ISSN: 2348 - 2117

International Journal of Engineering Technology and Computer Research (IJETCR)
Available Online at www.ijetcr.org
Volume 5; Issue 4; July-August: 2017; Page No. 73-80
Journal Approved by UGC

EMPIRICAL STUDY OF LATEST SOFTWARE TESTING TECHNIQUES AND TEST METRICS

Mr. Dheeraj[1], Dr. Kalpana Sharma[2]

[1] Research Scholar, Department of Computer Science and Engineering, Bhagwant University, Sikar Road, Ajmer, Rajasthan 305004, India

Dheeraj.chhillar@gmail.com

[2] HOD, Department of Computer Science and Engineering, Bhagwant University, Sikar Road, Ajmer, Rajasthan 305004, India

Kalpanasharma56@gmail.com

Received 14 July 2017; Accepted 02 August. 2017

Abstract

With the increasing complexity of today's software applications, software testing tools, techniques and methodologies are also getting intricate. Few decades ago, there used to be window based applications and later when internet age came in early 90s, most of the applications moved to web. Web based applications have also taken shape of mobile applications, which are hosted majorly on cloud. Now a days computing devices are embedded in all objects which can communicate with each other over internet, popularly called as IOT (Internet of Things). Paper currency has also taken shape of digital currency and these days crypto currencies are in inception phase. Now a days ensuring software interoperability and error free transactions have become a challenge. In this paper, we provide an accounting of some of the effective work accomplished in software testing in past and most important issues and challenges in this field. To be further comprehensive in this determination, and to go outside our own individual views and prejudices, we started by communicating with many of our associates who are lively in the software testing field. We queried them for assistance and to understand latest trends in software testing. The research is conducted to understand issues and challenges faced by current industry and opportunities for future research in this field.

Key Words: Algorithms, Experimentation, Verification, Testing Methodologies, Software Testing Life Cycle, Testing Frameworks, Automation Testing, Test Driven Development, Test optimization, Quality Metrics.

1. INTRODUCTION

Testing is defined as a process of evaluation that either the specific system meets its originally specified requirements or not. It is mainly a process encompassing validation and verification process that whether the developed system meets the requirements defined by user. Therefore, this activity results in a difference between actual and expected result. Software Testing refers to finding bugs, errors or missing requirements in the developed system or software. So, this is an investigation that provides the stakeholders with the exact knowledge about the quality of the product.

Software Testing can also be considered as a risk-based activity. The important thing during testing process the software testers must understand that how to minimize a large number of tests into manageable tests set, and make wise decisions about the risks that are important to test or what are not [1]. Figure 1 shows the testing cost and errors found a relationship. The Figure 1 clearly shows that cost goes up dramatically in testing both types i.e. functional and nonfunctional. The decision making for what to test or reduce tests then it can cause to miss many bugs. The effective testing goal is to do that optimal amount of tests so that extra testing effort can be minimized [1].

According to Figure 1, Software testing is an important component of software quality assurance. The importance of testing can be considered from life-critical software (e.g., flight control) testing which can be highly expensive because of risk regarding schedule delays, cost overruns, or outright cancellation [2], and more about this [3][4].

Corresponding author: Mr. Dheeraj

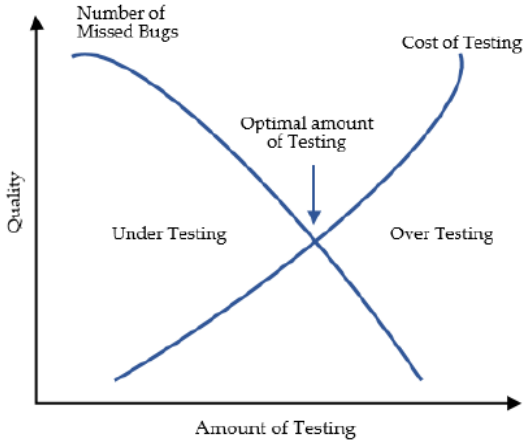

Figure 1: Every Software Project has optimal test effort [1].

