A Survey on Test Case Generation Techniques Using UML Diagrams

Nisha Rathee[1], Rajender Singh Chhillar[2*]

[1] Department of Information Technology, IGDTUW Delhi, India.
[2] Department of Computer Science and Applications , MDU Rohtak, Haryana, India.

* Corresponding author. Tel.: +91-8199023624; email: Chhillar02@gmail.com
Manuscript submitted January 10, 2016; accepted March 8, 2016.
doi: 10.17706/jsw.12.8.643-648

Abstract: Software testing is the most important step for the development of software. Software programs are executed against a number of test cases to check for efficiency and quality, by comparing actual output results with expected results. Model based testing allows the generation of test cases at early stage of software development even before coding. Early detection of faults during the software design phase reduces product developmental and testing costs. In this paper, we present a survey of various object oriented testing techniques which have been implemented for the generation of test cases, and for prioritizing the generated test cases using uml diagrams.

Key words: Activity diagram, class diagram, object diagram, sequence diagram, uml diagram.

1. Introduction

Software testing is the most time intensive and important task of software development life cycle (SDLC) [23]. It consumes almost 40% - 70% of the software system development resources [18], [21]. The main aim of software testing is to reveal maximum detection of faults in software systems using minimal test cases. Automated software testing can reduce time and costs associated with testing. In recent years, various object oriented techniques have been proposed for automatic generation of test cases.

1.1. UML Diagrams and Model Based Testing

Unified Modeling Language (UML) was released by Object Management Group (OMG) in 1997 [24]. Object oriented paradigm has rapidly come into existence in industries and academics for the development of software systems. Uml is used for designing, modeling and documenting the object oriented software systems.

Uml diagrams are classified into fourteen diagrams on the basis of two category levels: a) the structural level (class diagram, object diagram, package diagram, composite diagram, deployment diagram and profile diagram) and b) the behavioral level (use case diagram, state machine diagram, activity diagram, sequence diagram, communication diagram, interaction overview diagram and timing diagram). Structural diagrams show how different components of the system are related to each other on different levels of abstraction whereas behavioral diagrams show dynamic behavior of objects including temporal changes in objects, and their interactions with each other.

Model-based testing is an object oriented testing technique where test cases are generated from models using uml diagrams. Models preserve essential information from requirement specifications and are the basis for final implementation [2]. Test cases can be extracted from uml diagrams and their combinations, for testing all the features of software systems. Model based testing can be done at the design phase and test cases generated here, help in early detection of faults in the system. It also helps in building confidence in developers, and reducing development costs and effort. Model based testing is more efficient and effective than a code based approach to software testing, as it is a mixed approach between source code and specification requirements [22].

The rest of the paper is organized as follows: section 2, describes the literature review on test case formation using uml behavioral and structural diagrams and section 3, describes literature review on prioritizing the generated test cases. Section 4 of the paper describes the conclusion and future work.

2. Literature Review

The most important part of software development is test case generation. Uml is most generally used to describe design specifications and it has been analyzed through literature review that uml is also widely used for test case generations. Literature review shows that in recent years various techniques have been proposed for generation of test cases from Uml diagrams and their combinations. Literature review also shows that a lot of research has also been done in test case optimization and prioritization. We have discussed some of the existing techniques in this paper.

2.1. Test Case Formation Using UML Behavioral and Structural Diagrams

Supaporn Kansomkeat, Phachayanee Thiket and Jeff Offutt [1] have proposed a condition- classification tree method for the generation of test cases using uml activity diagram for case study of SALES (sales order processing system for a retail shop). In this technique, an activity diagram is first converted into condition classification trees by gathering control flow information from every decision point, and guard conditions, in the activity diagram. A test case table has been generated based on these trees for generation of test cases, in which each row of the table represents a test case. Experimental data shows that test cases generated by condition classification tree method (CCTM) are relatively small in number and have a strong capability to detect faults.

