ISSN 2277-3061

Generation of test cases using UML models

Gurpreet Singh[1], Rasbir Singh[2]
[1]Research Scholar, RIMT Mandi Gobindgarh, Punjab, India
Preet_gur_grewal@yahoo.co.in
[2]Asst. Prof. RIMT Mandi Gobindgarh, Punjab, India
rasbir.rai@gmail.com

Abstract

Software Testing plays an important role in Software development because it can minimize the development cost. UML is widely used in the software development, there consists of the designing and coding of the software. Designing phase is done with the help of the UML models, which consists of the sequence diagrams, activity diagrams, use case diagrams, state chart diagrams etc. After designing of the system, the next task is coding. Since the software development is time and human resource consuming, the reduction of consumption is done with the help of the code generation automatically. This work mainly focuses on the UML sequence diagram and state chart diagram as the model. We Propose a Technique for Test Case Generation using UML Models. UML models give a lot of information that should not be ignored in testing. An innovative approach of generating test cases from the combination of UML design diagrams have been discussed in this paper. Present work used an approach where sequence diagram and state chart diagram has been used to generate test cases. The test cases thus generated are suitable for dynamic testing of system.

Keywords

BOUML; Code Generation; Java; Sequence Diagram; UML; Test Case Sequence; Model Based Testing; Model Junit; State chart diagram; Test cases.

1. INTRODUCTION

Software Testing is the process of executing a program with the intention of finding errors [1]. Every software code has been reviewed and verified through SQA activities but these activities are not sufficient. Every time the software delivered to the client has been thoroughly tested by client before sending it to the production. Thus developer has to test the software before it gets to the client. Testing has been generally performed by three ways: white- box testing, black box testing and gray-box testing [2-3]. Day by day with increasing functionality of software has caused increasing complexity , cost and size of software applications due to this reason more emphasis has been sited on object oriented design strategy to cut down software cost and boost software reusability. The Object oriented design strategy used designing and implementing software has created new challenges in testing. Object oriented features like polymorphism, abstraction, encapsulation and inheritance has created lot of challenges for tester while testing the software. Thus object-oriented software needs different tactics for testing software effectively during all phases of a development effort. The review of analysis and design models is the base for testing object oriented programs [4, 5]. The testing techniques discussed so far is good to test software what it was supposed to do but it will not test what is missing in software code. From last few years there has been slow development made to the testing of object-oriented systems. One innovative approach is to use UML models for test cases generation. This approach can lead to find out faults in the design phase prior to the development phase and thus causes correction of faults in early life cycle stages of software. This type of testing comes under the category of gray-box testing [6-10].

UML models are nothing but diagrammatical representation of specification document. UML diagrams can be used as a base to derive test cases and to develop testing environments. However, using UML diagrams to derive test case is not an easy task. A test case contains initial state, test sequence, constraints, final state and expected output, where constraints are the pre- and post condition for that input values. Collecting information like pre and post condition from UML diagram is a difficult task. The solution to this problem is to augment the design with the information like pre condition, post condition, input and output related to that system which will cover all aspects of the system to increase the efficiency of the system. However information like these will complicate the automated testing. Several researches have been done to automate test case generation from UML design and this paper also provides contribution to automate the test case generation from UML diagrams [19][21] . It is cumbersome for generating test models like control flow graph from source code.

The UML sequence diagrams are used for modeling discrete behavior of an object through sequence graph. Such states and transitions are critical to decide the specific operation invocations that would be made based on the conditions arising during ascenario execution. For unit level testing, we can derive tests from UML state machine diagrams, which embody the behavioral description of each component [14].

2. Related Work

Lot of work has been done on generating test cases from UML diagram. John D. McGregor, David A. Sykes [11] presented a work to generate test cases on the basis of class diagram. They have used language such as Object Constraint Language/OCL [12] or a natural language, and/or as a state transition diagram to generate test cases for a class. Information from all these diagrams have been accumulated and whichever form is most consistent is used to develop test cases. They used operations of classes to generate test cases and execution-based testing to test the class. In execution based testing assertion checks has been added to the class code to find the bugs. Some more research has been reported to generate test cases based on class [13]. Some research has been reported to generate test cases based on interaction diagram. Sequence diagram along with collaboration diagram comes under modeling called Interaction modeling. Interaction diagram has been used to represent a combination of dynamic and structural modeling where dynamic modeling has been represented by sequence diagram as it gives emphasis on time ordering of messages and structural modeling has been represented by collaboration diagram as it emphasizes on structural organizations of objects those participate in messages communication. The name interaction as such reveals that this type of modeling concentrates on control flow through multiple interacting instances.

