A Novel approach for Model-Based Software Testing

Jyoti Tiwari
Department of Computer Engineering
Shri GS Institute of Technology Indore, India
jyotitiwariscs@gmail.com

ABSTRACT: Software Testing is an important and critical phase in software development that deals with software quality. Testing of software requires a great deal of planning and resources. Model-based testing is a testing approach where test cases are automatically generated from models. The models are the expected behaviour of the system under test and can be used to represent the testing strategy. Model-Based Testing (MBT) gaining its importance in an industry as well as academics. In Scenario-based testing, test scenarios are used to generate test cases, test drivers etc. Model-based testing adopts architectural models of a system under test or its environment to derive test cases.

1. INTRODUCTION

Software Engineering is a systematic approach to the development, operation, maintenance, and retirement of software. It is a discipline that provides methods and tools for the construction of quality software within budget and time in the context of constantly changing requirements. Improving software productivity and quality is an important goal of software engineering, now a day's software plays a critical role in businesses, government, and societies. The ever-increasing demand for software in all aspects of day to day life has led to examine the critical research issues and challenges in software engineering. Software testing is one of the significant research domains in software engineering.

"Software Testing is the process of executing a program with the intent of finding the error"[1]. It plays a very vital role in the software industries and it is the inevitable phase of the software development lifecycle. Often, the testing phase is the single largest contributor towards the whole development time. Testing of software requires a great deal of planning and resources as it is a time-consuming activity. Software testing mainly focuses on test case generation, test execution and evaluation [2]. Testing can be performed on requirement, design, and code. Testing can be started a process of validating and verifying the product which meets the requirements that guided design and development; if it works as expected, can be implemented.

Software testing approaches mainly divided into code-based testing, specification-based testing, and model-based testing. Structural testing is also known as code-based testing. Structural testing is dependent on the coding phase and being initiated as soon as the code becomes available in the initial stage of software development. This type of testing detects the error of commission after the implementation. Test cases are constructed based on the code that implements the software. The output of each test case must be determined by the specification, but input can have determined by analyzing the code itself for the execution path to be taken. The main advantage of this approach is improved coverage [3].

The significant advantages of testing analysis and design models are test cases can be identified earlier in the process. Even as requirements are being determined. Early testing helps analytics and designer to better understand and express requirements and to ensure that specified requirements are "testable". Bugs can be detected early in the development process, saving time, money and effort. Test cases can be reviewed for correctness early in a project. The correctness of test cases especially system test cases is always an issue[3]. It is characterizing by programmer, tester, and manager that work principles, the relationship

between design and requirements, the frequency of specification change, and requirements are important in analyzing and design phase for selecting the testing techniques [4].

2. SOFTWARE TESTING APPROACHES

2.1 Specification-Based Testing

Specification-based testing acquires information from a specification or scenario of the system under test, rather from implementation. Specifications are the output of an analysis and design phase in the textual form generated from the application. Specification-based testing is an automated software testing method based on algebraic specifications, it is also called conformance testing. Confirmation testing starts with writing specification in some formal notation followed by the implementation of the system under test. Test cases generated from the specification and applied to the implementation.

Test cases are generated directly from the UML behavioral diagram, where the design is reused. Firstly, scenarios from activity diagram are generated and then each scenario corresponding sequence and a class diagram is generated. After that sequence diagram is analyzing to identify interaction category and a class diagram is to find setting category. After analyzing each category, its significant values and constraints are generated, and respective test cases are derived. By using this approach defects in the design can be detected during the analysis of the model itself [21].

2.2 Model-Based Testing

UML diagrams like activity diagrams, sequence diagrams, class diagram, and state chart diagram can be represented as graphs at the higher level of abstraction of a system. In the graph representation test cases is the path of the graph that specifies some functional requirement and data. Data satisfy some conditions on the path while traversing from start node to end node. Test case generation from graphs compromises certain steps: identify the requirements, build a graph, identify test requirements, select paths to cover those requirements.

