WEB Based Applications Testing: Analytical Approach towards Model Based Testing and Fuzz Testing

[1]Umer Zaheer Khan and Dr. Salman Afsar

***Abstract*- Web based applications are complex in structure which results in facing immense amount of exploiting attacks, so testing should be done in proactive way in order to identify threats in the applications. The intruder can explore these security loopholes and may exploit the application which results in economical lose, so testing the application becomes a supreme phase of development. The prime objective of testing is to secure the contents of applications either through static or automatic approach. The software houses usually follow fuzz based testing in which flaws can be explored by randomly inputting invalid data while on the other hand model based testing is the automated approach which test the applications from all perspectives on the basis of abstract model of the application. The main theme of this research is to study the difference of fuzz based testing and model based testing in terms of test coverage, performance, cost and time. This research work guides the web application practitioner in selection of suitable methodology for different testing scenarios which save efforts imparted on testing and develop better and breaches free product.**

Keywords: ESG, MBT, SUT, TOE, UML.

I. Introduction

Web applications are the key part of today's world of internet where users perform different kinds of activities via remote device with ease such as online transactions, reservation of tickets for flights etc. This trend of services are now widely utilized by every organization such as banking sectors which manages a lot of daily transections and transport sector for issuing tickets etc., this trend looks easy and calm from end users perspective but it arises questionable debate when we talk about privacy and confidentiality of information that is, how much it is secure and what sort of parameters are followed in order to sure its safety?. Unfortunately, the web application didn't get much trust and satisfaction from users in context of safety of information and this put a question mark on its ability of protection and security of the information of an organization, providing the services through that application. In order to understand these factors, we need to understand the simple workout of banking sectors which utilizes this platform to manage their daily-based activities like transferring and depositing fund, balance inquiry, transections history withdrawal and so on. All

[1] Raiumer23@gmail.com

these details are stored in centralized storage called database which linked with server of that bank. Now, if hacker accesses the server side's script or even if he able to crack the database of that application, then surely the whole system is under custody of that person and he able to steal those sensitive credentials of customers and bank and put both of them on the edge of forfeiture like transferring bank balance of customer to another account without prior to his and bank managements knowledge. Such uncertainties put both stakeholders and users at the edge uncertain risk; the primary factors of such uncertainties are actually come from customer's side when they continuously requesting for changes that in turn intended the developers to more be focused on user's modifications instead of prioritize security structure [11].

The secondary factor of susceptibilities and flaws are triggered due to flaws in applications designs and code which can be undone by following quality of testing experiments where several behavior aspects of applications are being analyzed in context of quality and security which verifies whether the application fulfills the properties of intended behavior or not as it makes sure that application remains in safe and quality shells and also shows the power of quality testing. Yes! The fact is confessed that it quite tired procedure as it demands a lot of effort and cost to ensure the applications compliance because most of the testing is done by following old-fashioned approaches instead of automatic testing which ensures the safety of application by consuming less time and effort yields betters results as compared to manual approaches.

Model based testing is increasing broad significance because of speedier and creation of test scripts for verifying application's security and reliability. This approach is seems to be a semi-automatic kind of approach that generate test cases on the basis of models where these models are actually action and events of the system that reflects the functionality and intended behavior of the system and are abstracted from set of requirements. These models are created manually or by the help of software's. Due to swift and complexity in the development industry, there comes a supreme task of testing application full-fledged at earliest possibility. Model based testing is type of black box testing because in such condition the test cases is obtained from model that are designs for system under test. These test cases are then packed in the form of test suits and executed on the system in systematic series. The priority reason to select model based testing is that its principle objective is to automate manual based activities by minimizing the cost of generating models for coverage and to reduce the efforts for designing and implementation of test cases [20].

While, the fuzz technique targets the application in order to verify its conformity and reliability by exploring important security loopholes in the application. The concept behind this approach is to act like an intruder in order to target the system by continuously inserting testing data that compose of malicious inputs and then analyze the system by monitoring it behavior. Actually fuzzing was initially used to find zero days in black-hat community. The main concern of this approach is to generate data for testing that able to strike the victim application or at least create a negative impact on the execution of the application and then monitoring the outcome. Although there are many others alternates available to explore hidden vulnerabilities but fuzz is more popular at industrial level as it does not required the source code and does not have blind spots to like human testers, and also cost saving technique than manual approaches [10].