Testing has certain levels and steps according to which the person who does the testing differs from level to level. The three basic steps in the software testing are Unit testing, Integration testing and System testing. Each of these steps is either tested by the software developer or the quality assurance engineer who is also known as a software tester [5]. The testing mentioned above steps is inclusive in the Software Development Lifecycle (SDLC). It is essential to break the software development into a set of modules where each module assigned to a different team or different individual. After the completion of each module or unit, it is tested by the developer just to check whether the developed module is working by the expectation or not, this is termed as Unit Testing. The second step of testing within the SDLC is Integration Testing. Once the modules of a single software system have been developed independently, they are integrated together and often errors arise in the build once the integration has been done. The final testing step in the SDLC is System Testing, which is testing of the whole software from each and every perspective. Also, software testing ensures that the integrated units do not interfere or disturb the programming of any other module. However, testing of a large or intensely complex systems might be an extremely time-consuming and lengthy procedure as the more components within the application, the more difficult it gets to test each combination and scenario, consequently leading towards a dire need for enhanced software testing process for premium optimization [6].

Testing cycle is mainly composed of six phases, from Test Planning to the analysis of Test Results. Requirement Analysis is the first phase in the SDLC and signifies that 80% of defects can be typically attributed to requirements, Standard approaches to requirements testing & analysis: Walk-throughs, Graphical aids, Modeling tools. Test Planning being the second phase is mainly the plan of all the test activities that are to be conducted in the whole testing process. Test Case Development is the third phase of the testing life cycle, where the test cases that are to be used in the testing process are developed. A detailed test case life cycle involves test case generation, test case selection, test case minimization, test case prioritization and test case evaluation [21]. There are techniques and tools used in each phase of test case life cycle. Some of the techniques are based on UML diagrams, Dynamic Path Testing, Evolutionary Technique, Graph Traversal Algorithms, genetic algorithm and Critical Path Method.

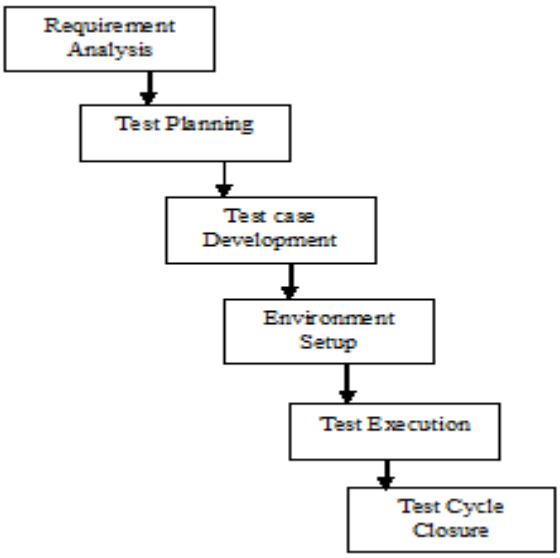

Figure 2: Software Testing Life Cycle.

Next phase is the Test Environment Setup which decides the software and hardware conditions under which a work product is tested. Test execution is the next phase of the Testing cycle that encompasses the execution of the tests cases, and the relevant bugs are reported in the next phase that is the Test Reporting phase. Test Result Analysis is the last stage of the testing process in which the defect analysis is done by the developer who developed the system or the software, this step can also be handled along with the client as it will help in the better understanding of what to ignore and what exactly to fix or enhance or simply modify [7].

© 2017 IJETCR. All Rights Reserved.

2. EXISTING TESTING METHODS

For the commencement of the Testing process, the first step is to generate test cases. The test cases are developed using various testing techniques, for the effective and accurate testing. At a high level testing types could be classified as Functional Testing, Performance Testing and Security Testing [22]. The major testing techniques are White box testing, Grey Box testing and Black Box testing [8]. White Box testing is significantly effective as it is the method of testing that not only tests the functionality of the software but also tests the internal structure of the application. While designing the test cases to conduct white box testing, programming skills are requisite to design the test cases. White box testing is also called clear box or glass box testing. This kind of testing can be applied to all levels including unit, integration or system testing. This type of testing is also called Security Testing that is it fulfills the need to determine whether the information systems protect data and maintains the intended functionality.[9][10]
Black Box testing is a testing technique that essentially tests the functionality of the application without going into its implementation level detail. This technique can be applied to every level of testing within the SDLC. It mainly executes the testing in such a way that it covers each and every functionality of the application to determine whether it meets the initially specified requirements of the user or not. It is capable of finding incorrect functionalities by testing their functionality at each minimum, maximum and base case value. It is the most simple and widespread testing process used worldwide [9] [10].

