International Journal For Technological Research In Engineering
Volume 4, Issue 12, August-2017 *ISSN (Online): 2347 - 4718*

REVIEW OF LITERATURE FOR MODEL BASED TESTING

Arpita Mathur[1], Ajay Mathur[2]
[1]Department of Computer Science, Lachoo Memorial College of Sc. & Tech., A-Sector, Shastri Nagar, Jodhpur (India)
[2]Department of Computer Science, Govt. Polytechnic College, Jodhpur (India)

Abstract: Due to the huge complexity of modern software systems, it is required to specify precisely what a software component should do and how it should behave. If the final implementation deviates from the expected behavior, then the use of the developed component may fail. This also applies for the development of communicating protocols as they are merely implemented in the software. A protocol definition defines the syntax, semantics, and synchronization of communication.

I. INTRODUCTION

A literature survey was done on how model based testing can be used to test the protocols that shows sequential (time based) temporal relationship between entities. Unified Modeling Language (UML) sequence diagram was also studied for its use as a base for designing the dynamic model for protocol to shows how the entities interact/communicate. The protocol specification mechanism may be developed for the temporal protocols using UML.

II. LITERATURE SURVEY

Glenford J. Myers has given definition of testing as "Testing is the process of executing a program with the intent of finding errors" in his book titled "The Art of Software Testing". According to him understanding the true definition of software testing can make a profound difference in the success of your efforts. If our goal is to demonstrate that a program has no errors, then we tend to select test data that have a low probability of causing the program to fail. On the other hand, if our goal is to demonstrate that a program has errors; our test data will have a higher probability of finding errors. The latter approach will add more value to the program than the former.

Antonia Bertolino and Eda Marchetti provide a comprehensive overview of software testing, from its definition to its organization, from test levels to test techniques, from test execution to the analysis of test cases effectiveness in their paper titled "A Brief Essay on Software Testing". According to them testing is an important and critical part of the software development process, on which the quality and reliability of the delivered product strictly depend.

As explained in "Testing and Quality Assurance for Component-Based Software" random testing is a strategy that requires the "random" selection of test cases from the entire input domain. The authors Jerry Zeyu Gao, H. S. Jacob Tsao and Ye Wu the advantage of random testing is its efficiency. Test cases can be generated automatically and require no further efforts such as sub domain classification in partition testing.

According to paper titled "Model-based Software Testing" by Ibrahim K. El-Far and James A. Whittaker software testing requires the use of a model to guide such efforts as test selection and test verification. When the mental models are written down, they become sharable, reusable testing artifacts. In this case, testers are performing what has become to be known as model-based testing." This paper introduces model-based testing and discusses its tasks in general terms with finite state models as examples.

In the paper titled "Model-Based Testing" author Mark Utting explains that model-based testing is a break-through innovation in software testing because it completely automates the validation testing process. Model-based testing tools automatically generate test cases from a model of the software product. The generated tests are executable and include an oracle component which assigns a pass/fail verdict to each test. The paper gives an overview of the variety of methods and practices of model based testing, and then speculate on how model-based testing might promote or complement the program verifier grand challenge.

The author Bernhard Rumpe suggests that an approach using models as central development artifact needs to be added to the portfolio of software engineering techniques, to further increase efficiency and flexibility of the development as well as quality and reusability of results in his paper "Model-based Testing of Object-Oriented Systems". A number of test patterns are proposed that have proven helpful to the design of testable object-oriented systems. In contrast to other approaches, this approach uses explicit models for test cases instead of trying to derive (many) test cases from a single model.

In the paper titled " UML Profiles for Modeling Real-Time Communication Protocols" the authors Barath Kumar and Juergen Jasperneite explains that validation of non-functional and functional properties of the protocols during the early stages of design and development is important to reduce cost resulting from protocol anomalies, design errors like deadlock or livelock situations and/or violations of time constraints. In this paper they review the most important real-time UML profiles which can be used for designing time-critical Industrial Communication Protocols (ICPs). In the paper "The Use of UML Sequence Diagram for System-on-Chip System Level Transaction-based Functional Verification" the authors Yu JinShan, Li Tun, Tan QingPing propose a method to support transaction level verification of system-on-chip based on UML sequence diagram. They use UML sequence diagram to capture the communication and collaboration behavior among IP cores in system-on-chip and build high level specification for transaction level verification. Then these sequence diagrams were used to guide the generation of transaction test sequence. They verified a typical SoC design. UML sequence diagram can capture the complex communication behavior among IP cores in SoC design and it efficiently supports SoC system level functional verification was shown by an experiment.