Graph theory is a field of mathematics that helps to derive test cases from the design model in different ways. Here researchers first convert UML models to graphs and then identify the test data because the UML models are graphical representations of the system specification, they cannot directly useful for deriving the test cases.

2.2.1 Tree Based methods

A tree is a connected acyclic graph. In tree sequence of vertices and edges beginning at the root and end at the leaf node. Condition classification tree method is an approach to generate test cases from activity diagrams (AD). The classification tree is generated based on the guard condition in an activity diagram. Decision point in AD is represented as a category node whereas transitions of decision point are used to create child nodes. A test case table is created and then test cases are generated by using a series of condition classification trees, that conditions are considered one by one from left to right. The result shows that test cases generated by this approach have a strong ability to detect faults [8]. Extenics is a discipline for modeling contradiction problems with formalized methods and transformation. The activity diagram is transformed into Euler circuit based on transformation theory and test cases are generated automatically by the Euler circuit algorithm. This can maximize the fault detection and minimize test cases [9].

2.2.2 Graph-Based methods

A graph is an ordered pair G = (V, E) where V is set of vertices or nodes and E is set of edges. In a directed graph edge e = (x, y) is directed from x to y. from the testing perspective it is important to search paths in a graph presentation of the UML model, and test data values that solve constraints along the path.

An activity diagram is translated into an activity graph to generate the test cases. Genpath algorithm based on business flow controls approach is used to generate all activity path (test cases) which can ensure coverage for both loop testing and concurrency from activity diagram. Here modified DFS is used to generate test cases to avoid path explosion [11].

Activity diagram converted into activity diagram graph and sequence diagram translated to sequence diagram graph. After that modifies depth-first search (DFS) is used to generate test cases from each graph. On the next step system testing graph is formed by combining the activity diagram graph and sequence diagram graph. The necessary information to form the test cases is pre-stored in the graph to generate the testing base on that system testing graph. The modified depth-first search first traversed the graph to find each last node in the graph, then stored those nodes in a stack for the last node. After that the graph is traversed again from the start to every node that is stored in the stack, every time the algorithm finds the node that has more than one branch, that node is stored in a stack for multiple branches. When the last node is found the current path is stored in a stack for the path and change the current path to the next path until each last node is visited. Comparison result shows that test cases from combined UML diagram may not necessarily result in better test cases, due to redundant test cases for some test scenario [10].

2.2.3 Heuristic-Based methods

In UML model most commonly used meta-heuristic techniques are genetic algorithm and ant colony optimization. The application of genetic algorithm in search technique is promising because exhaustive testing is infeasible considering the size and complexity of the system under test. Genetic algorithm imitates the process of natural evaluation. It is commonly used to generate useful solutions for optimization. A genetic algorithm evolves a population of candidate solutions, and the fitness function to achieve given coverage criteria [15].

The generation of optimized test cases plays an important role in software development because it can minimize development cost. In the state chart diagram one state can lead us to multiple states which lead us to the optimum result. A sequence diagram covers the maximum number of interaction between the objects during the operation. In this paper, they convert state chat diagram into state chart graph (SCG) and sequence diagram into a sequence graph (SG) ant the integrate these two graphs into system testing graph (SYTG). After generating SYTG optimized test cases are generated using the genetic algorithm. So, it will cover the maximum number of test cases for all possibilities and able so solve faults like integration, scenario faults and pre-post condition faults [16].

The cost of software is too high if we want to test all the test cases. An approach to prioritizing the test scenarios by using the genetic algorithm and local search strategy derived from the UML activity diagram. The test scenario is generated early in the development process which helps to find out many errors in the development phase and prioritize test scenarios can increase the probability of early fault detection, improve efficiency and reduce the cost of software testing [17].

