RESEARCH ARTICLE OPEN ACCESS

Comparative Analysis of Model Based Testing and Formal Based Testing - A Review

Vatsal Mishra*, Shashank Shekhar*, Shashank Shankar* Manjula R**
** (B.Tech Computer Science and Engineering, VIT University, Vellore)*
*** (Associate Professor School of Computer Science and Engineering, VIT University, Vellore)*

ABSTRACT
Software testing is one of the most important steps in the process of Software Development. Testing provides the glimpse of the proper functioning of the system under different conditions. It makes it a necessary step to choose the best testing method for the software system to be successful and accepted by a large number of people as the market is really competitive these days and only error free systems can survive for a longer period of time. This paper gives the comparative analysis of two major methods of testing : Formal Specifications Based Software Testing and Model Based Software Testing, which are used widely in the process of software development process. It brings out how these two methods of testing can provide reliability to software system including the major uses, advantages, and disadvantages of both the testing methods. It briefly gives the detailed comparative analysis of these two methods of software testing. It also brings out the situations where formal specifications based testing is more effective and efficient while model based testing being effective in others. This comparative analysis will help one in deciding on a better testing technique, depending upon the situation, and requirements of software, for the software to be successful in long run.
Keywords : Model Based Testing, Formal based Testing, Specification Based Testing, Software Testing.

I. INTRODUCTION

Software defects days are not limited to only coding error but the errors can happen in different ways [5], thus comes software testing which is an integral element of software development which should be done in a systematic approach [1].

Software testing is a component of software engineering which used for evaluating the functionalities of the system to find out whether the system is in line with functional requirements given. It can also be termed as investigation to check for different errors in the system and appraise the stakeholders with all the dimensions of the system implementation i.e. it tells the user about various utilities as well as potential risks of the system which was put to the test [3].

The history of software testing dates back to 1979 when Genford J Meyers initially introduced the separation of debugging from the process of testing[14]Further in the year 1988 Dave Gelperin and William C. Hetzel did the classification of software testing into goals and phases.[16-21]

In software testing we check the software by implementing it and then marking it on various parameters which indicates the quality of the system being tested. Testing is not a single activity but is a collection of activities and can be termed as a process .The properties to evaluate depends upon the type of system being tested, on what parameters the client's desire and the audience on target.

Testing can have static or dynamic in nature. In static testing the fault finding is done without execution of the code and in dynamic testing the fault finding is done while the code is executed. The verification process has static nature whereas validation process has dynamic nature.

As we all know testing and verification is one of the most important step in making a software, it constitutes around 30-40% of effort and time of the developer .There are many ways of testing the software like white box testing, black box testing, specification base testing ,model based testing, visual testing, grey box testing etc[2].

Both Model Based testing and formal specification based testing comes under black box testing.Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied to virtually every level of software testing: unit, integration, system and acceptance [13].

Here the paper concentrates on two major testing models which are the model based testing and the formal or specification based testing. These two models are extensively used in various commercial and educational fields these days. These models work towards single motive to evaluate the software but the methodology adapted by both of them is different. The comparative analysis shows how they differ from each other in various aspects and variations in their functionality and utility.

II. RELATED WORK

In Dalal, Siddhartha R., et al. [16] the report on various practices of development of tools and various methodologies related to model based testing. Some case studies have been shown which provides details and results of application of combination of test-generation methods on a large scale to diverse applications. Based on these, an insight has been given into what is practiced and what are the obstacles in transferring these technology to testing organizations Lyu, M.R [1] offers a view of the development, testing, and evaluation schemes for software reliability, and its integration to form a unified and consistent paradigm . Specifically, techniques and tools for the three software reliability engineering phases which are modeling and analysis, design and implementation, testing and measurement have been elucidated by the author. The book
Practical model-based testing: a tools approach [4] gives an insight to model-based testing in a practical manner, shows the way to write models for testing purposes and usage of model-based testing tools to generate test suites. It aims at testers and software developers who wish to use model-based testing, rather than at tool-developers or academics used in specification based testing. In Fujiwara, S.et al. [7] methods for the selection of appropriate test case, an important issue line with testing of protocol implementations as well as software engineering, is shown. Several issues that have an impact on the selection of a suitable test suite including the consideration of interaction measures, several test architectures for protocol testing and the fact that many specifications do not satisfy the assumptions made by most test selection method grounds have been shown. These papers are closely related to each other and give an insight of evolution of model based testing since its inception.