In the paper titled "An Evaluation of Random Testing" authors Duran, W. Joe, Ntafos, C. Simeon presents simulation results which suggest that random testing may often be more cost effective than partition testing schemes. Results of actual random testing experiments are also presented which confirm the viability of random testing as a useful validation tool.

The authors I. Ciupa, Pretschner, A. Leitner, M. Oriol, B. Meyer, of the paper titled "On the Predictability of Random Tests for Object-Oriented Software" have evaluated the variance of the number of faults detected by random testing over time through an experiment to show the predictability of random testing on object-oriented programs. Their study provides evidence that the relative number of faults detected by random testing over time is predictable but that different runs of the random test case generator detect different faults. The study also shows that random testing finds faults quickly.
Dick Hamlet explains in his paper titled "When only random testing will do" that in some circumstances, random testing methods are more practical than any alternative, because information is lacking to make reasonable systematic test-point choices. In this work he examined some situations in which random testing is indicated. He discusses issues and difficulties with conducting the random tests.

The paper "Weak Mutation Testing and Completeness of Test Sets" by author William E. Howden describes an error-based method called weak mutation testing. In this method, tests are constructed which are guaranteed to force program statements which contain certain classes of errors to act incorrectly during the execution of the program over those tests. Weak mutation testing requires the identification of classes of elementary program components and of simple errors that can occur in the components.

M. R. Woodward gives a brief review of the program testing technique known as mutation testing, an error-based testing technique in his paper titled "Mutation testing-an evolving technique". In this technique a large number of simple changes (mutations) are made to program one at a time. Test data then has to be found which distinguishes the mutated versions from the original version. Mutation testing as originally developed was considered costly. The author discusses strong and weak mutation testing and considers its use in testing algebraic specifications.

The paper "A practical system for mutation testing: help for the common programmer" outlines a design for a system that will approximate mutation, but in a way that will be accessible to everyday programmers. The author A. J. Offutt visualizes a system to which a programmer can submit a program unit, and get back a set of input/output pairs that are guaranteed to form an effective test of the unit by being close to mutation adequate.

III. CONCLUSION

An overview of the literary survey is done on software testing, model based testing, random testing, mutation testing and Unified Modeling language. Some of the popular testing techniques were also reviewed. Study shows the need of modeling software and that the process of model based testing of dynamic model and temporal relationship may be validated through model based testing. After validation this dynamic model may become the oracle for validating the proposed protocol. An experiment may be conducted on this by giving random inputs to the temporal relationship and dynamic model.

REFERENCES

[1] Glenford J. Myers, The Art of Software Testing, second edition, John Wiley & Sons, Inc., 2004, ISBN 0-471-46912-2.
[2] Practical Software Testing, A Process-Oriented Approach, Ilene Burnstein, Springer-Verlag New York, Inc.,2003, ISBN 0-387-95131-8.
[3] Software Testing and Continuous Quality Improvement, Second Edition, William E. Lewis, Auerbach publications, ISBN 0-8493-2524-2.
[4] IEEE Standard Glossary of Software Engineering Terminology (Std610.12-1990), Copyright 1990 by IEEE.
[5] UML based Test Specification for Communication Systems, A Methodology for the use of MSC and IDL in Testing, Dissertation, 2004
[6] H. Balzert, Software Management, volume 2, first edition, 1998, ISBN 3827400651.
[7] Kaner, C., J. Falk, & H. Nguyen, 1999, Testing Computer Software, Wiley Computer Publishing, second edition, ISBN 0471358460.
[8] E.J. Weyuker, "On Testing Non-testable Programs", The Computer Journal, vol. 25, no.4, pp. 465—470, 1982.
[9] Antonia Bertolino, Eda Marchetti, "A Brief Essay on Software Testing"
[10] Ibrahim K. El-Far and James A. Whittaker, "Model-Based Software Testing", Florida Institute of Technology.
[11] J.J. Marciniak , "Encyclopedia on Software Engineering", Wiley, 2001
[12] Harry Robinson, "Intelligent Test Automation ", 2000

[13] Dehla Sokenou, "Generating Test Sequences from UML Sequence Diagrams and State Diagrams", GEBIT Solutions.
[14] 'Unified Modeling Language Specification", Version 2.0, OMG, 2004.
[15] Lu Luo, "A UML Documentation for an Elevator System Distributed Embedded Systems", PhD Project Report, December 2000.