For testing these two types of diagrams a control flow graph has been developed that contains multiple entities. Now the entire traditional graph based test coverage techniques can be applied to this control flow graph as outlined in [15]. This includes branch coverage, path coverage and round trip scenario coverage criteria [16]. Since UML diagrams are always more abstract and provides ease to generate test cases than control flow graphs so researchers have started using UML diagrams to generate automated test cases from UML diagrams. Philip Samuel, Rajib Mall and Sandeep Sahoo have presented a novel testing methodology to test object-oriented software based on UML sequence diagrams [17]. This paper has presented an approach to generate test cases automatically from UML sequence diagrams using dynamic slicing technique [17]. With the help of message guards and conditional predicates of sequence diagram dynamic slices have been created. Then these slices were used to generate test cases. A unique approach called slice coverage criterion has been used to validate the test cases. Dynamic slice approach accomplishes sufficient test coverage without excessively increasing the number of test cases. Li Bao-Lin, Li Zhi-shu, Li Qing and Chen Yan Hong have presented a new test cases generation approach that is based on UML sequence diagrams and Object Constraint Language/OCL [18]. The sequence diagram has been transformed to a tree representation. Firstly by selecting conditional predicates from sequence diagram whole constructed tree has been traversed. Then, pre and post conditions have been applied with the help of OCL. OCL alter the conditional predicates on sequence diagram and thus test cases have been generated by applying function minimization technique [18]. Message path coverage and constraint attribute coverage can be achieved from the test cases thus generated for all the objects which are related to the message [18]. Test cases generated by this approach covers class, operations, attributes, data limits and objects based test cases. Where class, attributes, operations based test cases have been generated from class diagram, data limit based test cases have been generated from OCL and

objects from sequence diagram. Monalisa Sarma, Debasish Kundu, Rajib Mall have presented a novel approach of generating test cases from UML design diagrams [19]. In this research attempt, test cases have been derived from SDG (Sequence diagram graph).

Firstly sequence diagram has been altered into a graph called Sequence diagram graph (SDG). SDG nodes have been augmented with different information from use case templates, class diagrams and data dictionary to compose test vectors [19]. Then SDG has been traversed to generate test cases. Test cases thus generated are helpful in detecting interaction faults, scenario faults and system testing can also be achieved. State chart diagram and activity diagram are two diagrams through which UML supports behavioral modeling. Every object respond on the receipt of event and state chart diagram represent the behavior of an object by specifying how that object responds to the particular event. State based testing can be used to detect correct implementation of component's state model. Test cases generated from state chart diagram concentrates on individual states of object and transitions between different states. P. Samuel R. Mall A.K. Bothra has developed a novel method to automatically generate test cases based on UML state models [20]. This research paper has presented an attempt to generate automated test cases using sequence diagram and state chart diagram to represent the behavior of the system.

3. IMPLEMENTATION METHODOLOGY

The UML sequence diagram is the input to the system, as the model itself is not textual format an intermediate representation of the diagram is needed. For this XMI format is used, which is having the metadata information of the design model. The XMI representation of the UML sequence diagram is generated with the help of a tool called BOUML [8], in which the XMI generated for java [9]. Since this tool have the capability of supporting UML 2.x version and also XMI exporting feature, it is choose for the UML modeling.

4. Implementation Steps:

1. UML sequence diagram and state chart diagram is modeled in BOUML.

2. Export the XMI representation of the sequence diagram and state chart diagram from the tool for java language.

3. Extract the metadata from the XMI file.

4. Then transfer UML diagram designs into Graph form.

5. Create java code.

6. Generate test cases.

Implementation step 1 and 2 can be done with the help of the BOUML tool, which is available in the internet freely. From this XML file metadata can be extracted. After that it is easy to transfer these designs into graph which is helpful in developing java code. With the help of this java code it is easy to generate test cases of the system.

5. CONCLUSION

As the demand of the new software increase in the current field of software engineering, new technics and methods are needed for fulfilling the needs of the market. In order to make this possible in the case of design model, in which the models are converted to the corresponding code, the code generation approach is used. The work, deals with the generation of the code for the UML sequence diagram and state chart diagram, with the help of the XMI file of the corresponding sequence diagram and state chart diagram using the BOUML tool. Simple sequence flow can be extracted from the XMI file efficiently, but the conditional and looping statements are avoided. But these statements can be included into the sequence by complex programing methods with the help of this XMI file generated.