Hans-Martin Horcher explained a way to get more benefits from formal specifications apart from specification phase, in verifying the implementation against the specifications. He explained the use of specification in order to derive input test data and to evaluate the test results. This approach is described using the specification language Z which provides a greater degree of automation, improving the quality of testing process [26]. Charles proposed a formal semantics for the production of test cases from requirements giving a syntactical characterization of the method, which is described over the LTL formulae. He showed various examples to prove the application of the approach [29]. Mirza Mahmood Baig described about an important problem in software testing i.e. time complexity. He has decreased the time complexity related with the software testing with the help of Grover's Search Algorithm [34]. Mona Batrapresented a comprehensive analysis of formal methods including their goals, advantages, and limitations. Her research work aimed to help the software engineers in order to identify the uses of formal methods at various level of software development, and had a good reference of the requirements phase[37].

Andersson and Runeson et al. [42] presented a qualitative survey of the verification and validation processes in 11 Swedish companies by exchanging experience between the companies. They concluded that in large companies, the documented process is emphasized while in small companies, key individuals have a dominating impact on the procedures. Commercial tools are used in large companies while small companies make in-house tools or use shareware. Despite the differences in approach verification and validation is important in all industries. Gotel and Finkelstein [44] investigated and discussed the underlying nature of requirements traceability problem. They introduced the distinction between pre-requirements specification (pre-RS) traceability and post-requirements specification (post-RS) traceability and explained how the majority of the problems are due to poor requirements traceability are due to inadequate pre-RS traceability and show the fundamental need for improvements. Malaiyamodeled the relation among testing effort, coverage and reliability, and present a logarithmic model that relates testing effort to test coverage (block, branch, e-use or p-use). The results are consistent with the known inclusion relationships among block, branch and p-use coverage measures and eliminates variables like test application strategy from consideration.

III. Model Based Testing

Model based testing is a software application which is used for testing a system or a software in which test cases (called as models) are generated in whole or in part from a model that describes functional aspects of the system under test (SUT) which represents testing strategies and testing environment. [4] the models are used to generate tests which can be deployed both online as well as offline. Model based testing can also be called as Model Driven Testing.

The basic idea in model based testing is generation of models which can be transition system, UML State Machines, finite state machines, class diagrams along with constraints, etc.[7-12]from which complete test cases which is input and

believed output pairs that can be generated. It supports *investigation*, *construction*, and *prediction* of the modeled system.
In online testing the model and the considered system for test are connected and the tests happen dynamically whereas in offline testing the test cases/suites are generated which can be later be tested with the system using a tool.

A model which describes the system or software under the testing process can be taken as its partial or abstract behavior towards the system. Hence the test cases which are taken out from a model can be said as functional tests on the similar platform of abstraction as the model. These test cases are collectively known as an abstract test suite.

The following diagram shows a simplified workflow for MBT. [6]

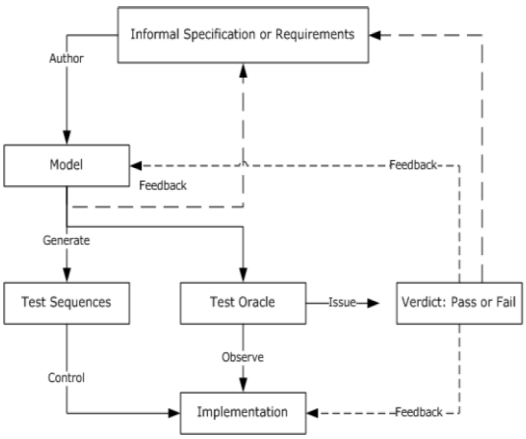

Figure 1. ***Workflow of Model Based Testing***

The process followed in model based testing:

1) The creation of model can be done in several ways. [7-12] The model is created based on the requirements, specifications or use cases which are provided. The model goes through various feedbacks from the user before being developed as a formal final model. The implementation is done based on the model.

2) After the model is created the test suites are generated. These test suites can be derived in various manners because the testing is experimental and it is based on heuristics these test suites contain test sequences and test oracle.

3) The role of test sequences is to control the system under test, which makes it go into the different conditions under which it can be tested whether the system has followed the model on which it was developed or not . The test oracle checks the growth of the system in terms of its implementation and delivers a pass or fail verdict based on its conformation with the model.

