ISSN No. 0976-5697

Volume 4, No. 11, Nov-Dec 2013

International Journal of Advanced Research in Computer Science

REVIEW ARTICLE

Available Online at www.ijarcs.info

A Novel Model Based Testing (MBT) approach for Automatic Test Case Generation

Syed Usman Ahmed
Department of Information Technology
Jodhpur Institute of Engineering and Technology
Jodhpur, India
syedusman.ahmed@jietjodhpur.com

Mohammed Asim Azmi
Fachhochshule Frankfurt am Main
University of Applied Science
Frankfurt, Germany
azmi@stud.fh-frankfurt.de

Abstract: Software testing is a domain of software engineering that deals with the validation and verification of software work products. This domain has gained a lot of importance after the advancement and realization of the concept of globalization. Software testing was done manually till last decade, however this paradigm has seen a shift from manual techniques to automatic process because of the increasing concern about software reliability and usability, cost incurred on software testing, scaling, performance and compatibility issues. We present an approach for automatic test case generation based on Model Based Testing (MBT) methodology. In this approach a UML model is created using a standard tool from the set of user scenarios and then based on this model automatic test case are generated by applying certain algorithms.

Keywords: software testing, test cases, automated, Model Based Testing (MBT), UML diagrams.

I. INTRODUCTION

Software testing is a very important and expensive process, thorough software testing is necessary to produce highly reliable systems [1]**.** This reliability is achieved by exercising a given program with well designed input data with the intend of observing failures [2, 3, 4]. Testing a software work product is done to find out the difference between expected behavior as specified by system models and the observed behavior of the implemented system [5].

Unified Modeling Language (UML) is today's most widely used technique to describe the used specification and design specification which is developed at the time of software work product development. These UML models can be used as an important source of information for designing test cases. We in this part of the introductory chapter present various UML models that can be used for the same.

II. LITERATURE SURVEY

Use case diagrams are generally used to gather requirements from the customers at the time of requirement elicitation process. The basic components of use case diagram are: ***Set of use cases***, ***use cases***, ***lines***, and ***actors***. The actors are related to the use cases with the help of lines [10]. The test case generation based on use-case diagram is being the forte of many of the researches, however these scholars differ in the approaches they follow for generating test cases using use case diagram. Some of these scholars used approach of coverage of all scenarios based on each used case whereas other worked on the dependencies between use cases and created graphical notation for generating automatic test cases.

Collaboration diagram is also called as communication diagram when used in context of Unified Modeling Language (UML) version 2.0. In a collaboration diagram various messages are passed among entities or objects. These messages are numbered in a sequence. Various researchers have proposed many ways to generating test cases based on communication diagrams.

A Sequence diagram represents interaction between different process and the order in which they interact. These interactions are arranged in time sequence, and represent the objects and classes that are involved in the scenario. The messages that are required to carry out the functionality between object are also depicted with the help of sequence diagram.

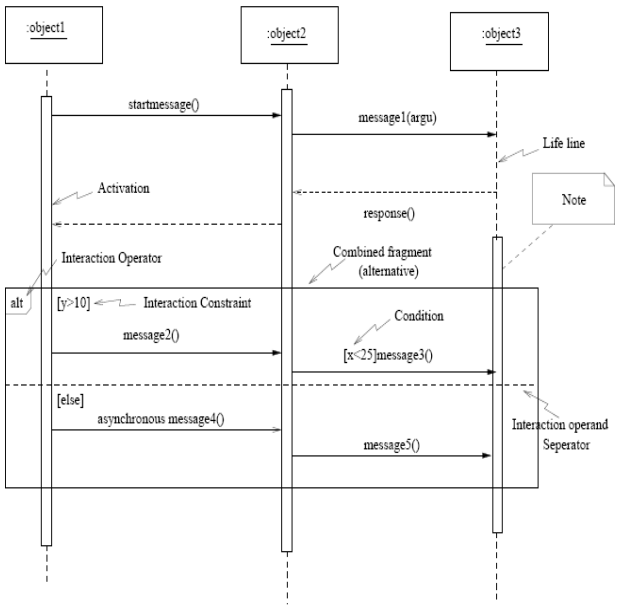

Figure 1- A Sequence Diagram Showing Various Notations

Martin Glinz et al. [6] have mentioned that scenarios (Use cases) are used to describe the functionality and behavior of a (software) system in a user-centered perspective. As scenarios form a kind of abstract level test cases for the system under development, the idea to use them to derive test cases for the system test is quite intriguing. Yet in practice scenarios from the analysis phase are seldom used to create concrete system test cases.

As per Sandeep et al. [7] manual testing is a very costly and time consuming process. In order to reduce this cost and increase reliability, model based testing approach is used. Model Based Testing (MBT) is a process of generating test cases and evaluating test results based on the design and analysis models. MBT lies in between specification and code based testing [7]. Hence it is also known as Gray-Box testing approach. A wide range of models like UML, State charts, Data flow diagrams, Control flow diagrams etc. have been used in MBT [7].

Nirpal et al have mentioned that Genetic Algorithm starts with a set of first generation individuals, which are then grouped at random from the problem area. The algorithms are generated to execute a series of executions that transforms the present generation into a new and filter generation [8]. Each individual in each generation is evaluated with a fitness function. Based on this evaluation the individual may approach the optimal solution.

III. RELATED WORK

As software testing is gaining momentum many researchers are getting associated with this domain to explore the real potential of testing. Forhlich et al [11] have used the Use Case diagram and transformed the same into state machine representation with the help of pre and post conditions. In a state machine representation states represents intervals between two successive messages. For validating their approach Forhlich have used transition coverage as testing criteria and validated the test sequence generated from the state machine based model. Philippow and Gotze in their research work have generated test cases from use case by converting the same to state diagrams [12]. They have created a usage model [28] and the same will be traversed to generate test cases.

