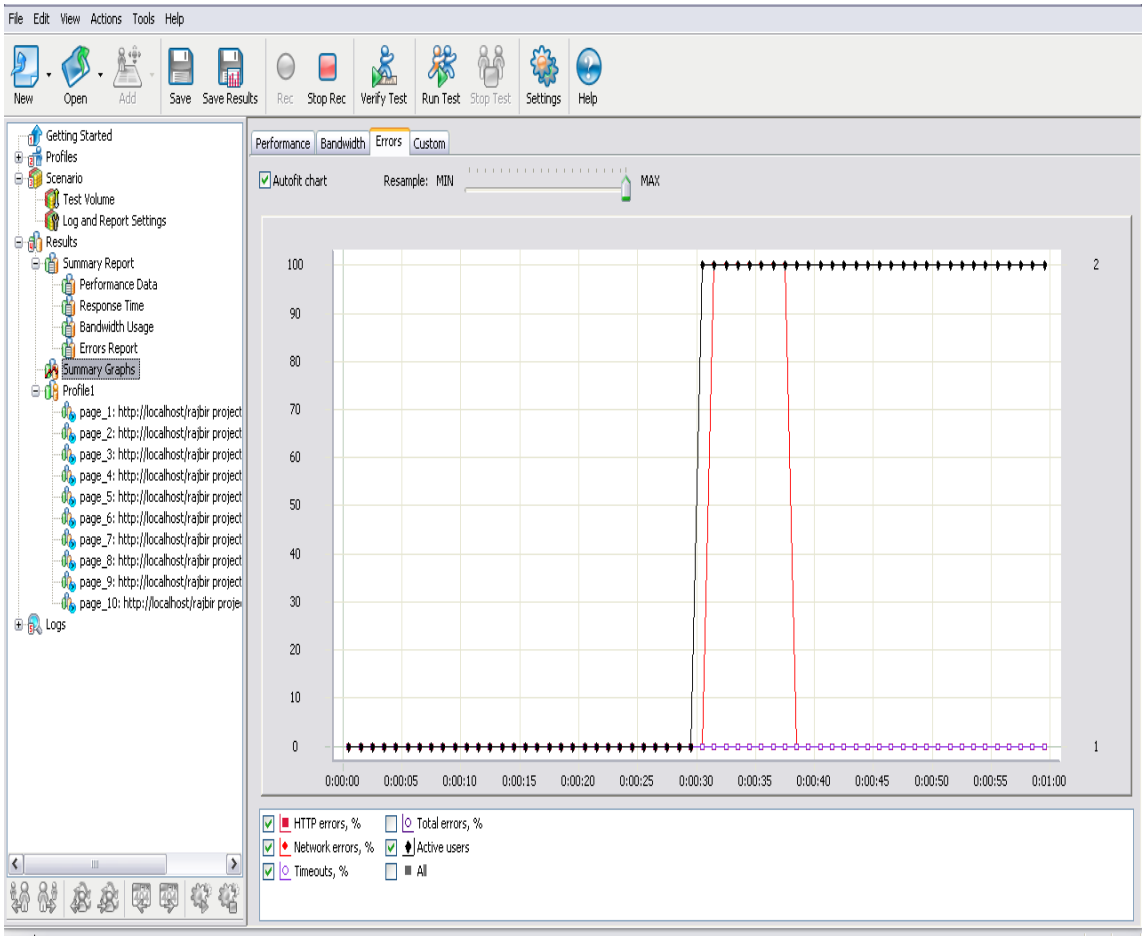
Testing metrics helps the testers to take better and accurate decisions; like when to stop testing or when the application is ready for release, how to track testing progress & how to measure the quality, correctness and error occurred of a product at a certain point in the testing cycle. The best way for tester is to have a fixed number of test cases ready well before the beginning of test execution cycle. Finally measure the testing progress by recording the total number of test cases executed using the following metrics which are quite helpful in measuring the quality, correctness and error occurred of the software product.