II. REVIEW OF LITERATURE

Testing is actually an estimation of application applicability by observing its execution. Testing involves dynamic validation of tangible behavior of the system against its predictable behavior. This is achieved with a determinate set of test cases termed as test suites appropriately designated from the infinite set of anticipated traces of execution. The behavior of application is examined by smearing invasive tests that stimulated the application and then evaluate the system responses by observation. After execution of test cases, the actual and anticipated behavior of the application under test is compared with each other which results in conclusion, where the test oracle is used as mechanism for determining conclusion. The conclusion can either be pass, fail or inconclusive. The pass conclusion suggests that application's behavior is conformed while the fail and inconclusive conclusions suggests that application is either not conformed or unknown respectively [6].

The procedure of model based testing by stated that it's quite similar for every application that is first developing models for system which are under examined and then translate these models into test modeling and finally generate test cases which are executable on system based on coverage criterion. Though, implementation of model based testing to a new industrial application always faces tedious difficulties because of practices follows at industrial level are quite complex and need of proficient tools and category of the SUT [1].

The experimental study estimate the usage of model based testing in creation of models, their generation, and accomplishment of automatic test cases in context of mobile apps. However, they followed ESG (Event Sequence Graph) technique to design the artifacts of test models in order to define requirements and features of mobile apps which are under examination of testing procedure.

Their focus is on mobile apps, so they employed testing cases using a framework called Robotium. The results after analysis favors the model based approach and shows that this approach works well in terms of test case generation and ability to detect faults and flaws in application and the same time it enhanced the quality of test cases while reducing time and cost for testing and development of test models. Although there are supreme challenges might be occur during this technique that are problems of test modeling, concretization of test in context of mobile applications and the need of proficient tools. The experimental study concluded that Model based testing along with Event Sequence Graph modeling can be used as an efficient approach to test the applications based on android operating system [7].

Fuzz testing technique is actually based on comprehensive assessment. It is different from traditional way of fuzz testing that follows the concept of blind injection, this method divides the input into various fields by executing the dynamic assessment and make a rank for each field on the base of comprehensive assessment results in detecting vulnerabilities more quickly as compare to old ones. Experimental analysis shows that the field division is more effective as it makes fuzz testing approach more efficacious and coherent [10].

The study presented a methodology based on fuzzy logic with collaboration of model based methodology in order to rank test cases by means of information accessible from the symbolic tree's execution that acquired from a model. The responses to the fuzz based logic system are test suits, symbolic tree's execution dimensions, and comparative test case size. The fuzz based logic system yields particular crispy output termed as importance for each test case. The test cases are ordered on the basis of crisp output. This methodology presumes that the information covered from test cases at the model-level is accessible and uncertain to follow those models which are at higher abstraction level [17].

In order to explore security flaws in services of webs, the processes of the target system need to be thoroughly observed on the behalf of fuzz approach. The presence of susceptibilities in target system are recognized and evaluated based on the consequences achieved after monitoring. They also discussed the process of fuzz testing by stated that common methods of monitoring comprises observation analysis, following via debuggers, and dynamic binary composition. After that, a regular request is presented after sending a series of abnormal tests to the target system and the status of operations can be examined by the analysis of the responses of the systems for the regular or typical requests [10].

III. Procedure

The research work is comprised of three sections: - Survey Experiments and proposed framework. The survey was mandatory part of research and data is collected by conducting survey which generates data that used for further analyzed by the help of statistical tools. While the experimental work dynamically validates the findings of survey results, however the least portion discuss the framework that based on the strengths of two above discussed techniques.

A. Survey Results

The respondents believe that model based testing works much better as semi-automated technique as it comes with more quality and reliability in context of applications testing while fully automatic suited more when deadline is little far away from the end as shown in Figure. 1.