Samuel and Mall [13] have used a cluster level approach to generate test cases based on UML communication diagrams. They transformed the communication diagram into tree representation and did a post order transversal to create conditional predicate. Test cases are then generated from these conditional predicate by applying the function minimizing technique. The test cases so generated by Samuel et al [13] performed the path coverage as well as boundary coverage. Path coverage tests all the valid paths in the tree representation and boundary coverage test all the boundary values for a valid test case.

Samuel and Mall [14] have proposed a novel methodology for the generation of test cases based on sequence diagram. They created a Method Dependency Graph (MDG) from sequence diagram. MDG shows the dependency of a given node with others. A dynamic slice of the sequence diagram is created from its respective MDG, for dynamic slicing they have used the edge marking method. Edge marking algorithm is based on marking and unmarking the unstable edges as and when the dependency arises and cease at run time.

IV. PROPOSED WORK

To fully understand and generate test cases we have to follow the below mentioned steps as per the proposed method.

a. Create a UML model using any requirement gathering tool like Rational Rose or Rational Software Architect (RSA).
b. Convert this UML model into XML or XMI format.
c. Generate a parser in any object oriented language and using the same to read the XML or XMI format.
d. Based on the parsing of the XML or XMI file generate a graph, G (V, E) where V is set of nodes and E is set of edges.
e. Traverse the graph G (V, E) with the help of Prism algorithm or Dijkstra's algorithm [15].
f. Find the basis path and record the test cases.

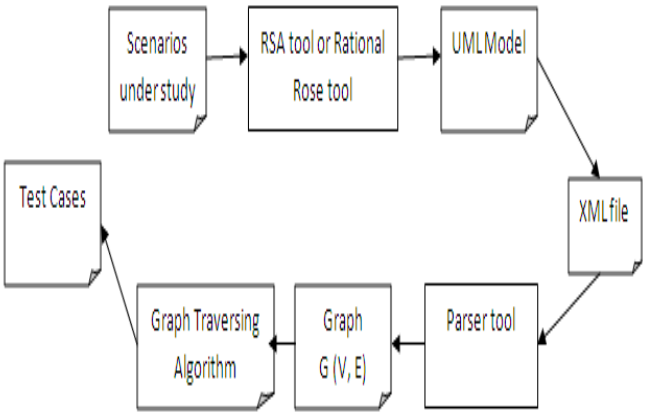

Figure 2- Flow graph for the proposed system

V. CONCLUSION

The proposed system will automate the process of automatic test case generation by making use of rational tool. Automating test case generation will provide greater efficiency as the defects or errors will be detected at an early stage in Software Development Life Cycle (SDLC).

As this system is making use of Rational Software Architect (RSA) portability is enhanced as well as conversion of UML model to various different format is possible. Moreover as we get an XML or XMI interpretation of the UML model we can apply various techniques available on this data and transform the same as per our organization needs.

VI. REFERENCES

[1] Santosh Kumar Swain, Durga Prasad Mohapatra, "Test case generation from Behavioral UML Models", International Journal of Computer Applications (0975-8887), Volume 6-No. 8, September 2010

[2] R. Mall, Fundamentals of software engineering, 2nd ed. New Delhi: Prentice-Hall of India Ltd, 2008.

[3] P. Jalote, An Integrated Approach to Software Engineering, 3rd ed. Springer/Narosa, 2006.

[4] G. Myers, The art of software testing, 2nd ed. Hoboken, New Jersey: JohnWiley & Son, 2004.

[5] R. V. Binder, "Testing Object-Oriented System Models, Patterns, and Tools", NY: Addison-Wesley, 1999.

[6] Johannes Ryser, Martin Glinz, "A scenario-Based Approach to Validating and Testing Software Systems Using Statecharts",12th International Conference on Software and Systems Engineering and their Applicaions ICSSE '99.

[7] Sandeep K. Singh, Sangeeta Sabharwal, J. P. Gupta, "A Novel Approach for deriving test scenarios and test cases from events", Journal of Information Processing Systems, Vol. 8, No. 2, June 2012.

[8] Premal B Nirpal, K. V. Kale, "Using Genetic Algorithm for Automated Efficient Test case generation for Path testing", Int. J. Advance Networking and Application, Volume 02 Issue 06, pp 911-915 (2011).

[9] Yong Chen1, Yong Zhong, Tingting Shi1 and Jingyong Liu, "Comparison of Two Fitness Functions for GA-based Path-Oriented Test Data Generation", 2009 Fifth International Conference on Natural Computation, IEEE, 2009.

[10] G. Booch, J. Rumbaugh, and I. Jacobson, The Unified Modeling Language User Guide. Addison-Wesley, 2001.

[11] P. Frohlich and J. Link, "Automated test cases generation from dynamic models," in Proceedings of the European Conference on Object Oriented Programming. Springer-Verlag, 2000, pp. 472 – 491, lNCS 1850.

[12] M. Riebisch, I. Philippow, and M. Gotze, "Uml-based statistical test case generation," in Proceedings of ECOOP 2003, S. Verlag, Ed. LNCS 2591, 2003, pp. 394 – 411.

[13] P. Samuel, R. Mall, and P. Kanth, "Automatic test case generation from uml communication diagrams," Information and Software Technology, vol. 49, no. 2, pp. 158 – 171,2007.

[14] P. Samuel and R. Mall, "A novel test case design technique using dynamic slicing of uml sequence diagrams," e-Informatica Software Engineering, vol. 2, no. 1, pp. 71 – 92, 2008.

[15] S. Sahni, "Data Structures, Algorithms and Applications in C++", McGraw Hill Press, 2006.