V. EXPERIMENTAL ANALYSIS AND RESULTS

Graphical results

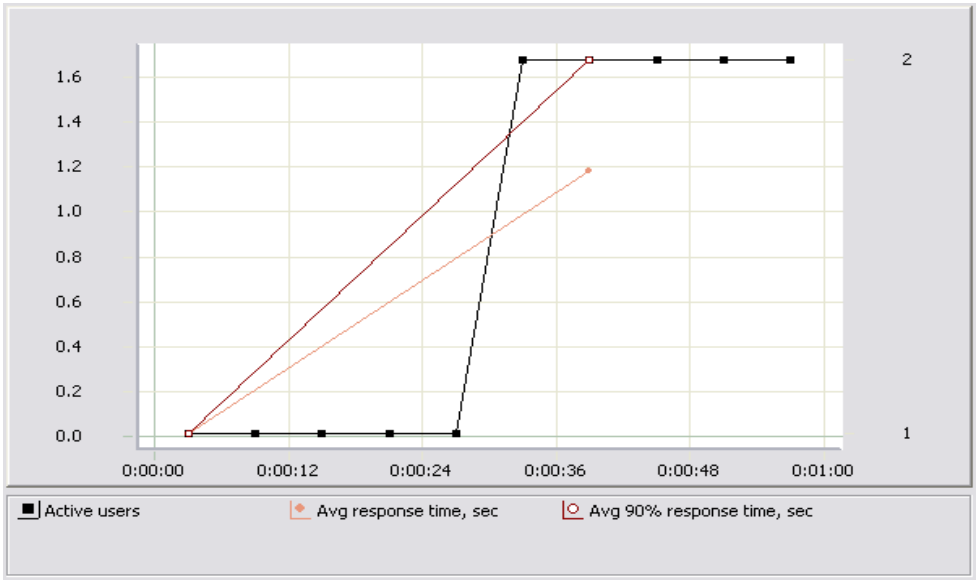
Now using WAPT results are represented in the form of Graphs and graphs for each user profile and single request are shown below:

Error rate - number of page requests completed with errors as a percentage of total number of requests or sessions. Errors can be either reported by the server or detected as a result of network problems, wrong server response and timeouts.



(Fig:2)Error Graph

Response time - time required by the web site to provide a correct reply to a single page request. It can be measured together with the time required to download all page elements or without it.



(Fig:3)Response time Graph

VI. CONCLUSION

Software Testing consumes large amounts of resources in development projects. Hence, it is of general interest to assess the effectiveness and efficiency of current test methods and compare these with new or refinements of existing test methods to find possible ways of improving the testing activity. Software activity like testing is essential for software quality but also consume a large part of software development costs. Therefore efficient and cost effective software testing are both a priority and an essential considering the demands to decrease time-to-market and intense competition faced by many. It is then perhaps not unexpected that decisions related to software quality, when to stop testing, testing schedule and testing resource allocation needs to be as accurate as possible. Although software testing is crucial to build a good quality software. This dissertation elaborates all testing strategies, their interrelations, when to perform and how. We design an interrelation diagram of various testing strategies. Tester can define the time to stop testing is when, all the test cases, derived from white-box testing methods like equivalent partitioning, cause-effect analysis & boundary-value analysis are executed without detecting errors.

This dissertation elaborates of when to stop software testing, and the methods discussed should be executed sequentially so that the tester can assure the customer that the software quality is on the higher side and all types of requirements are fulfilled and the software can have negligible number of errors at the end of testing phase.

FUTURE SCOPE

The future scope includes a novel scheme to help in finding out the sufficient level of testing. Quality of software may also be achieved by using restricted testing. And the implementation result shows the betterment in the quality, time, budget, correctness, completeness and consistency of the software.

Trends in the industry suggest that software testing in the future will look very different than it does today. The major trends include greater adoption of SQA, web services, SaaS, wireless and mobile technologies. Each of these trends is further complicated by a more agile approach to software development and an increasing emphasis on the **Repeatability, Reliability, Re-use and Robustness.**

REFERENCES

[1] Robert Nilsson and Jeff Offutt "Automated Testing of Timeliness: A Case Study" Published in IEEE 2007.

[2] Glenford J. Myers "The Art of Software Testing, Second Edition" published in 2004. Antonia Bertolino" SOFTWARE TESTING" published in *IEEE – Trial (Version 0.95) – May 2001.*

[3] R. Boddu, G. Lan, G. Mukhopadhyay, B. Cukic, "RETNA: from requirements to testing in a natural way," 12th IEEE International Requirements Engineering Conference, pp.262-271, 2004.

[4] Beck, K. (2003). Test Driven Development -- by Example. Boston, Addison Wesley.