4) The verdict is the final conclusion of the testing process and it provides details about all the artifacts. A failure indicates that the behavior of the system under test does not go in line with the model predictions. This generally means errors or faults in the implementation process, but sometimes it can also mean that a there is a flaw in model creation or design or that the informal requirements from which it was created were incorrectly taken.

There are various tools which can be used for model based testing. Some of them are:

Conformiq designer [22]commercial tool in which models can be generated as UML State Machines and in Qtronic Modeling Language (QML). Tests can be exported to test management tools or TTCN-3.

Graphwalker which is an open source tool in which test cases are made from finite state machines and uses search algorithms to A* or random search algorithm to cover various states,edges, requirements.
JSXM[23] which is an academic type software, it is a model animation and test generator which uses different types of EFSMs as its input.The generated tests are then converted into JUnit test cases.

PyModel[24] which uses python as the coding language which is an open source tool which supports online as well as on-the-flying testing methods.It uses compositions for controlling the scenario. The guidance of the coverage can be done as programmable strategy.

Spec Explorer [25] which develops the programs in C# is a commercial tool is a successor of AsmL test tool which is now integrated with Visual Studio.

Some applications of model based testing are arithmetic and table operators, message of parsing and building, rules based system and user interfaces .

Although this type of testing requires significantly more up-front effort in building themodel, it offers an upper hand over traditional software testing methods.

IV. FORMAL SPECIFICATION BASED TESTING

Formal Specifications based Testing has now gained much importance in the field of software development.The objective of Formal Specifications based Testing is to test the functionality of software according to the required specifications. Formal methods of testing usually include mathematical notations, formal logic and proofs in some cases used in verifying the functionalities of software. Formal Specifications helps greatly in simplifying the process of testing in software development process.It is one of the most convenient testing method which helps in detecting errors and bugs in software functioning. It is capable of producing not only the test data sets for testing purpose but also helps in analyzing test results effectively and efficiently. In fact, this is one of the major uses of Formal Specifications. These tests might be functional or non-functional in nature.

Most of the existing tools and techniques used for software testing requires testing of a written set of programs. But the evolution of formal methods of software testing has given rise to the possibility of starting from the specifications for introducing some testing methodsin formal framework. Prolog is one of the tool used in formal based testing for test set production.

Formal Specifications based testing can be easily fit in the software development processes without having the need to replace traditional methods of software development as cited in [26]. It is really effective in case of unit testing in which individual components are tested independently to ensure that they operate correctly [32]. The mathematical notations used in formal methods helps in defining the required functionalities of the system.

There are many methods of testing based on the formal specifications of software such as Algebraic Specifications, Finite State Machines, transition Systems etc. Algebraic specification is basically used to specify the software behavior with the help of methods rose from abstract algebra. Some tools for developing algebraic specification languages are LARCH, ASL etc.Some of other formal specifications languages includeOBJ, LOTOS, ASM, LARCH, Communicating Sequential Processes (CSP) etc [35].

ASM (Abstract state machine) is a state machine which operates on arbitrary data structures known as states. The ASM Method is a scientific and practical systems engineering method which bridges the gap between the two ends of system development. It uses 3 concepts:

a) *ASM*: a pseudo-code, which generalize Finite State Machines
b) *Ground model*: a form of rigorous blueprints
c) *Refinement*: a general scheme for instantiations of model abstractions to elements of the concrete system.

OBJ is basically a family of declarative languages and was created by Joseph Goguen in the year of 1976. It consists of generic modules, abstract data types, expressions to combine modules etc.

The family of Larch formal specification languagesaremeant to be used for the accurate specification of computer systems. They provide the capability forbetter specification of computer programs and the derivation of proofs regarding the behavior of the program. Larch family has a language called LSL (Larch Shared Language) for algebraic specification of abstract data types.

Communicating Sequential Processes (CSP) is a type of formal specification language and helps in description of various patterns of interaction in concurrent system. It belongs to the family of mathematics known as process algebras, and is based on message passing through channels.

LOTOS (Language of Temporal Ordering Specification) is a formal specification language which is basically based on temporal ordering of events as the name suggests. LOTOS is mostly used for specifying protocols in OSI Standard. It is an algebraic specification language that has two parts: a part for the data description and operations, and a part for concurrent processes description, which is based on process calculi.

The Z notation is one of the most popular formal specification languagethat is used for description and modelling of the computer systems. It mostly targets the specification of computer programs and computer-based systems in general. Z consists of a standard catalogue which is atoolkit of commonly used mathematical predicates and mathematical functions.