Ajay Kumar Jena, Santosh kumar Swain and Durga Prasad Mohapatra [2] proposed a model for the generation of test cases using uml activity diagram in which an activity diagram has converted into an activity flow graph (AFG). AFG is traversed using depth first search (DFS) and activity coverage criterion for the generation of test paths. Test cases are then generated from these test paths. Genetic algorithms have been applied for optimization of test cases. The case study is being done on ATM withdrawal system.

Pakinam Nagwa, L. Badr, Mohamed Hashem and Mohamed F.Tolba [3] proposed a method for generation of test cases from uml activity diagrams. They have proposed an algorithm which automatically generates a table called activity dependency table (ADT) and then an activity dependency graph (ADG) has been created from this table. Branch coverage criterion and depth first search (DFS) traversal technique is applied on the ADG for generation of final test cases. This proposed technique also includes validation of test cases using cyclomatic complexity, which assures that all branches are covered and test cases are efficient.

Preeti Gulia and R.S. Chillar [4] proposed a new approach for the generation and optimization of test cases using genetic algorithms from uml state chart diagrams, on the case study of driverless train. All the possible test paths are generated from the uml state diagram. For the generation of optimized test cases, they have randomly selected few test path sequences and applied genetic algorithms crossover technique. Efficiency of test cases has been evaluated using mutation analysis.

Chanda Chouhan, Vivek Shrivastava and Parminder S Sodhi [5] have proposed a test case generation based on activity diagram (TCBAD) model for the generation of test cases from the case study of contact field of mobile applications. This model constructs an activity diagram of mobile application and then forms an ADT which includes the following columns: activity name, dependency nodes, in degree value, dependent nodes, out degree values. An ADG is automatically generated from the table. This graph has been traversed using depth first search, for the generation of all possible test paths. Cyclomatic complexity is used for validation of test cases.

Ridham Khurana and Anju Saha [6] have presented a comparative study of five test case generation techniques using uml activity diagram on ten examples. Many techniques have been proposed for generation of test cases using activity diagram. In this paper, these techniques are applied on the same input, so that testers can have a brief idea of differences and similarities among these techniques. The result could help them to make a decision upon that which technique is helpful depending upon the phase and test data for test case generation.

Lizhe Chen and Qiang Li [7] have proposed an approach i.e interaction finite automation (IFA), a state based model for formalization of system's behavior from UseCases, without knowing its internal structures. The proposed approach includes transformation of usecases to IFA and then generates test cases from IFA. This proposed approach is implemented in Automatic Test Case Generation Tool (ATCGT) using the case study of sale processing system.

Syed Usman Ahmed, Sneha Anil Sahare and Alfia Ahmed [8] have proposed a technique for the generation of test cases from uml collaboration diagrams using prim's and dijkstra's algorithms on the case study of online examination system. The generated test cases cover branch and decision coverage cyclomatic complexity.

Ranjita Kumari Swain,Prafulla Kumar Behera and Durga Prasad Mohapatra [9] have presented an approach named Test Generation and Minimization for O-O software with State Charts (TeGeMiOOSc) using case study of Railway Ticket Vending Machine (RTVM). State chart diagrams are converted in to state transition graphs. The graph is traversed using depth search approach and test cases are generated from the graph. Test cases are minimized by calculating node coverage for each test case. Generated test cases satisfy path coverage criterion minimizing cost and time.

M. Prasanna and K.R. Chandran [10] have presented a new approach for the automated generation of test cases from object diagrams using genetic algorithms tree crossover technique on the case study of the banking system. The proposed approach constructs a uml object diagram using rational rose software and saves the file with .mdl extension. This file has been parsed for getting the object names and relation between these objects for the generation of tree structures based on this information. The genetic algorithms tree crossover technique has been applied for the generation of new tree structures. These new generation of trees is converted into binary trees. All test case sequences are generated by applying depth first search on binary tree structures. Effectiveness of the test cases has been analyzed by applying mutation testing technique. The mutation testing yielded 80.3% effectiveness in the process of generating test cases.

A.V.K and G. Mohan Kumar [11] have proposed a new approach using data mining concepts for the generation of test cases which are superior, less complex and easy to be implemented in any testing system. The proposed technique generates uml class diagrams using rational rose software, which is then saved as a .mdl extension. Tree structures are formed by extracting all the information from this file. Genetic algorithms are applied at two-point crossover technique for new generation of trees and the trees are converted into binary trees. The depth first search method is finally applied for the generation of test cases. Efficiency of test cases is analyzed by mutation technique of testing.