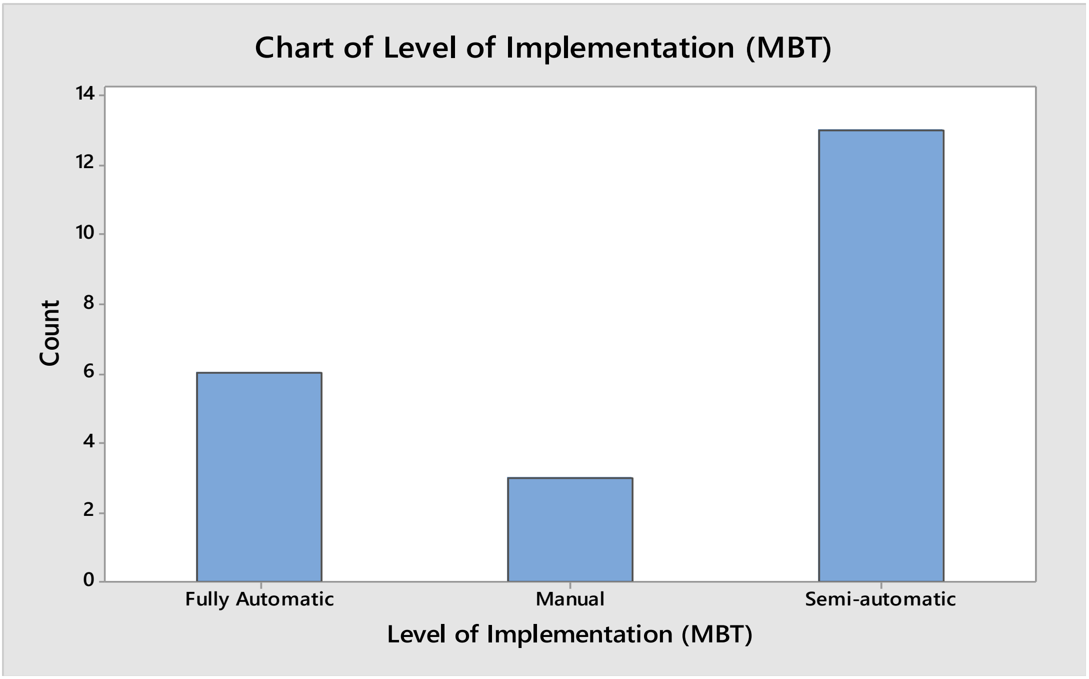

Figure 1: Level of MBT

While the survey results conclude that model based testing is best suited for testing functional requirements as it requirements are the raw information in order to develop models which further generated test cases on the basis of these models. If the raw information is available for nonfunctional requirements, then it somehow test them well but not up to that standard mark as shown in Figure. 2.