Apart from this formal based methods are useful in reduction of time complexity which is one of the major issue in software development. It can be done using Grover's Algorithm as cited in [34]. These methods of testing is are also capable of determining the causes of software failure during testing processes. Test sets and the related hypotheses can be generated easily using Horn Clauses [31].

V. COMPARATIVE ANALYSIS

5.1 Definitions used

5.1.1 Statement coverage (Block coverage)

In software testing, testers are required to generate test cases to execute every statement at least once [38]. A test case serves as an input to the program under test, and is executed during testing. Statement coverage is defined as the fraction of total number of blocks or statements that are executed by test data [38, 39].

5.1.2 Branch coverage (Decision coverage)

Branch coverage is defined as the fraction of total number of decisions or branches that are executed by test data [25, 39]. It also helps to ensure that no branch results to abnormal behaviour of application and validating all possible branches in the program [45].

5.1.3 Path coverage

In path coverage, test case is executed at least once i.e. all the execution paths of theprogram from entry to exit are executed during testing [40, 41].

5.1.4 Requirement Traceability

Requirement traceability is the ability to describe and follow a requirementin both forward and backward direction [42], by defining and maintainingrelationships to related development artefacts [43] such as code, configuration files andtest cases. Testing is a significant component in the software development lifecycle.Having many test cases leads to increase in effort and cost spent on testing, thus many industrial developers, testers and managers give a lot of importance to traceability [44, 41]. There are some tools support to maintain, retrieve and record trace information manually [46]. However, this is time consuming, labour- intensive and error-prone [47, 44]. It is more convenient and important to create, maintain and find the links of traceability in testing through an automated process as requirement traceability links are outdated when a software evolves.

e.g.: Calculation of statement coverage, branch coverage and path coverage for the following code snippet.

```
Read P
Read Q
IF P + Q > 100 THEN
Print "Large"
ENDIF
IF P > 50 THEN
Print " P Large"
ENDIF
```

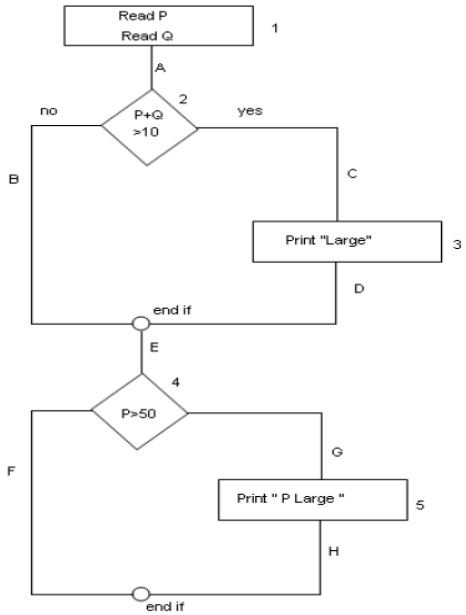

*Figure 2.***Flowchart for code snippet**

- **Statement Coverage**

To calculate statement coverage, find out the shortest number of paths following which all the nodes will be covered. Here by traversing through path 1A-2C-3D-E-4G-5H all the nodes are covered. So by travelling through only one path all the nodes 12345 are covered, so statement coverage in this case is 1.

- **Branch Coverage**

To calculate Branch Coverage, find out the minimum number of paths which will ensure covering of all the edges. In this case there is no single path which will ensure coverage of all the edges at one go. By following paths 1A-2C-3D-E-4G-5H, maximum numbers of edges (A, C, D, E, G and H) are covered but edges B and F are left. To covers these edges we can follow 1A-2B-E-4F. By combining the above two paths we can ensure of travelling through all the paths. The aim is to cover all possible true/false decisions.Hence, branch coverage is 2.

- **Path Coverage**

Path Coverage ensures covering of all the paths from start to end. All possible paths are
1A-2B-E-4F
1A-2B-E-4G-5H
1A-2C-3D-E-4G-5H
1A-2C-3D-E-4F
So, path coverage is 4.

5.2 Comparison of Model Based Testing with Formal Based Testing

We have used Lickert Scale (1-5) to show the average rating of Formal based testing (FBT) and Model based testing (MBT) approach while considering some aspects of software testing [48].

