ISSN 2278-7763

A Survey on Test Case Generation from UML Based Requirement Analysis Model

Mrs. Sulakshana Nagpurkar, Mr. Y.B.Gurav

Padmabhushan Vasantdada Patil Institute of Technology Pune-21, India
Email:snagpurkar@gmail.com
ybgurav@gmail.com

ABSTRACT

This Software testing is an important activity in the software development life cycle One of the important goal of a requirement specification is that the customer can validate the requirements and expectations from the viewpoint of the actual usage .The success of software can be improved by modelling .Modelling is one way to visualize the design and check it against the requirement before the developer begins the coding phase. One such modelling language widely used is unified modelling language (UML). UML diagrams produce a graphical representation of the system under design .If testing the software is addressed at the initial stage in software development lifecycle (SDLC); it will be easy for the testers to test the software in the later stages. This paper limelight the various UML models used for test case generation.

Keywords : Introductio UML diagram, Activity diagram, sequence diagram, state chart diagram

INTRODUCTION

Software testing is an important activity in the Software Development Life Cycle. Many methods that have been used for generating automated test cases are from formal methods, semi-formal methods or an integrated method. Formal Methods are mathematical-based techniques and tools. They are able to be proved and verified as they are built based on the behavioural/structural properties of the System under Test (SUT). This method has the potential to verify whether the system has been implemented correctly, discover the inconsistencies, and ambiguities. But the major downside of formal method oriented test case generation are i) insufficient tool support, ii) lack of expertise/training, and iii) specifications are easy to under-stand, but difficult to create. Semi-Formal methods are model-driven engineering that focus on creating models of a system at each stage in the development life cycle. With a well defined methodology automatic model trans-formations (e.g. to code) from the design level specifications or requirement level specifications are achievable. The major advantages of this method are they are intuitive, and produce more maintainable software.

There are various techniques used in automatic test case generation which includes SBST (Search Based Software Test case generation), FSM (Finite state Machine), MBT (Model Based Testing) etc., this paper focuses on UML based techniques, the key techniques found in the literature are:

UML diagrams, State chart diagrams, sequence diagrams, activity diagrams, class diagrams, collaboration diagrams and a combinational approach of them.

II. UNIFIED MODELING LANGUAGE

The Unified Modeling Language (UML) is a semi-formal modeling language for specifying, constructing and documenting system. The Unified Modeling Language (UML) is a collection of languages for specifying, visualizing, constructing, and documenting the artifacts of software systems [2] Sequence diagrams capture time dependent (temporal) sequences of interactions between objects. Sequence diagrams describe interactions among software components, and thus are naturally considered to be a good source for integration testing. Grady Booch, Ivar Jacobson and James Rumbaugh originally envisage the UML in 1970. UML is the designing of a software application prior to coding and is an important part of large software projects, and helpful to medium and even small projects as well. A model plays the similar role in software development that blueprints and other plans play in the construction of a building. By using a model, those responsible for software development project's success can guarantee themselves that business functionality is complete and correct, end-user needs are met, and program, design, support, requirements for scalability, robustness, security, extendibility, and other characteristics, before implementation of the code renders changes complicated and costly to make.

III. LITERATURE SURVEY

The UML model technique is most widely used in software design phase. The literature review shows that UML diagrams are typically used to automatic generate test cases. A number of approaches for generating test cases in a round-

ISSN XXXX-XXXX

about way from the UML analysis and design models, i.e. use case diagram, sequence diagram, collaboration diagram, class diagram, state chart diagram or a combinational approach are proposed. But most of them need to translate UML description into another description such as a graph (model based testing) or a table and then derive the test cases. In [2], Prasanna et al. have done a survey on automated test case generation techniques categorized under specification based test case generation and model base test case generation. They also touched on few other approaches for generating test cases as path oriented test case generation and intelligent techniques.

Test Case Generation from UML Structured Diagram

In [3], Prasanna and Chandran proposed a new model based approach for automated test cases for UML object diagrams using genetic algorithm. In their work, Object diagram of a banking system created using Rational Rose tool has been considered for test case automation process. The methodology followed was a tree based approach in which initially the object diagram was constructed using rational rose software and it was parsed using java swing to capture the object Names. A tree was built where objects are represented as nodes, attributes (object inputs) forms the left branch of the corresponding node and methods (object outputs) Forms the right branch of the corresponding nodes. In case of duplication, it is added as a prefix for the nodes to form a general tree. Later this tree is changed into binary tree and depth first search technique was applied on it to generate the test case set which gave the valid, invalid and termination sequences for the application In order to prove the effectiveness of their approach, mutation analysis was carried on where faults were injected deliberately to reveal the presence of faults. Their experimental results have shown that it has the capability to reveal 80% fault in the unit level and 88% fault in the integration level.
In [4], Shanthi and Mohan Kumar proposed a technique for generating test cases for object oriented software that uses data mining concepts to design and derive test cases. The data mining technique was applied to generate optimal test cases. The proposed approach used UML Class diagram, in which a tool is used to extract the information from the diagram and are mapped into a tree structure. A cross over operator was applied to discover all patterns from the tree. They followed an approach similar to [3].