REFERENCES

1. Abdurazik and J. Offutt. "Using UML collaboration diagrams for static checking and test Generation". In International Conference on the Unified Modeling Language (UML 2000), York, UK, October 2000, pp 383- 395.
2. Amit Kumar and Rajesh Bhatia, Testing functional requirements using B model specifications. ACM SIGSOFT Software Engineering Notes. Volume 35 Issue 2, March 2010, Pages 1-7.
3. Beizer. "Black-Box Testing, Techniques for Functional Testing of Software and systems."Wiley, New York, 1995.
4. Blanco, R., Fanjul, J.G., Tuya, J.: Test case generation for transition-pair coverage using Scatter Search. International Journal of Software Engineering and Its Applications 4(4) (October2010).
5. Gill, M.S., Bhatia, R.K.: Formal Specification Based Software Testing: An Automated Approach; In Software Engineering Research and Practice (2003) 656-659.
6. H.-G. Gross. Measuring Evolutionary Testability of Real-Time Software PhD thesis, University of Glam organ, Pontypridd, Wales, UK, June 2000.
7. IEEE. Standard Glossary of Software Engineering Terminology, Volume IEEE Std. 610.12- 1990.IEEE, 1999.
8. J. Hartmann, C. Imoberdorf, and M. Meisinger. "UML-based integration testing". In International Symposium on Software Testing and Analysis (ISSTA 2000), Portland, USA, August 2000, pp 60 -70.
9. John D. McGregor, David A. Sykes "A Practical Guide to Testing Object-Oriented Software", Addison Wesley, March 05, 2001, pp. 167.
10. Jos Warmer and Anneke Kleppe. "The Object Constraint Language: Precise Modeling with UML".Boston, MA: Addison-Wesley. 1999.

11. J. Offutt and A. Abdurazik. “Generating tests from UML specifications”. In International Conference on the Unified Modeling Language (UML 1999), Fort Collins, USA, October 1999, pp 416-429.
12. Li Bao-Lin, Li Zhi-shu, Li Qing, Chen Yan Hong, “Test Case automate Generation From UML Sequence diagram and OCL expression” School of Computer Sichuan University, Chengdu 610064, China. pp 1048-52.
13. Monalisa Sarma Debasish Kundu Rajib Mall, “Automatic Test Case Generation from UML Sequence Diagrams”, 15th International Conference on Advanced Computing and Communications. pp 60-64.
14. Myers, Glen ford J. The art of software testing / Glen ford J. Myers; Revised and updated by Tom Badgett and Todd Thomas, with Corey Sandler.—2nd ed.p.cm. ISBN 0-471-46912-2 pp 6.
15. P.Samuel R. Mall A.K. Bothra, “Automatic test case generation using unified modeling language (UML) state diagrams”, IET Softw., 2008, Vol. 2, No. 2, pp. 79–93/doi: 10.1049/iet-sen: 20060061.
16. Philip Samuel, Rajib Mall and Sandeep Sahoo, “UML Sequence diagram Based Testing Using Slicing”, IEEE Indicon 2005 Conference, Chennai, India, 11-13 Dec. 2005, pp 176-178
17. Q. Nguyen Hung Testing Application on the Web: Test Planning for Internet-Based Systems John Wiley & Sons 2003.
18. R. Heckel and M. Lohmann. “Towards model-driven testing”. Electronic Notes In Theoretical Computer Science, 82(6), 2003, pp 33-4.
19. R. Binder. “Testing Object-Oriented Systems: Models, Patterns and Tools”. Addison-Wesley, 2000.
20. R. S. Pressman. “Software Engineering: A Practitioner’s Approach”, 6rd Edition, McGraw Hill, New York, 2005, pp. 424, 434, 449.
21. Supaporn Kansomkeat and Wanchai Rivepiboon “Automated-Generating Test Case Using UML State chart Diagrams” Proceedings of SAICSIT 2003 [23] Jonathan Gennick and Sanjay Mishra, Oracle SQL*Loader the Definitive Guide, O’Reilly & Associates, Inc., April 2001, pp 296-300.
22. Wang Linzhang, Yuan Jiesong, Yu Xiaofeng, Hu Jun, Li Xuandong and Zheng Guoliang “Generating Test Cases from UML Activity Diagram based on Gray-Box Method” Proceedings of the 11th Asia-Pacific Software Engineering Conference (APSEC’04) pp 284-291.
23. Y.G. Kim, H.S. Hong, D.H. Bae, and S.D. Cha. “Test cases generation from UML state Diagrams”. IEEE Proceedings Software, 146(4), 1999, pp 187-192.