5.2.1 Test Coverage

MBT Score: 5
FBT Score: 4

Test coverage is a strength of both MBT and FBT. The coverage usually depends on the experience of the tester who is writing the test. In companies, generally the test cases are written by experienced testers, if not, the test cases are approved by experienced testers. Also, in MBT, the test coverage is higher than FBT because the test cases are generated by considering the test coverage.

FBT has fairly high test coverage (i.e. branch, path and statement coverage) but MBT has more coverage because of its zero - tolerance towards the test coverage.

5.2.2 Requirement Traceability

MBT Score: 2
FBT Score: 4

There are several ways to make the requirements traceable through the test cases using formal specifications. Different applications are used in order to link the test cases to the software requirements. There can be a separate column in the test case which indicates that a test case belongs to a specific requirement.

In MBT, the traceability is done in a different way. Requirement traceability is a challenge in MBT and industries usually find it difficult to track the results back to the system requirements in the MBT approach. Recently, some major studies [1, 2 ,10] have been done in order to find out a better way to make requirements more traceable in MBT process.

5.2.3 Understandability of Test cases

MBT Score: 4
FBT Score: 3
The understandability of the test cases depends on the experience of the tester who is writing the test cases. It is a major challenge of FBT because every tester writes the test cases according to his own knowledge of the system and business.

In MBT, the automated test cases are not fully understandable by humans. It depends on which tool you are using. For example, ConformiqQtronic and Microsoft's Spec Explorer adds reasonable details on the test cases, so thathumans can understand what are the details and what is to be tested.

5.2.4 Cost and Time

MBT Score: 5
FBT Score: 3

Cost and time are one the most important attributes of any approach. It cannot be analyzed which is more costly FBT or MBT. It depends on different scenarios, the application to be tested, method of testing and conditions of testing. In the total cost of a project, there is around 50-70% cost for quality assurance and if any defect arises in the release, the cost is increased even more. MBT is known for decreasing the overall testing cost and it takes less time as compared to traditional software testing approaches.

5.2.5 Test Design and Planning

MBT Score: 4
FBT Score: 2

Test design and planning depends on the system requirements. Understanding the requirements is one of the most important tasks before test design and planning. If the test plans and designs are made without fully understanding the system requirements, there is a high chance that re-work will be required.