2.2. Test Case Formation Using UML Combinational Diagrams

Anbunathan and Anirban Basu [12] have presented a method for the generation of test cases from uml class diagrams using basis path test case generation approach. A corresponding state chart diagram has been drawn from the class diagram. This diagram is then converted into a control flow graph. Decision-to-decision (DD) paths are derived from the graph using basis path test case generation technique. Test cases are generated manually as well as automatically, and effectiveness of test cases has been performed using mutation analysis. Results show a significant decrease in cost as compared to other existing techniques.

Baikuntha Narayan Biswal, Soubhagya Sankar Barpanda and Durga Prasad Mohapatra [13] have applied genetic algorithms constraint based technique on uml activity and collaboration diagram for the generation of optimized test cases using the case study of ATM cash withdrawal. They have considered transition coverage as a test adequacy criterion. Input values are defined in the form of a set of sequence of events and a pass/fail (Boolean Variable such as "1" for pass and "0" for fail) technique has been deployed for the evaluation of test cases generated. This proposed technique guarantees the minimal error in test cases.

Vinaya Sawant and Ketan Shah [14] have presented a novel technique for generation of test cases from uml usecase diagrams, class diagrams and sequence diagrams, and then transform it in to Sequence Diagram Graph (SDG). A data dictionary is presented in the form of Object Constrained Language (OCL). The uml diagrams are drawn with the help of magic draw tool and then exported to XML format. The XML file has been parsed in java for extracting different nodes of the graph and generates all set of the scenarios from start node to end nodes. Now these set of scenarios along with the usecase template and OCL data dictionary are traversed using breadth-first search for the generation of test cases. The case study of bank ATM systems is considered for the generation of test cases.

Supaporn Kansomkeat, Jeff Offutt, Aynur Abdurazik, and Andrea Baldini [15] have performed an experiment for revealing the fault finding capabilities of model-based test cases using uml state and sequence diagrams. The experiment includes a comparative analysis of test cases generated from these two diagrams. The result shows that large numbers of test cases are generated from a uml state chart diagram i.e. 81 test cases, whereas only 43 test cases are generated from uml sequence diagram. It has been also analyzed that sequence diagrams did better at revealing integration level faults and state chart diagrams did better at revealing unit level faults. This result shows that different uml diagrams perform different roles in testing. A case study of a cell phone handset system has been considered. The experiment was performed using java programming language.

Santosh Kumar Swain, Durga Prasad Mohapatra and Rajib Mall [16] have proposed a novel technique, State Activity Test case Generation (SATEC), for test case generation using uml activity and state chart diagrams. Activity and state chart diagrams are converted in to state – activity diagram (SAD) which contains information from both diagrams. SAD is made up of various components like, state–activity node, and-or node, decision node, start node and end node. Depth first search traversing technique has been used for generation of test cases. The proposed technique is helpful in revealing integration faults. Case study of bank ATM login system is considered for generation of test cases from SAD.

Swagatika Dalai, Arup Abhinna Acharya and Durga Prasad Mohapatra [17] have proposed an approach for generation of test cases for concurrent object oriented software systems using the combinational features of uml sequence and activity diagrams. The diagrams are first converted into graphs and then an integrated graph called sequence-activity graph (SAG) has been generated by integrating the features of these two diagrams into one. The graph is traversed using "graph traversing algorithm". Wherever a fork node is encountered, breadth first search graph traversing algorithm is applied and while the depth first search graph traversing algorithm is applied for rest of the nodes. Activity path coverage criterion is used for the generation of test cases. The generated test cases have better coverage and fault detection capabilities.