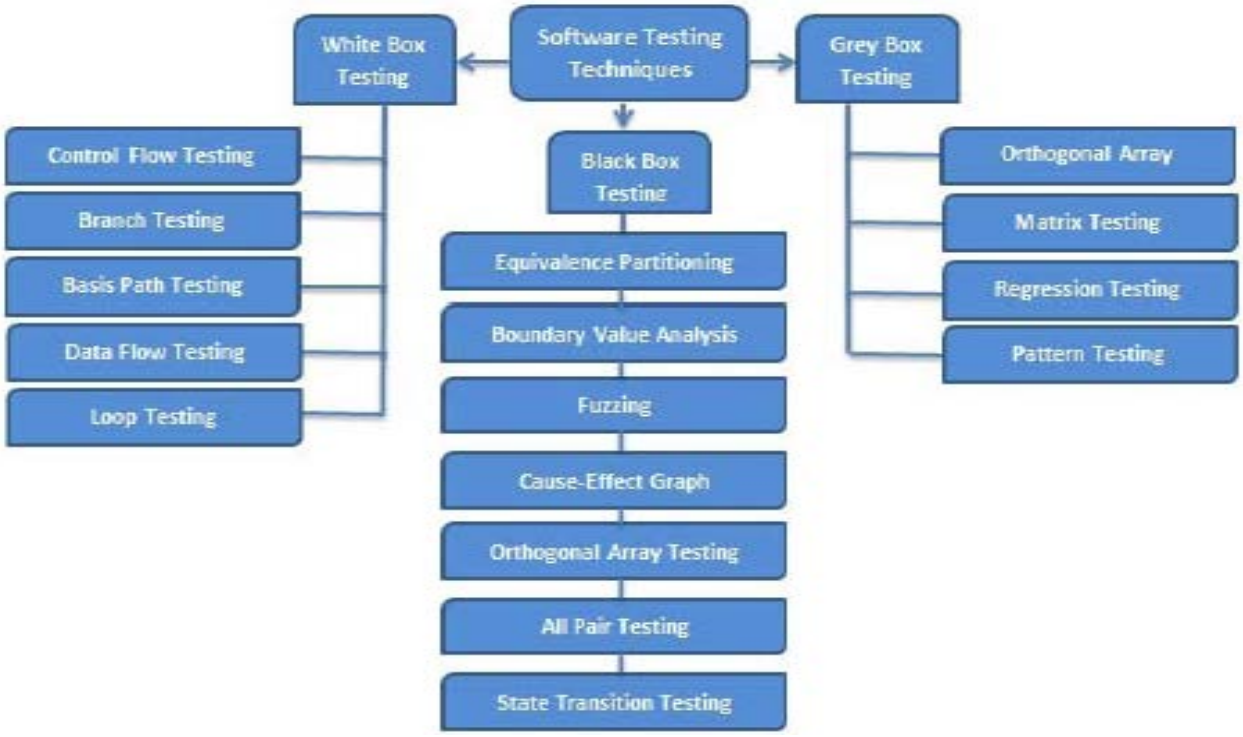

Figure 3: Software Testing Techniques [8].

Grey Box Testing is the combination of the White Box and Black Box Testing Technique serving the advantages of both. The need for such kind of testing aroused because in this type of testing the tester is attentive of the inner configuration of the application, hence testing the functionality in a better way taking the internal structure of the application into consideration. Figure 2 is referenced from author J. Irena [8].

3. ENHANCEMENT IN TESTING PROCESSES

Test Suite Prioritization does enhancement in the testing process by Combinational Criteria. The major methodology behind such test case prioritizing is the conversion of the weblogs into the test suites relevant with the user session, and further writing it down into an XML format. The Algorithm used for this approach should be accurately prioritized by the coverage based on combinatorial test suites.

© 2017 IJETCR. All Rights Reserved.

Moreover, empirical studies should be carried out for analyzing the effectiveness of the specific application and its relevant test suites [11]. A tool used in this regard is known as C-PUT which essentially formats the logs of the web applications into test suites which are formatted in XML; it is then used for the provision of the functionality for the prioritization of these tests. There is an ongoing research about if these test suite prioritization techniques can be used to enhance the fault detection ratio or not [12]. The usage of genetic algorithms (GAs) for the purpose of automated test data generation for testing the application is yet another enhancement in the testing process, as previously the dynamic means of test data generation remained a big issue in the software testing process, so the usage of Genetic Algorithm based testing is an effective of the test data generation, it also capable of handling the data generation keeping in line with the complexity of program.