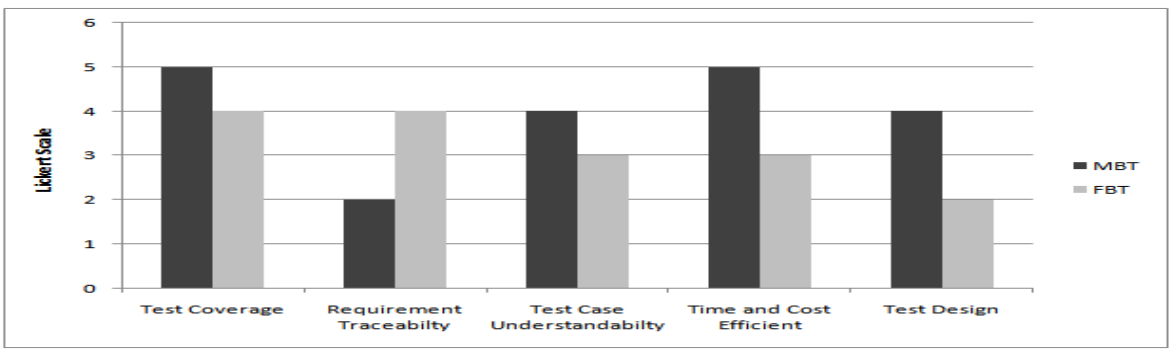

Figure 3.Comparative Analysis Scores

5.3 Discussion and Inference

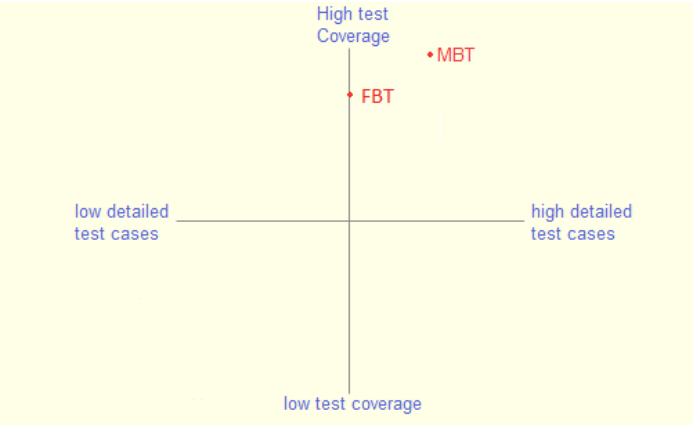

*Figure 4.***Cross plot graph for Test Coverage and Test Case detail**

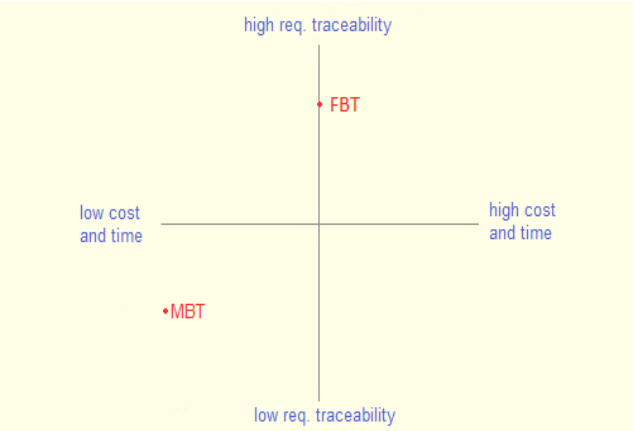

*Figure 5.***Cross plot graph for Requirement Traceability, Time and Cost**

The rating of the two approaches on constructs helped in generating the results of our findings.

FBT has high coverage but the test cases are not much detailed. MBT has higher test coverage and more detailed test cases if compared to the FBT approach. The coverage depends on the extent of testing and understanding of requirements. Thus, the coverage depends on the quality of the test cases, which depends on the quality of the system requirements.MBT has comparatively low requirement traceability as compared to FBT. But at the same time, MBT is more cost and time efficient.
Advantages of model based testing includes less cost in project maintenance and requirement specification frequency. Further there is fluidity in the designing aspect i.e. in case of addition of a feature, a new finite state machine can be added without disturbing the existing machine model. Which means simple change can automatically ripple through the entire suite of test cases and through this higher level of automation is achieved. Hence more importance is given to the design rather than coding. This testing model has vast coverage which means exhaustive testing is possible.

However there are some difficulties which are faced in model based testing. First, it requires extensive formal specification to build the model and test it. Further the test cases are highly dependent on the model on which is structured upon.
In formal specification based testing has many advantages which makes it really powerful testing technique. The main advantage of formal testing methods is that it greatly reduces the amount of time and effort used during later stages of testing by efforts used in earlier stages of system construction. It helps in removal of inconsistencies in the process of software development and also provides validation to every step in software development [26]. It also provides support in Model Based Testing.
Another major advantage as mentioned above include the production of best test cases or test data sets and test result analysis which helps in detecting very minute errors during testing process. It is based on mathematical notation and proofs which makes it a really reliable technique for software testing in comparison to other techniques of software testing. It provides the accuracy measure of the functions of software and at the same time is cheaper method to implement. Abstraction is another advantage in formal specifications method.
Apart from these, formal based methods of software testing also has some limitations. Formal Specifications Based Testing techniques are quite complex for integration testing which include system testing and sub-system testing but can be applied. This is because these techniques do not describe the architecture and interrelationships between the operations of the system. Apart from this it does not take into account the informal measures and hence may not produce completely correct results in complex situations [35]. There are some advancements need to be done for making formal specification based testing a more effective tool. These can include ambiguity resolution technique, combination of two or more mathematical models etc. in order to get even better results.

Attribute \ Test	MBT	FBT
Test Coverage	High	High
Requirement Traceability	Low	High
Test Case Understandabilty	High	Medium
Time and Cost	Low	High
Test Design	High	Low
Reusability	Medium	Low
Accuracy	Medium	High

Comparison table between MBT and FBT

VI. CONCLUSION

In this research paper we have compared two black testing techniques: formal based testing and model based testing. Model based testing generates test cases in the form of models and tests are done in conformance with it while in formal based testing, specific methods are applied to test the software. On the basis of our findings we conclude that formal based testing is better in terms of requirement traceability but model based testing is more feasible in terms of cost and time of project maintenance. Formal based testing supports and enhances the features model based testing to a great extent .The combined model is termed as Formal Model Specification based Testing which combines the characteristics of both the models and works more efficiently.