3. Prioritization of Test Cases

Preeti Kaur, Priti Bansal and Ritu Sibal [18] have proposed an approach for the prioritization of test cases generated from uml activity diagrams on the case study of shipping order system. An activity diagram is first converted into activity flow graph after which test scenarios are generated from this graph using the DFS traversing method. Prioritization of test cases is done using path complexity which includes path length, information flow metric, predicate node and logical conditions traversed. The proposed approach analyzed that by prioritizing test cases, a tester can concentrate more on high priority test cases by testing the critical test case sequences first. This also helps in detecting high number of faults in relatively less time.

Rishi Seth and Sanyam Anand [19] have proposed a technique for prioritization of test cases from uml sequence diagrams on the basis of a) prioritization on depending upon depth, b) prioritization on dependency of number of parameters, c) prioritization upon depending on code coverage and finally, the combination of all. The case study considered is facebook authorization. Priority is set according to these parameters and test cases with the highest priority are tested first.

L.Fernandez and S. Misra [20] have proposed an approach for the generation of a complete set of functional test scenarios from uml activity diagram. Test cases have been prioritized by considering software risk information as the main parameter. Risk information is gathered by considering all the factors associated with risk and effect of these parameters on risk is also taken into account.

Sangeeta Sabharwal, Ritu Sibal and Chayanika Sharma [21] have proposed a technique for prioritization of test cases by identifying critical path cluster scenarios with the help of genetic algorithms and information flow (IF) metric model using uml activity and state chart diagrams. Concurrent activities in the activity diagram have been considered. The proposed approach also takes care of change in requirements using a stack-based approach, by assigning weights to nodes of activity and state chart diagram.

4. Conclusion and Future Work

This paper provides an overview of various test case generation techniques using single uml diagrams, combinational uml diagrams and prioritization of the generated test cases. This paper will help researchers to identify what work has been done in their respective fields. The study shows that existing methods mostly concentrate only on the combination of behavioral diagrams. This approach does not work for a significant number of test cases with maximum test coverage. New techniques need to be identified for the generation of test cases from combinational uml diagrams. These techniques should include features of both behavioral and structural diagrams, so that they will have maximum coverage criteria and maximum fault detection capability as compared to existing systems. Of the various proposed techniques, very few concentrate on elimination of redundant test cases and automatic generation of test cases. It has been analyzed that some improvement needs to be done for optimization of test cases as this will reduce developmental costs and time. In the future, we have planned to develop an efficient test path method with minimum test case size/cost, and maximum code coverage, by integrating or improving the existing condition or decision coverage criterion.

References

[1] Supaporn, K., Phachayanee, T., & Jeff, O. (2010). Generating test cases from uml activity diagram the condition classification tree method. *Proceedings of the 2nd International Conference on Software Technology and Engineering (ICSTE)@ IEEE.*

[2] Ajay, K. J., Santosh, K. S., Durga, P. M. (2014). A novel approach for test case generation from uml activity diagram. *Issues and Challenges in Intelligent Computing Techniques (ICICT), International conference on IEEE.*

[3] Pakinam, N. B., Nagwa, L. B., Mohamed, H., & Mohamed, F. T. (2011). A proposed test case generation technique based on activity digrams. *International Journal of Engineering and Technology IJET-IJENS.*

[4] Preeti, G., & Chillar, R. S. (2012). A new approach to generate and optimize test cases for uml state diagram using genetic algorithm, *ACM SIGSOFT Software Engineering Notes.*

[5] Chanda, C., Vivek, S., & Parminder, S. S. (2012). Test case generation based on activity diagram for mobile application. *International Journal of Computer Applications.*

[6] Ridham, K., & Anju, S. (2012). Empirical comparison of test data generation techniques using activity diagrams. *International Journal of Computer Applications .*

[7] Chen, L. H., & Qiang, L. (2010). Automated test case generation from use case: A model based approach.