A. Test Automation

The major enhancement in the testing process leads the testing process towards the Test Automation, which is the use of particular software to carry out the testing process as well as it makes the comparison of actual results with the expected results. Test Automation technique is time effective, as it saves the time of manual testing which can be quite laborious. In SDLC, Test Automation occurs during the implementation as well as the testing phase. Throughout the world, Test Automation is being practiced instead of manual testing as it saves a great amount of time accomplishing the testing processes in shorter time span. Test automation has taken over the manual testing process by reducing its need as well as by exposing the amount of errors, shortfalls that cannot be acknowledged via the manual testing process.

Regression testing being one of the major testing types requires much time when done manually. It typically tests whether the software or the application works properly after the fixation of any bugs or errors. Because sometimes after the error fixation, the code or application's error or bug ratio gets even higher. So, for the avoidance of the time taken for regression testing; a set of automated test suites is made to form a regression test suite for such purpose. Test Automation also helps in finding the problem at the much earlier stage, saving heaps of modification cost and energy at later stages [13].

Keyword driven testing Framework utilizes self-explanatory keywords which are termed as Directives. Such type of framework is used to explain the actions that are expected to be performed by the software or application that is to be tested. This kind of testing is a basically extension of Data Driven Testing as the data as well as the directives are kept in separate data files. It encompasses all advantages of the data-driven testing framework. Also, reusability of the keywords is another major advantage. The ill factor of this kind of testing framework is that due to the usage of keywords, it adds complexity to the framework making test cases longer and more complex.

4. TESTING METRICS

A. Prioritization Metrics

The usage of Test Metrics has prime significance as they can enormously enhance the effectiveness of the testing process. They serve as an important indicator of the efficiency and correctness and analysis of defined metrics. They can also help in the identification of the areas which require improvement along with subsequent action or step that needs to be taken to eliminate it. Test Metrics are not just a single step in STLC but acts as an umbrella for the constant improvement of the whole testing process itself .Software Testing Metrics focus on the quality facets relevant to the process and product and are categorized into Process Quality Metrics, and Product Quality Metrics both of whims aim to provide enhancements in not only the testing process but also in the product quality. However, there lays a critical issue faced by the existing testing process which is matching of the testing approach with the application being developed. Not every testing approach can be implemented in every application to be developed. For example testing of a network protocol software as compared with the testing of certain e-commerce application will be quite different with completely different test cases complexity, and that outlines the criticality of human involvement within the testing process and not just mere reliance on the existing test cases. Prioritization Metrics include the length of the test based on some HTTP requests within a test case. Frequency based prioritization enhances the testing process such that the test cases that encompasses most used pages are, selected for execution before those test cases that utilize less frequent ones [14].

B. Process Quality Metrics

© 2017 IJETCR. All Rights Reserved.

A process is the most eminent part as it is capable of producing a quality outcome within the least time in the most cost effective manner. This is the ultimate reason that why organizations throughout the world have put their focus on the enhancement of the processes performance, and this exactly where the need for the metrics emerged, as it is required to gauge the process from various dimensions efficiently. Measuring Efficiency of the process is the key metric of process quality which encompasses certain measurements of factors like Test progress Curve which depicts the planned progress of the Testing Phase by the test plan [15].

The cost of Testing is the next major step of the metric both phase wise and component wise. The major objective of which is to help in identifying the parts that require intensive testing and cost that they will bear according to it. Average Defect Turnaround Time is another metric which depicts average verification time by the testing team for the verification of the defects. Average Defect Response Time is the metric that is an indicator of the operational efficiency. It is the measure of average time taken by the team for responding to the errors. Metrics for Process Effectiveness ensures that the resulted application or products will yield a high-quality output. Test coverage, Defect Removal Efficiency, Requirement Volatility Index, failed and executed test cases being major categories of it ensuring an overall enhanced Testing Process.