2.3 Direct UML Specification Processing

Agent-based regression test cases are generated using the UML class diagram, use case diagram & activity diagram. In this paper, they use eclipse and ArgoUML to draw all the diagrams after that they convert these diagrams into XML and XMI format and JADE is used for designing agents. Agent parses XML/XMI files old and modified UML diagrams and compares them. These agents submit changes to the repository after collecting these changes they categories test cases based on obsolete, retestable and reusable test cases. Their approach reduces the required time for regression test generation but not using any formal specification

3. DISCUSSION AND CONCLUSION

In general, exhaustive testing is not practical or tractable for all programs due to a large number of possible inputs and sequences of operations. As a result, selecting the set of test cases which can detect possible errors of the system is the main challenge in software testing. MBT aims to automate the generation and execution of test cases using models based on system requirements and behavior. MBT provides several important advantages such as improved test coverage and fault detection, reduced testing time, and reduction in cost. So far, various different models have been adopted to derive the test cases in MBT. As a modeling language, UML is widely used to describe analysis and design specifications by both academic and industry, thus UML model is a good source for test case generating.

4. REFERENCES

[1] Aditya P. Mathur, "Foundations of Software Testing" 2nd Edition by Pearson,

[2] Dr.LeetevathiRajamanicham, "Testing Tool for Object-Oriented Software", Internation Journal of Scientific Research and management, 2014

[3] JOHN D. McGregor, DAVID A. SYKES, "A practical guide to testing object-oriented software",3rd edition by Addison-Wesley, 2001

[5] Sanjeev Patwa, Anubha Jain, "Effect of Analysis and Design Phase factors on Testing of Object-Oriented Software", IEEE, 2016.

[6] P.D. Ratna Raju, Suresh Cheekaty, HarishbabuKalidasu, "Object oriented Software Testing", International Journal of Computer Science and Information Technology, Vol. 2, 2011.

[7] Rajvir Singh, "Test Case Generation for Object Oriented System: A Review", IEEE, Internation Conference on Communication Systems and Network Technologies, 2014.

[8] SupapornKansomkeat, Jeff Offutt, "Generating Test Cases from UML Activity Diagram using the Condition-Classification Tree Method", IEEE, 2nd International Conference on Software Technology and Engineering, 2010.

[9] Liping Li, Xingsen Li, Tao He, JieXiong, "Extenics-based Test Case Generation for UML Activity Diagram", ELSEVIER, Ptocedia Computer Science, 2013.

[10] Meiliana,Irwandhi, Ricky, Daniel, "Automated Test Case Generation from UML Activity Diagram and Sequence Diagram using Depth First Search Algorithm", ELSEVIER, ICCSCI, 2017.

[11] Walaithip, Suwatchai, Luepol, "Generating Test Cases from UML Activity Diagram Based on Business Flow Control", ACM, ICNCC, 2016

[12] Ranjita K. Swain, Vikas Panthi, Prafulla K. Behera, "Generation of Test Cases using Activity Diagram", Internation Journal of Computer Science and Informatics, vol. 3, Issue-2, 2013.

[13] Xinying Wang, Xiajun Jiang, Huibin Shi, "Prioritization of Test Scenarios using Hybrid Genetic Algorithm Based on UML Activity Diagram", IEEE Software, 2015.

[14] Pankaj Jalote, "An Integrated approach to Software Engineering", 3rd Edition, India: Narosa Publishing House, 2006.

[15] Altaf Hussain, Aamer Nadeem, M. Touseef Khan, "Review of Formalizing Use Cases and Scenarios: Scenario Based Testing", IEEE, 2015.

[16] Baikuntha Biswal, Prgyan Nanda, Durga Mohapatra, "A Novel Approach for Scenario-Based Test Case Generation", IEEE, International Conference on Information Technology, 2008.

[17] Md khalid Hussain, Dr.Krisna Prasad, "A Literature Survey for Test Case Generation Using UML Model", International Journal of Computer Science Trends and Technology (IJCST), Vol- 4, 2016.

[18] Paramjit Kaur, Rupinder Kaur, "Approaches for Generating Test Cases Automatically to Test the Software", International Journal of Engineering and Advanced Technology, 2013.