Advances in Computer Science and Information Technology (ACSIT)
Print ISSN: 2393-9907; Online ISSN: 2393-9915; Volume 2, Number 8; April-June, 2015 pp. 6-8
© Krishi Sanskriti Publications
http://www.krishisanskriti.org/acsit.html

A Survey on Model Based Test Case Generation using UML Diagrams

Afreen Ali[1] and Shailja Pandey[2]

[1,2]*Babu Banarasi Das University, Lucknow*
E-mail: [1]afreenali67@gmail.com, [2]*shailjabbd@rediffmail.com*

Abstract—*Software testing is an important phase of Software Development Life Cycle to maintain quality control, performance and to produce high reliable system. At the early phase of software development life cycle (SDLC), nobody including client and engineer can see the product; just at the last phase of the item advancement it is conceivable. Any issues discovered at the last stage, it acquires a ton of expense and time to redress, which is all that much pivotal in any industry. Modeling language like UML allows the visualization of software at early stages of SDLC. Therefore, bugs can be identified early so it serves to optimize the scheduled time and development cost. Developing test cases is an imperative testing artifact. There are often an immense amount of feasible test-cases for testing software, but the fewest test cases with maximum coverage should be the goal. Effective test case should identify the defects in the system and also fulfills 100% coverage criteria. This paper presents a survey on various test case generation techniques that resides in the current literature.*

Keywords*: UML diagrams, test case generation*

1. INTRODUCTION

Software testing is one of the most crucial phase of software development. In software testing, the tester tests the system with well designed inputs with intent to find errors.

The quality and reliability of the end product depend to a large extend on testing. Therefore, more than 50% of software development effort is spent on testing. Testing consists of three steps: 1) Test case generation 2) Test case execution and 3) Test case evaluation. A test case is defined as a triplet set [I, S, O], where I is the data input, S is the state of the system and O is the output from the system. A test case is said to have good coverage if it uncovers maximum number of faults with minimum number of test cases. Combination of test cases required to test a software is called test suite.

Various methods have been used to generate test case automatically, which include formal and semi-formal methods. Formal methods are based on mathematical languages, techniques and tools. They are built based on the structural properties of the System under test (SUT). Hence, they serve the ability to verify the correctness in implementation, discover ambiguity and inconsistency. Major disadvantages of formal method are tool support is insufficient, difficult to implement and lack of expert training.

Semi formal methods are based on models. This method is very popular in current scenario and is used widely in software industries. Sometimes a combination of formal and semi formal methods is also used. Various approaches proposed for automatic test case generation can be found in literature and these have been classified as Model based test case generation, Search based test case generation, Finite State Machine, Path oriented, Goal oriented, etc.

In this paper literature survey is done on various methodologies available for generating test cases from UML models. The paper is organized as: Section 2 gives brief introduction to Unified Modeling Language, Section 3 gives Literature review, Section 4 enlists various UML tools and Section 5 gives Conclusion.

2. UNIFIED MODELING LANGUAGE

The Unified Modeling Language (UML) is a visual modeling language, which is a collective approach of James Rambaugh, Grady Booch and Ivar Jacobson at Rational Software during 1994–95. It was adopted as a standard by the Object Management Group (OMG) in 1997. UML is a tool supported, industry-standardized and process independent modeling language [6]. It is likewise utilized for designing, constructing, documenting and modeling the artifacts of a software system.

UML diagrams are classified into three categories - structural diagram, behavioral diagram and implementation diagram. Structural diagrams are also called static diagrams. They represent the objects of the system in terms of their classes, operations, attributes, interface and relations. It includes class and object diagrams. Behavioral diagrams describe the dynamic behavior of the system in terms of their interaction. It includes use case diagram, activity diagram, sequence diagram, collaboration diagram and state chart diagram. Implementation diagrams depict the physical structure of the system during build and run time. It includes component and deployment diagram.

3. LITERATURE REVIEW

In [1], Ranjita, Vikas and Prafulla have proposed an approach to generate test cases from UML 2.0 activity diagrams. Firstly an activity diagram in drawn in IBM Rational Rose tool. Activity flow graph (AFG) is derived from the activity diagram by traversing from beginning to the end. Then, a required control flow sequence is extracted by traversing the AFG by depth first traversing technique. Finally an algorithm is proposed to generate test paths. A case study on Soft drink vending machine is also presented.

In [2], V. Mary Sumalatha has described various techniques used for software testing. Then, a new technique is proposed to generate test cases from sequence diagram. Initially, a sequence diagram is drawn then the sequence diagram is converted into sequence graph using the proposed algorithm. Weights are assigned to the nodes as the parent weight is the weight of the node. If a node has multiple parents then sum of parent's weight is assigned to the node. Then, genetic algorithm is applied to the sequence graph. All paths between the source and destination are identified. Finally fitness value and probability of individual is calculated. Crossover is performed and fitness value is reevaluated until maximum number of generations is reached or all the paths have been covered or the fitness value minimizes. Best test path is generated. At the end, case study of spider game card is presented.

In [3], Robson and Turner have used black box testing method for testing the interactions between the features of an object and its states. They have focused on testing the state dependent object's behaviors. Object's features are implemented as objects methods. Finite state machine is used to model state dependent object behaviors. And these models generate test cases.