5. AUTOMATED TEST INPUT GENERATION [20]

The generation of the test data is not the new topic of research, before 2000 sufficient amount of work has been done on the test data generation, while the running time has gone through with the lots of research join the following topic and shown interesting results and contribution to the field. Where the results shown in the field can be split into parts as the improvement of the power of processing and also the improvements in the different platforms used for the computation. The improvement in the field is just due to the splendid work done by the researchers in the field which also includes the advancement in the research field and the different technologies supporting the research area, such as symbolic execution, search-based testing, random and fuzz testing, and combinations thereof.

A. Symbolic Execution

Advances in symbolic execution are one of the main reasons automated test input generation has become more relevant. King in 1976 proposed a technique for the analysis of the program as static symbolic execution. Different symbolic inputs are being used in the static symbolic execution instead of going for the concrete inputs for executing the program. The state of the program at point of time is being represented as the state of symbol expression which is the function of the inputs, different sets of constraints in the form of the conjunction are being used so as to express the different conditions on input and the conditions are termed as the path condition (PC). More formally, the symbolic state can be seen as a map S: M 7! E, where the M depicts the programs memory address, symbolic values which are possible are represented by E, that is, expressions in some theory T such that all free variables are input values.
At the time of the symbolic execution the symbolic state and the path conditions are generated, true is being assigned to the PC, inputs are represented using the symbolic variables, and the language semantic is being used for the initialization of the S. At the time when the statement stmt modifies the memory address m of the program for that the new symbolic value e0 for mis is computed on the basis of the semantics of the statements stmt's, and S is updated by replacing the old expression for m with e0 (S0 = S _ [m 7! e0], where _ indicates an update). At the time of execution of the predicate statement pred which actually changes the control flow, both branches are being followed by the forks of execution. An additional conjunct is being augmented in PC on every branch which are used to express the conditions for input, shown as symbolic state, that makes the predicate in pred true or false (depending on the branch)[16].

B. Search-based Testing

While symbolic execution techniques received the largest number of mentions in our colleagues' responses, where test input generation techniques are concerned, The test data generation technique which are search based are second in most number of mentions in research area, and are termed as the search based software testing (SBST). Different surveys for the search based software testing techniques is being provided by the Harman and colleagues which is based on the use in software engineering in general. (Several other surveys are also available, including [2, 3, 6]

© 2017 IJETCR. All Rights Reserved.

They point out that a majority of papers on the use of SBST (54%, considering papers through the end of 2008) address topics in test input generation, and that the number of papers on search-based techniques has been increasing regularly, and quite rapidly, ever since the year 2000. They also cite several instances in which industrial organizations such as Daimler, Microsoft, Nokia, Ericsson, Motorola, and IBM have considered the use of SBST techniques. In general, search-based techniques target optimization problems such as (in the area of testing) finding the smallest set of test cases that cover all the branches in a program. They do this by employing meta-heuristic search-based optimization techniques, which seek good solutions from among a space of candidate solutions, guided by fitness functions that can differentiate and rank such solutions.

C. Random Testing

Another test data generation technique which is quite matured in past years is the random testing (RT) which is a automated test input generation technique other than the dynamic symbolic execution and different search based techniques. Rather going for the random test data generation which is quite straight forward the researchers have gone for the other techniques investigating more sophisticated, and to some extent principled, approaches that can improve the effectiveness of this traditional technique. This increase in effectiveness is achieved by defining techniques that can either improve the random input generation process or manage the often overwhelmingly large number of test inputs generated. One example of these new random-testing approaches is adaptive random testing. Adaptive random testing (ART) [17] is testing methodology which is being used for improving the efficiency of the random testing as the test inputs used in the program execution are diverse in nature in the domain of the input. Number of test input candidates are being generated first in ART technique which are generated as random, for creating more test inputs while executing the program under test. The next candidate is being selected as the next input candidate should be most “distant” from the previous input executed and discarding the rest of other available test candidate inputs. Various studies have shown that ART techniques can require substantially less number of test inputs as compared to the other random techniques so that the failure of the program can be revealed. The ART for the test data generation is disadvantageous as it fails to handle the complex data format and also the overhead of the overall methodology is quite high [10]. The new methodology is being proposed which is based on the concept of mirroring to overcome the previously defined problem, forgetting, and Voronoi tessellation [18].