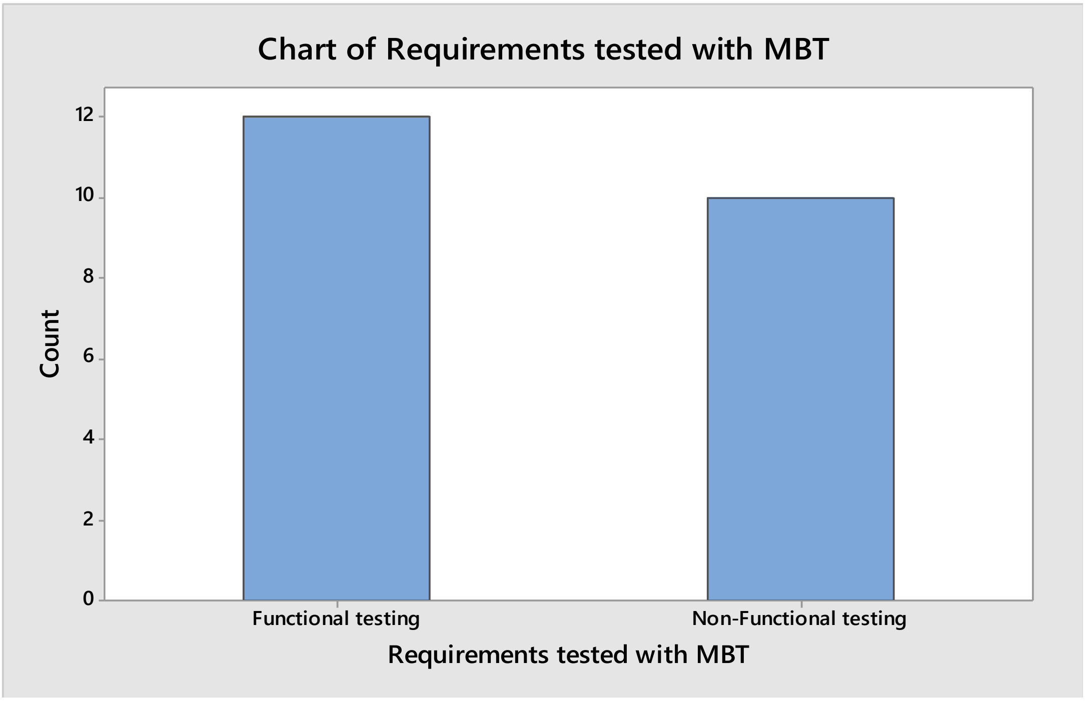

Figure 2: Requirement level in MBT

The model based testing is moderately cost effective, as it requires development of models and second, its best tools are available at commercial cost but it somehow reduces cost in terms of efforts as well as moderately effective in terms of time, as it requires development of models and test cases to generate which at the initial phase demands more time than other techniques. Where is the fuzz testing is very cost effective, as it requires test data and second, its best tools are available at lower cost and very effective in terms of time, as it requires simple testing inputs related to target application and then execute these inputs and verifies the behavior of the application. It also very efficient terms of execution as it simply requires a set of inputs and tools to insert these into target applications.

There are two ways to see the test coverage, one is as strength of Fuzz testing and other is model based testing. The coverage is solely depending on the experience of the tester, the one who is writing that particular test and in the case if the test cases are not written by experience tester, then it should be approved by experience testers. Also note that in model based testing, the test coverage is always higher than Fuzz testing due to the fact that the cases is derived by examine the test coverage.

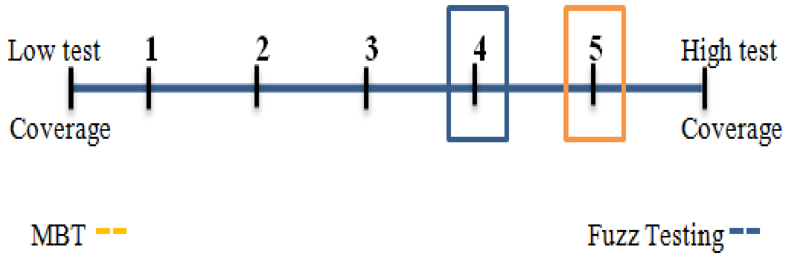

Figure 3: Test coverage of MBT and Fuzz

By consideration of test coverage and taking the help of average score of fuzzy testing and MBT the Figure.3 was created. It indicates that Fuzz testing has comparatively maximum test coverage then is path, branch, and statement coverage, it has also observed that due to zero- tolerance towards the test coverage the model based testing shows maximum coverage. The requirement traceability is one of strengths of model based testing. As per the data of survey, the requirement can be traceable through various ways by the test cases. However in MBT, the traceability is done in a different way. According to the respondents, it is a great challenge for companies and MBT to trace the requirement often it difficult in MBT approach to track the results back to the system requirements. Currently, remarkable studies related to finding out more appropriate way to make requirements traceable in MBT process have been done. According to the survey data analysis, figure 4 shows the requirement traceability in Fuzz testing and MBT.

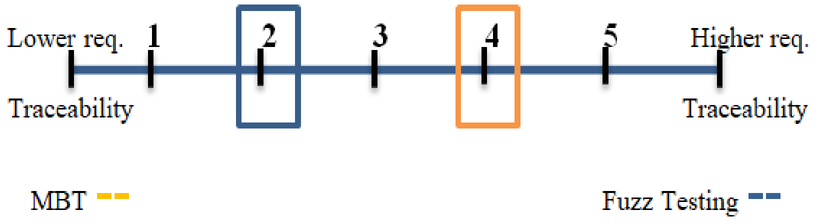

Figure 4: Requirement Traceability in MBT and Fuzz

B. Experiment Results

The comparison of test coverage results of the test cases generated from both approaches (MBT and Fuzz testing) are presented in Table 1. The test cases written by Fuzz group – 1, showed that there are two test cases required to test all conditions which sorted out during the analysis and design phase, and also two test cases are required to cover all the statements i.e. statement coverage and total of four test cases were required to test all possible paths i.e. path coverage. Similarly from Fuzz group – 2 test data, the branch, path and statement coverage was 3, 3 and 1 respectively. From test data gathered from Fuzz group – 3, branch, path and statement coverage was 3, 4 and 1 respectively. The test data from the three MBT teams resulted in branch, path and statement coverage of (3, 3, 2), (3, 4, 2) and (3, 3, 2) respectively.