In [4], Hyungchoul Kim et al. have used I/O explicit activity diagram to generate test cases. Activity diagram represent system's dynamic behavior by interaction of different objects among themselves. Firstly they have drawn activity diagram. Then they have generated activity directed graph from activity diagram. This graph is used to generate test cases. The problem of state explosion is avoided by using single stimulus principle. This method reduces time and cost of the software development process without compromising quality.

In [5], P. Samuel et al. have generated test cases using UML state models. They have explored the data flow and control flow logic of state diagrams. They have traversed the state machine graph to identify the conditional predicates on each transition. Then function minimization technique is applied on these predicates to generate the test cases. Generated test cases can be used for testing both class and cluster level behaviors.

In [7] S. Shanmuga Priya proposed an approach to generate test cases from UML sequence diagram. The proposed work is presented taking an example of medial consultation system. Initially, sequence diagram of medical consultation system is drawn using IBM rational rose tool and it is saved with .mdl extension. The stored file is then parsed using java swing to obtain a Sequence Dependency Table (SDT). The objects in SDT are used as nodes to construct Sequence Dependency Graph (SDG). Then, SDG is traversed using depth firth search process to derive the test paths.

In [8], Noraida Ismail et al. proposed automatic test case generation from UML use-case diagrams. GenTCase tool is being built which automatically generates the test cases from system functional requirements. Initially, requirements are transformed into a use case diagram. Then use cases are used to generate test case which can further be used by a programmer to validate system requirements. They paid major attention to reduce the cost of testing the system. The procedure is explained by taking example of a book store system.

In [9], Emanuela et al. have proposed a technique to generate test cases from UML sequence diagrams. Then, these sequence diagrams are translated into Labeled Transition Systems (LTSs) early in the development phase. This reduces the development cost. LTS was traversed using Depth First Search method to derive test paths. An example of Motorola mobile phone application was given.

4. UML TOOLS

Variety of UML tools is widely available. Some open source UML tools are: Agro UML, Umbrello, FUJABA, Astade and Coral. Commercially available modeling tools that support UML Diagrams include IBM Rational Rose, Magic Draw, gModeler, Visual thought, Eclipse UML, Rhapsody, Modelistic, UM-Studio, Smart-Draw, MacA & D, Select Component Architect, Sequence Sketcher, Ecto Set Modeller, Proxy Designer, JVisi on, iUML, Embarcadero Describe, WinA & D, HAT (HOORA Analysis Tool), Visual Paradigm for the Unified Modeling Language (VP-UML), Object Domain, Visual UML and Together. Few drawing tools are: Visio and Omni raffle. Various open source drawing tools are: Violet, DIA, UML etc

5. CONCLUSION

Unified Modeling Language (UML) has now become a defacto model in the field of software testing. New techniques for the generation of test case from these models required to be discovered. This paper presents a literature survey on generating test cases form UML models. Various procedures that have been used for test case generation are been discussed. This paper will help researcher to find out what work has been done in their concerned field.

REFERENCES

[1] Ranjita Kumari Swain, Vikas Panthi, Prafulla Kumar, "*Generation of test cases using activity diagram*", International Journal of Computer Science and Informatics, ISSN (PRINT): 2231 –5292, Volume 3, Issue 2, 2013.

[2] V. Mary Sumalatha, "*Object Oriented Test Case Generation Technique using Genetic Algorithms*", International Journal of Computer Applications (0975 – 8887) Volume 61– No.20, January 2013.

[3] D. J. Robson and C. D. Turner, "*The state-based testing of object-oriented programs*"; in Proceedings of IEEE Conference on Software Maintenance, 1993, pp. 302 – 310.

[4] Hyungchoul Kim, Sungwon Kang, Jongmoon Baik, Inyoung Ko, "*Test Cases Generation from UML Activity Diagrams*", 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/ Distributed Computing 2007, pp. 556-561, doi:10.1109/ SNPD.2007.189.

[5] Samuel R. Mall A.K. Bothra, "*Automatic test case generation using unified modeling language (UML) state diagrams*", IET Software, 2008, Vol. 2, No. 2, pp. 79–93/ doi: 10.1049/iet-sen:2006006179.

[6] Peter Fettke, "*Overview of the Unified Modeling Language*", ISSN 1617-6332, URN urn:nbn:de:0006-0175.

[7] S. Shanmuga Priya, "*Test Path Generation Using Uml Sequence Diagram*", Volume 3, Issue 4, April 2013 ISSN: 2277 128X International Journal of Advanced Research in Computer Science and Software Engineering.

[8] Noraida Ismail, Rosziati Ibrahim, Noraini Ibrahim, *"Automatic Generation of Test Cases from Use-Case Diagram."* Proceedings of the International Conference on Electrical Engineering and Informatics Institute Technology, Bandung, Indonesia, June 2007 pg:17-19.

[9] Emanuela G. Cartaxo, Francisco G. O. Neto and Patrıcia D. L. Machado, "*Test Case Generation by means of UML Sequence Diagrams and Labeled Transition Systems*". Proceedings of the IEEE International Conference on Systems, 7-10, October, Man and Cybernetics, Montréal, Canada 2007.