D. Combined Techniques

For the test data generation in the previous years the researchers have also worked for combining the techniques so as to obtain the better results, the researchers have also worked for combining different techniques with different verification techniques. With the work on the combining of the different techniques the work also goes for merging the techniques with different verification techniques as static and dynamic verification techniques. Yogi project ongoing at the Microsoft Research is the better representation for this type of work, which seeks to combine testing, which under-approximates program behavior (i.e., can produce false negatives), error discovering is quite effective and adopts the static verification technique, which is complete but over approximates program behavior (i.e., can produce false positives), in order to leverage their strengths while reducing their weaknesses. The test data and the abstraction are being combined using the Synergy and Dash algorithm for working synergistically and also iteratively. The abstraction are being refined by the test data, and the new test data is being generated using the abstraction [19] [20].

6. CHALLENGES AND OPPORTUNITIES [20]

A. Testing Modern, Real-World Systems

A lot of testing techniques especially those proposed by the academicians which are still to get name at the industrial level, target traditional software- class of software which are written in a single language and are homogenous in nature, non-distributed and also are small in size. The software system that are existence are quite different from the traditional software’s and just due to the characteristics of those they are not fit for the current testing techniques. The software system used now a days are diverse in nature and are in different components and are also provenance, where the coupling with one another is different, they are dynamic in nature and are also distributed. Numerous classes of applications that are

© 2017 IJETCR. All Rights Reserved.

increasingly popular, such as mobile applications, web applications, software product lines, service-oriented architectures, and cloud-based applications, have these characteristics.

B. Oracles

Long ago recognized as a significant problem, the oracle problem is being defined as the problem in computing the efficiency of the behavior of the program which is under test-is still relevant today. The test oracles active works are the recent criterions relatively, and many of the authors have focused on the importance of the oracle problem while the evaluation of the test efficiency of the overall testing methodology [6]. (A recent survey by Harman and colleagues provides a comprehensive discussion of the state of the art in test oracle research.) The oracle construction problem is still open largely in automated and semi-automated fashion and some initial work is being going on this direction.

C. Probabilistic Program Analysis

Probabilistic program analysis is the research area which has attracted the researchers in the recent years and considerable amount of research is going on it and in coming time will put an impact on the field of testing and the test data generation process. In probabilistic program analysis the approach goes beyond the levels of the tradition testing techniques which actually tries to find out that the defined behavior of the program is possible or not. Probabilistic program analysis, conversely, aims to quantify the probability of an event of interest, such as the failure of an assertion, to occur. The researchers have also used the idea of the probabilistic program analysis in the technique of the symbolic execution. As in the case of the traditional symbolic execution technique actually tries to find out the path in the program and decides whether the decided path is feasible or not on the basis of the constraint solver, the probability is being assigned to each path in the symbolic execution technique using the probability analysis.

D. Testing Non-Functional Properties

The functional correctness of the program is the point of focus for the researchers now a days, which is termed as the debug testing in some of the quarters. There present several other properties of the program which the engineers wish to have while testing the program, from which very few properties have attention while testing as seen in the testing methodologies literature. Performance was the most cited property of the program by most of the researchers. The issue of performance is there in almost all software system but in the case of some it is quite critical part to play. For example the performance matters almost everything if we talk about the web applications and web services where the response time is considered as the key element, for analyzing the performance of such type of software systems there are available lots of tools as open source and also the commercial tools. With the increase in the use of the mobile phone the importance of the performance of the real time software systems has great concern.

E. Domain-Based Testing

In the case when new technology successfully comes to the market the mint set of researchers turns towards that technology. Because which it is being seen that the researchers goes with their research by considering the new technology like component based systems, web applications, mobile applications. In the field of IT there always will be need for addressing the new emerging technologies and applications, which provides greater opportunities for the researchers in testing field. Dynamic multi-tier web applications are being considered by the researchers in there techniques for testing, which is quite new in the field and may have the case of argument for the testing product lines and testing android applications.

F. Leveraging the Cloud and the Crowd

In the last decades the cloud is the most pervasive field of research. The cloud services are providing the facility for the users to shift from the hosting of the computational power on the local machine and also the storage to the remote machines and using of the data centers and at the same using the computational power of the remote servers. Some short of research is being ongoing for using the cloud in the field of software testing. Cloud IDEs (Integrated development Environment) is the best common example for that work in using the cloud in the field of software testing. As the testing of the software is one of the expensive part of the overall development and the IDEs might work for making the things easy in the field of the software testing.