TABLE I

MBT Group	Model Based Testing	Test Coverage	Fuzz Based Testing	Fuzz Testing Group
Group 1	3	Coverage of Paths	4	Group 1
	2	Coverage of Statements	2	
	3	Coverage of Branches	2	
Group 2	4	Coverage of Paths	3	Group 2
	2	Coverage of Statements	1	
	3	Coverage of Branches	3	
Group 3	3	Coverage of Paths	4	Group 3
	2	Coverage of Statements	1	
	3	Coverage of Branches	3	

While, the cost calculated by giving each team "x" value as standard cost per minute. For calculation we have given a standard value which is 0.5 units per minute i.e. 30 units per hr. The cost was calculated with respect to time consumed by each team. The results that were calculated are mentioned in the Table II.

TABLE II

MBT Group	Model Based Testing	Cost and Time	Fuzz Based Testing	Fuzz testing Group
Group 1	35 unit	Cost	57.5 unit	Group 1
	70 unit	Time	11.5 unit	
Group2	45 unit	Cost	55.5 unit	Group2
	90 unit	Time	111 unit	
Group 3	37.5 unit	Cost	48 unit	Group 3
	75 unit	Time	96 unit	

C. Proposed Framework

The conceptual framework is proposed based on the strengths of both approaches. The definition of good framework is the one which is appropriate to indicate the several types of bugs depend on the three components:

I) The ability to develop models of the target application accurately termed as Target of Evaluation (TOE).

II) The ability to cover the behaviors of target application in a broad variety with the generated set of test patterns.

III) The ability to cover a wide range of vulnerabilities and flaws with the generated test cases.

However, the proposed framework is composed of three primary sections such as Target Application profile, modeling of data and algorithm testing which merges the strengths of both testing techniques in quite different way.

a) Target Application Profile

Before the fuzz testing is used or organized, the dynamic behavior of the target application that is termed as target of Evaluation (TOE) should be recognized and studied also involved the internal behavior of entities and the interaction behavior between them. The modeling for which UML state machine model is established that is based on the analysis of specifications either static or manual. Then applying selected graph traversal algorithm to the finite state model which generates a set of test patterns that represent the abstract behavior of the target of evaluation (TOE).

b) Modeling of Data

This phase enhances the code coverage, grammar-based data descriptors are followed to create the well-formed input vectors, which usually comes from the application-specific knowledge. The smart test vectors can pass the parameters error check near the code interface, and go inside the program. The XML descriptors are best suited to describe the structure of the input vector.

c) Test Algorithms

In this phase numerous generic based algorithms can be developed for different levels of vulnerabilities such as security flaws, buffer-overflows, coding bugs, integer-overflow type bugs and null-pointer type bugs etc. A set of executable test cases that usually large in size is created by the test schema which merged with the test vector and testing algorithmic modification. Furthermore, for fully automated testing, few other factors need to be considered, such as observation of the responses given for manipulated inputs and verify whether the target of evaluation took some unexpected action etc. The figure illustrates the working of the framework.