VII. ACKNOWLEDGEMENT

We would like to acknowledge the immense support of our Professor Dr Manjula R without whom it was not possible to write the paper. Her guidance formed the base of the paper and her expertise comments greatly enhanced the manuscript. We would also like thank our university (VIT University, Vellore) which has always encouraged the students to go in the field of research and develop an interest towards exploration and innovation.

REFERENCES

[1] Lyu, M.R. 1998. An integrated approach to achieving high software reliability. Aerospace Conference, 1998. Proceedings. , IEEE (1998), 123 – 136.

[2] Patton, Ron. *Software Testing*. , International Standard Book Number: 0-672-31983-7, 2001

[3] Kaner, Cem (November 17, 2006). "Exploratory Testing" (PDF). Florida Institute of Technology, QualityAssurance Institute Worldwide Annual Software Testing Conference, Orlando, FL.

[4] Utting, Mark, and Bruno Legeard. Practical model-based testing: a tools approach. Morgan Kaufmann, 2010.

[5] Infamous Error Case Studies,Patton, Ron. *Software Testing*. , International Standard Book Number: 0-672-31983-7, 2001,pg10-13

[6] https://msdn.microsoft.com/en-us/library/ee620399.aspx

[7] Fujiwara, S., Bochmann, G., Khendek, F., Amalou, M., and Ghedamsi, A., "Test Selection Basedon Finite State Models", IEEE Transactions on Software Engineering, Vol. 17, No. 6., June 1991.

[8] Chow, T. S., "Testing Software Design Modeled by Finite-State Machines," IEEE Transactions on Software Engineering, Vol. SE-4, No. 3, May 1978.

[9] Holzmann, G., Design and Validation of Computer Protocols, AT&T Bell Laboratories,Prentice Hall, 1991.

[10] Musa, J.D., "Operational profiles in software reliability engineering," IEEE Software, 10(2), pp14-32.

[11] Apfelbaum, L., "Automated Functional Test Generation", Proceedings of the Autotestcon'95 Conference, IEEE, 1995.

[12] Savage, P., Walters, S, and Stephenson, M., "Automated Test Methodology for Operational Flight Programs", Proceedings of the 1997 IEEE Aerospace Conference, 1997.

[13] Black Box Testing,Patton, Ron. *Software Testing*. , International Standard Book Number: 0-672-31983-7, 2001,pg55

[14] Myers, Glenford J. (1979). The Art of Software Testing. John Wiley and Sons. ISBN 0411043281.

[15] Gelperin, D.; B. Hetzel (1988). "The Growth of Software Testing". CACM 31 (6): 687–695.doi:10.1140/62959.62965. ISSN 00010782.

[16] Dalal, Siddhartha R., et al. "Model-based testing in practice." *Proceedings of the 21st international conference on Software engineering*. ACM, 1999.

[17] Until 1956 it was the debugging oriented period, when testing was often associated to

debugging: there was no clear difference between testing and debugging. Gelperin, D.;B. Hetzel(1988). "The Growth of Software Testing". CACM 31 (6). ISSN 00010782.

[18] From 1957–1978 there was the demonstration oriented period where debugging and testing was distinguished now – in this period it was shown, that software satisfies the requirements. Gelperin, D.; B.Hetzel (1988). "The Growth of Software Testing". CACM 31 (6). ISSN 00010782.

[19] The time between 1979–1982 is announced as the destruction oriented period, where the goal was to find errors. Gelperin, D.; B. Hetzel (1988). "The Growth of Software Testing". CACM 31 (6). ISSN 00010782.

[20] 1983–1987 is classified as the evaluation oriented period: intention here is that during the software lifecycle a product evaluation is provided and measuring quality. Gelperin, D.; B. Hetzel(1988). "The Growth of Software Testing". CACM 31 (6). ISSN 00010782.

[21] From 1988 on it was seen as prevention oriented period where tests were to demonstrate that software satisfies its specification, to detect faults and to prevent faults. Gelperin, D.; B. Hetzel (1988). "The Growth of Software Testing". CACM 31 (6).ISSN 00010782.

[22] A. Huima (2007) "Implementing Conformiq Qtronic," Testing of Software and Communicating Systems, Springer, pp. 1-12. DOI: 10.1007/978-3-540-73066-8_1

[23] D. Dranidis, K. Bratanis and F. Ipate (2012) "JSXM: a tool for automated test generation". In Proc. of SEFM'12. DOI: 10.1007/978-3-642-33826-7_25

[24] J. Jacky (2011), "PyModel: Model-based testing in Python", Proc. of the 9th Python in Science Conf. (SCIPY 2011) link.