[5] E. van Veendaal (2008), Test Improvement Manifesto, in: *Testing Experience*, Issue 04/08, December 2008.
[6] X. Qu, M. B. Cohen, and G. Rothermel. Configurationaware regression testing: an empirical study of sampling and prioritization. In ISSTA '08, 2008.
[7] R. Lutz, I.C Mikulski, "Requirements discovery during the testing of safety-critical software," Software Engineering 25th IEEE International Conference, pp.578-583, 2003.
[8] Wagner, S. & Seifert, T. 2005. Software Quality Economics for Defect Detection Techniques Using Failure Prediction. In Proceedings of the 3rd Workshop on Software Quality (3-WoSQ). ACM Press.
[9] S. Berner, R. Weber, R.K. Keller, "Requirements & testing: Observations and lessons learned from automated testing," Proceedings of 27th international conference on software Engineering, pp.571-579, 2005.
[10]Boehm, B., Huang, L., Jain, A. & Madachy, R. 2004. The ROI of Software Dependability: The iDAVE Model. *IEEE Software*, 21(3).
[11]K. R. Walcott, M. L. Soffa, Gregory M. Kapfhammer and Robert S. Roos. Time-aware test suite prioritization. In *Proceedings of the 2006 International Symposium on Software testing and Analysis*, page 1-12, July 2006.
[12]Peter Sestoft" Systematic software testing" published in IT University of Copenhagen, Denmark Version 2, 2008-02-25.
[13]D. Jeffrey and N. Gupta. Test Case Prioritization Using Relevant Slices. In *Proceedings of Computer Software and Applications COMPSAC'06*, Chicago, USA, Pages 411-420, 2006. K. Yukse, S. Dupont, D. Harnoir and C. Froidure, "FTTx automated test solution: Requirements and experimental implementation," IEE Magazine Electronics Letter, Vol.41, No.9, pp.546-547, 2005.
[14]Williams, L., E. M. Maximilien, et al. (2003). Test-Driven Development as a Defect- Reduction Practice. IEEE International Symposium on Software Reliability Engineering, Denver, CO, IEEE Computer Society.
[15]M. B. Cohen, J. Snyder, and G. Rothermel. Testing across configurations: implications for combinatorial testing. SIGSOFT Softw. Eng. Notes, 31(6):1–9, 2006.
[16]Herzlich, P. 2005. *The Need for Software Testing*. Ovum Research: London, UK.
[17]Gregory M. Kapfhammer "Software Testing" ACM 2008.
[18]John E. Bentley "Software Testing Fundamentals—Concepts, Roles, and Terminology" 2005.
[19]Goutam Kumar Saha "Understanding Software Testing Concepts" ACM 2008.
[20]Mark Uttinga, Alexander Pretschnerb and Bruno Legeardc"A Taxonomy of Model-Based Testing" White paper in 2007.
[21]*Peter Miller '*'Testing? What testing?" 2000.
[22]SANTOSH KUMAR SWAIN, SUBHENDU KUMAR PANI, DURGA PRASAD MOHAPATRA "MODEL BASED OBJECT-ORIENTED SOFTWARE TESTING" Journal of Theoretical and Applied Information Technology in 2006.
[23]Cem Kaner, J.D, Ph.D." The Ongoing Revolution in Software Testing" Software Test & Performance Conference, December 8, 2004.
[24]Inspection vs. Testing 2003.
[25]Gerry Gaffney "Conducting a Walkthrough" 2002.
[26]"USER ACCEPTANCE TESTING (UAT) PROCESS" 2008.
[27]Andreas Leitner, Ilinca Ciupa, Bertrand Meyer "Reconciling Manual and Automated Testing: the AutoTest Experience" 40th Hawaii International Conference on System Sciences – 2007.
[28]Rex Black "Shoestring Manual Testing" 2001.
[29]OTS Solution "Manual Testing V/S Automated Testing" 2006.
[30]Carl Erickson, Ralph Palmer, David Crosby, Michael Marsiglia, Micah Alles"Make Haste, not Waste: Automated System Testing" 2004.
[31]*Yashwant K. Malaiya"* Automatic Test Software" 2000.
[32]Girish Janardhanudu "White Box Testing" 2009.
[33]Laurie Williams "White-Box Testing" 2006.
[34]Kwang Ik Seo Eun Man Choi "Comparison of Five Black-box Testing Methods for Object-Oriented Software" Fourth International Conference on Software Engineering Research, IEEE 2006.
[35]Redstone Software" Black-box vs. White-box Testing: Choosing the Right Approach to Deliver Quality Applications" 2005.
[36]Laurie Williams" Testing Overview and Black-Box Testing Techniques" 2004.
[37]Sami Beydeda, Volker Gruhn, Michael Stachorski "A Graphical Class Representation for Integrated Black- and White-Box Testing" 2005.

[38]Harish V. Kantamneni Sanjay R. Pillai Yashwant K. Malaiya "Structurally Guided Black Box Testing" 2002.
[39]R.Venkat Rajendran" White paper on Unit Testing" 2000.