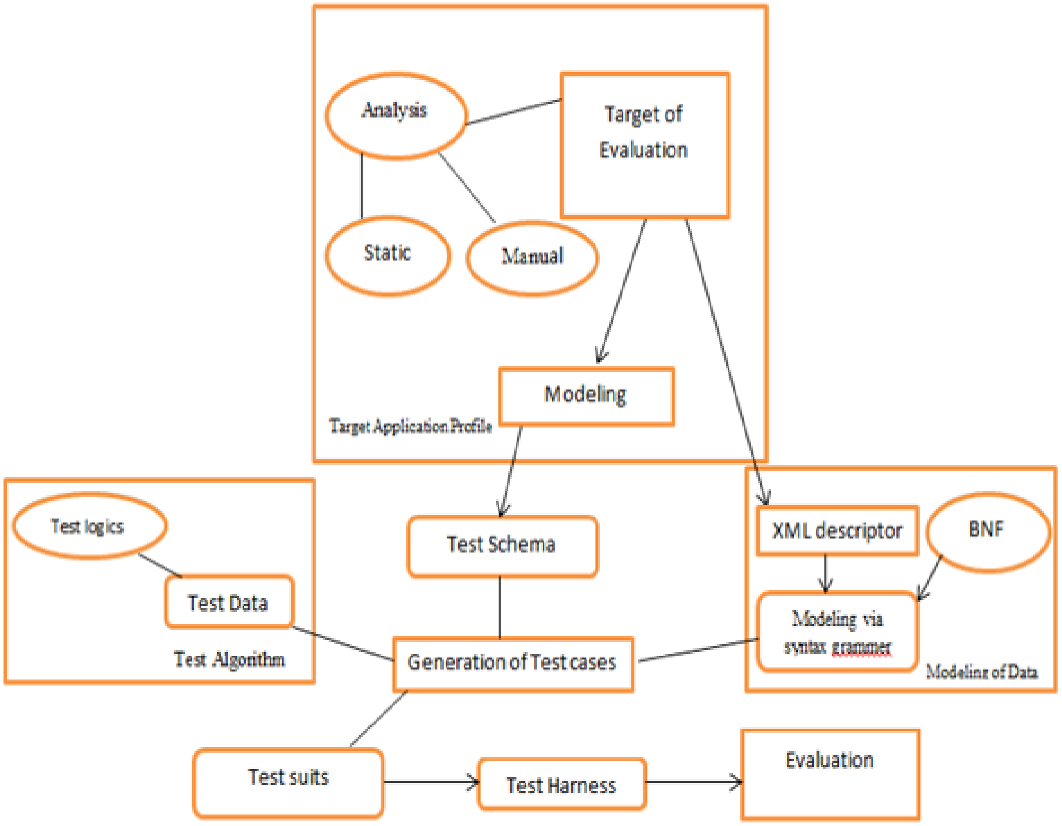

Figure 5: XMBFT FRAMEWORK

IV. CONCLUSION

The world of todays is totally based on internet, where tasks and operations are performed with in couple of minutes via remote access such as internet banking, seat reservation, or even online trading and shopping can be done via internet. The platform which is used for such activities are actually web based application which provide services to end users via remote devices by entering their secret credentials to login. The web application are actually a mixture of multiple programming languages such as JavaScript, AJAX and PHP which combine together to execute users operations. So, by keeping such concepts in mind, we must agree upon the fact that web applications should have a complex and tight security mechanism for securing users secrete credentials. But unfortunately the fact is against this concept, as security of websites are not still up to that mark where users feels that its credentials are saved on a secure place. The reasons behind these issues is that we still don't give value to testing and consider it as a last phase of development life cycle and follow the traditional and manual ways to test the application by neglecting the fact that how precious would be to test the applications at that time before they launched online for performing operations. The researchers and experts try to overcome these security issues by introducing advance techniques which executes automatically in order to test the application with in little time and cost, the examples of such techniques are model based testing, fuzz testing etc.

Model based testing is best considering to be a craftsman art where we have to focus on three import aspects such as understanding the target application, ability to establish accurate and precise models from raw information and ability to use the tools. This technique is good for discovering potential conflicts that causes the application to crash as this technique is automated that submits tests as input and executed it for specific period of time. Where, Synopsys report defines the fuzz testing as a valuable technology that used to uncover flaws and vulnerabilities in application by bombarding series of malformed inputs to a target application and then observe the spotted areas of the application for results. If the target application performs unexpected actions then examination of that failure is required, that examination uncover the root causes of that failure that may exploited for illegal purposes. Fuzzing plays a role of verification agent during implementation and deployment phases where undetected flaws may distress the integrity of the application. These techniques are approaches in analytical way in order to discover their strengths and weakness during web applications testing. After analysis the results concludes that model based testing is good in context of generating quality of test cases, requirement traceability then fuzz testing, but cost and effort required during fuzz testing is lesser as compared to model based testing, as model based testing requires a lot of time at initial level in order to analysis the raw information and to create application models while in fuzz we just need to develop data sets for application and then bombard them towards application. However it is true that model based testing is better in discovering vulnerabilities as compare to fuzz technique. But model based approach may halt where application structure is either too complex or there is no raw information in the form of requirements related to that application, then here fuzz testing comes and generate effective results to some extent. But we have to admired that, model based testing can cover large variety of scenarios with moderately little effort and random execution of models can expose those issues which are not easy to discover upfront such as design and specification issues. Where, the fuzz technique is good at discovering coding issues more accurately then other. But model based testing have some limitations too such as, if we need to test the application of large size then we need large set of random test cases of model based testing which requires a great deal of time and infrastructure. So we proposed a conceptual framework on the basis of their strengths which need further development and then require implementation in order to verify its outcome.

REFERENCES

[1] S. Ali, and H. Hemmati, "Model-based Testing of Video Conferencing Systems: Challenges, Lessons learnt, and Results," IEEE 7th International Conference on Software Testing, Verification and Validation (ICST), pp. 353–362, 2014.