[25] M. Veanes et al. (2008) "Model-Based Testing of Object-Oriented Reactive Systems with Spec Explorer," Formal Methods and Testing,Springer, pp. 39-76. DOI: 10.1007/978-3-540- 789178_2

[26] Hans-Martin Horcher and Jan Peleska, "Using formal specifications to support software testing", Springer, 1995

[27] YoonsikCheon and Myoung Kim, "A Specification-Based Fitness Function for EvolutionaryTesting of Object-Oriented Programs", ACM, 2006

[28] D. RICHARD KUHN, "Fault Classes and Error Detection Capability of Specification-Based Testing", ACM, 1999

[29] Charles Pecheur, Franco Raimondi and Guillaume Brat, "A Formal Analysis of Requirements-Based Testing", ACM, 2009

[30] James A. Whittaker and J. H. Poore, "Statistical Testing for Clean room Software Engineering", IEEE, 1992

[31] Gilles Bernot, Marie Claude Gaudel and Bruno Marre, "Software testing based on formal specifications :a theory and a tool", Software Engineering Journal, 1991

[32] Weikai Miao and Shaoying Lin, "A Formal Specification – Based Integration Testing Approach", Springer, 2013

[33] Marie-Claude Gaudel, "Software Testing Based on Formal Specification", Springer,2010

[34] Mirza Mahmood Baig and Dr.Ansar Ahmad Khan, "A Formal Technique for Reducing Software Testing Time Complexity", Springer, 2010

[35] Mona Batra, Amit Malik, and Dr.Meenu Dave, "Formal Methods : Benefits, Challenges and Future Direction", Journal of Global Research in Computer Science, 2013, Volume 4, No.5

[36] Pressman Roger S: "Software Engineering"-A Practitioner's Approach", McGraw Hill, 5th edition. 2000.

[37] Mona Batra, S.K Pandey: Formal methods in requirement engineering. International Journal of Computer Applications , pp- 7-14, Volume 70–No.13

[38] Malaiya, Y.K.; Naixin Li; Bieman, J.; Karcich, R.; Skibbe, B.; , "The relationship betweentest coverage and reliability," *Software Reliability Engineering, 1994. Proceedings., 5th International Symposium on* , vol., no., pp.186-195, 6-9 Nov 1994

[39] Hong Zhu, Patrick A. V. Hall, and John H. R. May. 1997. Software unit test coverage and adequacy. *ACM Comput. Surv.* 29, 4 (December 1997), 366-427

[40] Wood, M., Roper, M., Brooks, A and Miller, J. 1997. Comparing and combining software defect detection techniques : a replicated empirical study. *Software Engineering— ESEC/FSE'97*. (1997), 262–277

[41] http://ajoysingha.info/Documents/Branch%20Statement%20Path%20Coverage.pdf

[42] Andersson, C. and Runeson, P. 2002.Verification and validation in industry-a q ualitativesurvey on the state of practice. *Empirical Software Engineering,*

2002. Proceedings. 2002International Symposium n (2002), 37–41.

[43] Andrea De Lucia, FaustoFasano, Rocco Oliveto, and GenoveffaTortora. 2007. Recovering traceability links in software artifact management systems using information retrievalmethods. *ACM Trans. Softw. Eng. Methodol.* 16, 4, Article 13 (September 2007). DOI=10.1140/1276933.1276934 http://doi.acm.org/10.1140/1276933.1276934

[44] Gotel, O.C.Z. and Finkelstein, C. 1994. An analysis of the requirements traceability problem. *Requirements Engineering, 1994., Proceedings of the First International Conference on* (1994), 94–101.

[45] Tamai, T. and Kamata, M.I. 2009. Impact of Requirements Quality on Project Success or Failure. *Design Requirements Engineering: A Ten-Year Perspective*. (2009), 258–275.

[46] "Doors," http://www-01.ibm.com/software/awdtools/doors/

[47] Brinkkemper, S. 2004. Requirements engineering research the industry is and is not waitingfor. *10th Anniversary Int. Workshop on Requirements Engineering: Foundation forSoftware Quality. Riga Latvia* (2004), 41–54.

[48] Agruss, C. and Johnson, B. 2000. Ad Hoc Software Testing: A perspective on exploration and improvisation. *Florida Institute of Technology* (2000).