[8] Syed, U. A., Sneha, A. S., & Alfia, A. (2012). Automatic test case generation using collaboration UML diagrams. *World Journal of Science and Technology.*

[9] Ranjita, K. S., Prafulla, K. B., & Durga, P. M. (2012). Minimal testcase generation for object-oriented software with state charts. *International Journal of Software Engineering and Applications.*

[10] Prasanna, M., & Chandran, K. R. (2009). Automatic test case generation for uml object diagrams using genetic algorithms. *Int. J. Advance. Soft Comput. Appl.*

[11] Shanthi, A. V. K. & Kumar, G. M. (2011). Automated test case generation for object oriented software. *Indian Journal of Computer Science and Engineering.*

[12] Anbunathan, R. & Anirban, B. (2013). Dataflow test case generation from UML Class diagrams. *International Conference on Computational Intelligence and Computing Research.*

[13] Baikuntha, N. B., Soubhagya, S. B., & Durga, P. M. (2010). A novel approach for optimized test case generation using activity and collaboration diagram. *International Journal of Computer Applications.*

[14] Vinaya, S., & Ketan, S. (2011). Automatic generation of test cases from UML models. *Proceedings of the*

International Conference on Technology Systems and Management (ICTSM).

[15] Supaporn, K., Jeff, O., Aynur, A., & Andrea, B. (2010). A comparative evaluation of tests generated from different uml diagrams. *Proceedings of the Ninth ACIS International Conference on Software Engineering(2009).*

[16] Santosh, K. S., Durga, P. M., & Rajib, M. (2010). Test case generation based on state and activity models. *Journal of Object Technology Published by ETH Zurich, Chair of Software Engineering, JOT.*

[17] Swagatika, D., Arup, A. A., & Durga, P. M. (2011). Test case generation for concurrent object-oriented systems using combinational uml models. *(IJACSA) International Journal of Advanced Computer Science and Applications.*

[18] Preeti, K., Priti, B., & Ritu, S. (2012). Prioritization of test scenarios derived from UML activity diagram using path complexity. *CUBE Pune, Maharashtra, India*.

[19] Rishi, S., & Sanyam, A. (2012). Prioritization of test cases scenarios derived from uml diagrams. *International Journal of Computer Applications.*

[20] Fernandez-Sanz, L., & Misra, S. (2012). Practical application of UML activity diagrams for the generation of test cases. *Proceedings of the Romanian Academy, Series A.*

[21] Sangeeta, S., Ritu, S., & Chayanika, S. (2011). Applying genetic algorithm for prioritization of test case scenarios derived from UML diagrams. *IJCSI International Journal of Computer Science Issues.*

[22] Ali, S., Briand, L. C., Jaffar-ur-Rehman, M., Asghar, H., Zafar, Z., & Nadeem, A. (2007). A state based approach to integration testing based on UML models. *Journal of information and Software Technology.*

[23] Velur, R., Arun, B., & Satanik, P. (2008). Efficient software test case generation using genetic algorithm based graph theory. *Proceedings of the International Conference on Emerging Trends in Engineering and Technology.*

[24] Mall, R. (2009). Fundamentals of software engineering.

Nisha Rathee has done her graduation and post-graduation from the Maharishi Dayanand University, Rohtak, Haryana. Now she is working as an assistant professor in Indira Gandhi Delhi Technical University For Women, Delhi, India. Her research interests include software engineering, soft computing and software testing. She has a total of more than five years of teaching experience.

Rajender Singh Chhillar is a professor at the Department of Computer Science at Maharshi Dayanand University, Rohtak, Haryana, India. During his service in Maharshi Dayanand University, Rohtak. He served as the director at the University Institute of Engineering and Technology; a director at the Computer Centre. He served as the head at the Department of Computer Science, Chairman, a board of studies; a member, an executive and an academic councils. His research interests include software engineering, software testing, software metrics, web metrics, bio metrics, data warehouse and data mining, computer networking, and software design. He has published more than 91 journal and 65 conference papers over the last several years and has also written two books in the fields of Software Engineering and Information Technology. He has visited many countries including France, Hong Kong, China, U.K, Dubai and Nepal. He also won the best paper award in International Conference ICCEE- 2013 held in Paris, France during October 12-13, 2013.

Professor Chhillar is a director of board, CMAI Asia association, New Delhi and a senior member of IACSIT, Singapore and a member of Computer Society of India. Professor Chhillar has been serving as an editorial board member, guest editor and reviewer of multiple international journals, and serving as a program committee chair, keynote speaker and session chair of multiple international conferences. He also performs advisory work to Government agencies and academic bodies.