[2] D. Amalfitano, N. Amatucci, A. R. Fasolino, U. Gentile, and G. Mele, "Improving Code Coverage in Android Apps Testing By Exploiting Patterns and Automatic Test Case Generation," International Workshop on Long-term Industrial Collaboration on Software Engineering, pp. 29-34. Vasteras, Sweden: ACM, 2014.

[3] I. Andrianto, I. Liem, and A. Y. D. Wardhana, "Web Application Fuzz Testing," International Conference on Data and Software Engineering (ICoDSE), pp. 978-984, 2017.

[4] M. Al-Refai, W. Cazzola, and G. Sudipto, "A Fuzzy Logic Based Approach for Model-based Regression Test Selection," International Conference on Model Driven Engineering Languages and Systems, pp. 55-62, 2017.

[5] W. Chunlei, L. Liu, and L. Qiang, "Automatic fuzz testing of web service vulnerability," International Conference on Information and Communications Technologies (ICT 2014), pp. 25-35, 2014.

[6] M. Felderer, P. Zech, R. Breu, M. Buchler, and A. Pretschner, "Model-based Security Testing: A Taxonomy and Systematic Classification," Software Testing, Verification and Reliability, pp. 530-560, 2015.

[7] C. F. Guilherme, A. T. Endo, "Evaluating the Model-Based Testing Approach in the Context of Mobile Applications," Electronic Notes in Theoretical Computer Science, pp. 3-21, 2015.

[8] A. M. Jamil, M. Arif, N. S. A. Abubakar, and A. Ahmad, "Software Testing Techniques: A Literature Review," 6th International Conference on Information and Communication Technology for The Muslim World (ICT4M), pp. 177-182, 2015.

[9] A. S. Kalaji, R. M. Hierons, and S. Stephen, "Generating Feasible Transition Paths for Testing from an Extended Finite State Machine (EFSM)," Software Testing Verification and Validation, pp. 230-239, 2009.

[10] C. Li, Q. Wei, and Q. Wang, "RankFuzz: Fuzz Testing Based on Comprehensive Evaluation," Fourth International Conference on Multimedia Information Networking and Security, pp. 939–942, 2012.

[11] L. Li, Q. Dong, D. Liu, and L. Zhu, "The Application of Fuzzing in Web Software Security Vulnerabilities Test," International Conference on Information Technology and Applications, pp. 130–133, 2013.

[12] A. Morozov, K. Ding, T. Chen, and K. Janschek, "Test Suite Prioritization for Efficient Regression Testing of Model-based Automotive Software," International Conference on Software Analysis, Testing and Evolution, pp. 20-29, 2017.

[13] C. Ma, and J. Provost, "A Model-Based Testing Framework with Reduced Set of Test Cases For Programmable Controllers," 13th IEEE Conference on Automation Science and Engineering (CASE), pp. 944-949, 2017.

[14] M. Mussa, S. Ouchani, W. A. Sammane, and A. Hamou-Lhadj, "A Survey of Model-Driven Testing Techniques," International Conference on Quality Software, pp. 167-172, 2007.

[15] T. Mouelhi, F. Fleurey, B. Baudry, and Y. L. Traon, "A Model-Based Framework for Security Policy Specification, Deployment and Testing Models," Lecture Notes in Computer Science, 537-552, 2008.

[16] H. Nikolas, "Efficient Fuzz Testing Leveraging Input, Code, and Execution," 39th IEEE International Conference on Software Engineering Companion, pp. 417-420, 2017.

[17] E. J. Rapos, and J. Dingel, "Using Fuzzy Logic and Symbolic Execution to Prioritize UML-RT Test Cases," 8th International Conference on Software Testing, Verification and Validation (ICST), pp. 52-62, 2015.

[18] N. Rafique, N. Rashid, S. Awan, and Z. Nayyar, "Model Based Testing in Web Applications," International Journal of Scientific Engineering and Research (IJSER), pp. 56–60, 2014.

[19] S. Roohullah Jan, S. T. Shah, Z. Johar, Y. Shah, and F. Khan, " An Innovative Approach to Investigate Various Software Testing Techniques and Strategies," International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), pp. 682-689, 2016.

[20] M. Utting, P. Alexendar, and B. Legeard, "A Taxonomy of Model-based Testing Approaches," Software Testing, Verification and Reliability, pp. 297-312, 2012.