Test Case Generation from UML Behavioral Diagram

Noraida Ismail [12] proposed automatic test case generation from UML use-case diagram intended to reduce the cost of testing the system. A simple use case diagram for online bookstore system where only a registered user (customer) is allowed to place an order for available items on the web is considered in their work. They built a GenTCase tool that is able to generate the test cases automatically according to the system's requirements. At first, the requirements are transformed into a UML use-case diagram, and then the test cases were automatically generated according to the use cases. The test cases can be used as a checklist for a programmer to validate the system whether it meets the requirements. The major limitation of the built tool GenTCase is that it captures only the functional requirements of the system where as it fails to capture the non-functional requirements of the system.
Pakinam et al. [6] proposed an automated approach for generating test cases from UML Activity diagram. The proposed model created an activity diagram for ATM withdrawal functionality. The activity diagram is used to generate table called Activity Dependency Table (ADT) and converted it into directed graph called an Activity Dependency Graph (ADG). The branch coverage criterion is applied for covering the path and Depth First Search (DFS) traversal technique is applied on the graph for obtaining all the possible test paths. All the details are added to each test path using the ADT to have the final test cases.
Suppandeep Sandhu and Amardeep Singh [9] proposed a technique for generating test case based on gray box testing with UML Activity Diagrams, traversed in Depth First Search manner. This approach is used to generate test cases directly from UML activity diagram using java programming, and design is reused to avoid the cost of test model creation. Test cases were generated using PETA java/eclipse based platform an automated software testing tool. The tool is highly productive and flexible.

Test Case Generation by a Combinational Approach

Saru Dhir [11] analyzed the UML Structural and Behavior diagrams. He proposed UML Class diagram and
object diagram (structural model) can be used to generate minimal test cases. The technique used is first the class diagram and object diagrams were modeled, then coded, and the test cases are generated from both based on the model information. Different test cases are generated by object diagram and class diagram. A comparison between both these diagrams is made in order to identify the redundant test cases. Then with the help of instrumented code, the information as how many numbers of times the code is tracked is found.

OVERVIEW OF UML MODELING TOOLS

The most well known UML modeling tool is IBM Rational Rose. Factors that should be considered when choosing UML tools are: i) have a clear under standing
of what features are important to create models. ii) Select tools in such a was as it support most UML analysis diagrams such as class diagram, use-case diagram, collaboration diagram, sequence diagram and activity diagram.
iii) Narrow down the tools and try for a demonstration
tool. iv) The tool should be easy to use, reliable and scalable. Enormous amount of UML tools are widely available. Some open source UML modeling tools are: Umbrello, Astade, FUJABA, Agro UML, and Coral. Various open source drawing tools are: DIA, Violet, UML etc. The various commercially available modeling tools that support UML Diagrams are: Magic Draw, IBM Rational Rose,

ISSN XXXX-XXXX

JUDE, gModeler, Rhapsody, Modelistic, Visual thought, Eclipse UML, UM-Studio, Smart-Draw, MetaEdit+,Select Component Architect, Visual Paradigm for the Unified Modeling Language (VP-UML), Sequence Sketcher, EctoSet Modeller,ProxyDesigner, JVision, iUML, Embarcadero Describe, WinA&D, MacA&D, HAT (HOORA Analysis Tool), Object Domain, Visual UML and Together. Few drawing tools are: Omni raffle and Visio.

CONCLUSION

Unified Modeling Language (UML) has now become a defacto model in the field of software testing, particularly in the industry sectors. New technique for the generation of test case from these UML diagrams needs to be identified. Also, the study shows that existing methods mostly concentrates on the behavioral diagrams, in which some does not work for maximum test coverage where as some methods prepare and generate a significant number of tests with less test coverage. Much of the existing techniques very few concentrate on redundant test path generation and no solution have been proposed for eliminating redundant test cases. In future, we have planned to develop a test path method that minimizes size of tests, time and cost, while preserving maximum test coverage using Modified Condition / Decision Coverage criterion.

REFERENCES

[1] Jeevarathinam, Antony Selvadoss Thanamani. 2010. Towards Test Case Generation from Software Specification. International Journal of EngineeringScience and Technology, Vol. 2(11), pp. 6578-6584.

[2] Prasanna M., S. N. Sivanandam, Venkatesan, R. Sundarrajan. 2005. A Survey on Automatic Test Case Generation. Academic Open Internet Journal.

[3] Prasanna M. and K.R. Chandran. 2009. Automatic Test Case for UML Object diagrams using Genetic Algorithm. International Journal on Advance Soft Computing Application, July, Vol. 1, No. 1.

[4] Shanthi A.V.K. and G. Mohan Kumar.2011. Test Cases Generation for Object Oriented Software. Indian Journal of Computer Science and Engineering (IJCSE), Aug-Sep, Vol. 2, No. 4.

[5] Puneet Patel and Nitin N. Patel. 2012. Test Case Formation using UML Activity Diagram. World Journal of Science and Technology, pp. 57-62.

[6] Pakinam N. Boghdady, Nagwa L. Badr, Mohamed Hashem and Mohamed F.Tolba. 2011. A Proposed Test Case Generation Technique Based on Activity Diagrams. International Journal of Engineering and Technology, June, Vol. 11, No. 03.

[7] Pakinam N. Boghdady, Nagwa L. Badr, Mohamed Hashem and Mohamed F.Tolba. 2011. A Proposed Test Case Generation Technique Based on Activity Diagrams. International Journal of Engineering and Technology, June, Vol. 11, No. 03.

[8] Debasish Kundu and Debasis Samanta. 2009. A Novel Approach to Generate Test Cases from UML Activity Diagrams. Journal of Object Technology, June, Vol. 8, No. 3, pp. 65-83.

[9] Saru Dhir, 2012. Impact of UML Techniques in Test Case Generation. International Journal of Engineering Science and Advanced Technology, March- April. Vol. 2, Issue 2, pp. 214-217.

[10] Monalisha Khandai, Arup Abhinna Acharya, Durga Prasad Mohapatra, 2011. A Survey on Test Case Generation from UML Model. International Journal of Computer Science and Informational Technologies, Vol. 2(3).

[11] Vidhi vig, Systematic Review of Automatic Test Case Generation by UML Diagrams Journal of Engineering Research & Technology (IJERT) Vol. 1 Issue 6, August – 2012

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