7. CONLCUSION

Testing is the most critical part of the Software Development Lifecycle, as it is something on which

© 2017 IJETCR. All Rights Reserved.

final delivery of the product is dependent. It is time consuming and an intensive process, therefore, enhanced techniques and innovative methodologies should be used. There is need of automated testing to cover all dimensions of software including Functional requirements, Performance parameters to ensure it is suitable to bear maximum load, Security aspects to ensure it doesn't get hacked and data breach don't occur. Another most important area is to ensure application is easy to use, so there is need to work on usability testing as well. A focused test approaches needs to be designed to cover all these areas collectively to ensure better quality of software. It is required to enhance existing testing tool suites and methods, both for time effectiveness as well as for efficient and reliable final product which not only meets the specified requirements but also provides maximum operational efficiency.

REFERENCES:

1. P. Ron. Software testing. Vol. 2. Indianapolis: Sam's, 2001.
2. S. Amland, "Risk-based testing:" Journal of Systems and Software, vol. 53, no. 3, pp. 287–295, Sep. 2000.
3. Redmill and Felix, "Theory and Practice of Risk-based Testing", Software Testing, Verification and Reliability, Vol. 15, No. 1, arch 2005.
4. B. Agarwal et al., "Software engineering and testing". Jones & Bartlett Learning, 2010.
5. K. Bogdan. "Automated software test data generation". Software Engineering, IEEE Transactions on 16.8 (1990): 870-879.
6. Jacobson et al. The unified software development process. Vol. 1. Reading: Addison-Wesley, 1999.
7. Everett et al., "Software testing: testing across the entire software development life cycle". John Wiley & Sons, 2007.
8. J.Irena. "Software Testing Methods and Techniques", 2008, pp. 30-35.
9. Guide to the Software Engineering Body of Knowledge, Swebok, A project of the IEEE Computer Society Professional Practices Committee, 2004.
10. E. F. Miller, "Introduction to Software Testing Technology", Software Testing & Validation Techniques, IEEE, 1981, pp. 4-16.
11. S. Sampath and R. Bryce, Improving the effectiveness of Test Suite Reduction for User-Session-Based Testing of Web Applications, Elsevier Information and Software Technology Journal, 2012.
12. S. Sprenkle et al., "Applying Concept Analysis to User- session based Testing of Web Applications", IEEE Transactions on Software Engineering, Vol. 33, No. 10, 2007, pp. 643 – 658.
13. Memon, "A Uniform Representation of Hybrid Criteria for Regression Testing", Transactions on Software Engineering (TSE), 2013.
14. R. Bryce, "Test Suite Prioritisation and Reduction by Combinational based Criteria", IEEE Computer Society", 2014, pp.21-22.
15. R. Ramler, S. Biffl, and P. Grünbacher, "Value-based management of software testing," in Value-Based Software Engineering. Springer Science Business Media, 2006, pp. 225– 244.
16. J. C. King. Symbolic Execution and Program Testing. Communications of the ACM, 19(7):385–394, 1976.
17. T. Y. Chen, T. H. Tse, and Y. T. Yu. Proportional sampling strategy: A compendium and some insights. Journal of Systems and Software, 58(1):65–81, 2001.
18. T. Y. Chen, F.-C. Kuo, R. Merkel, and S. P. Ng. Mirror adaptive random testing. Information and Software Technology, 46(15):1001–1010, 2004.
19. N. E. Beckman, A. V. Nori, S. K. Rajamani, and R. J. Simmons. Proofs from tests. In Proceedings of the 2008 International Symposium on Software Testing and Analysis, pages 3–14, 2008.
20. Alessandro Orso, Gregg Rothermel, "Software Testing: A Research Travelogue", FOSE'14, May 31–June 7, 2014, Hyderabad, India.
21. Itti Hooda, Dr. Rajender Chhillar, "A Review: Study of Test Case Generation Techniques", International Journal of Computer Applications (0975 –8887) Volume 107–No. 16, December 2014
22. Itti Hooda, Rajender Chhillar, "Software Test Process, Testing Types and Techniques "International Journal of Computer Applications (0975 –8887) Volume 111 –No 13 February 2015.

© 2017 IJETCR. All Rights Reserved.