MODEL BASED TESTING APPROACH FOR AUTOMATIC TEST CASE GENERATION AND OPTIMIZATION USING GENETIC ALGORITHM

[1]Sridhar Yemul, [2]Kapil Vhtakar, [3]Vipul Bag
[1,2,3] Nagesh Karajagi Orcihd College of Engineeing & Tehcnology,Solapur,
[1]shridharyemul@gmail.com

Objectives:

Model based testing approach for both test case generation and test case optimization for object-oriented software using UML diagrams. The proposed approach addresses the issue of redundancy, size of test cases and optimization challenges. Automation of test case design process can result in significant reductions in time and effort, and at the same time it can help in improving reliability of the software through increased test coverage. The proposed approach uses genetic algorithm from which best test cases can be optimized. Moreover our method for test case generation inspires the developers to improve the design quality.

2. Literature Survey:

Software testing is the process of exercising a program with well-designed input data with the intent of observing failures. Software testing addresses the problem of effectively finding the difference between expected behavior specified by the system models and the observed behavior of the implemented system. At present, software testing on the average makes up as much as 40% to 60% of the total development cost and would increase even further with rapidly increase size and complexity of software. As systems are getting larger and more complex, the time and effort required for testing are expected to increase even further. Therefore, automatic software testing has become an urgent practical necessity to reduce testing cost and time .Our research is focused on the use of UML models for the above said purpose i.e. automatic test case generation and test case optimization. We found so many researchers who have already

worked in UML for test case generation [5]. Basic concepts on Object-Oriented Software Testing (OOST) strategy and use of UML diagrams as test model are covered. We also discuss some basic concepts of genetic algorithm (GA) and optimization. We gone through some research papers those have covered extensively the various aspects of object-oriented software testing using UML diagrams

State diagrams or state chart diagrams are used to help the developer better understand any complex functionality or depict the dynamic behavior of the entire system, or a sub-system, or even a single object in a system. GA can be used to generate the test data using UML state chart diagram as described by Biswal [3]. Sometimes after coding developers don't have time to test the software. Generating test cases from UML state chart diagram can solve this problem by generating them before the coding. Then the test cases can be generated as per the specifications of the software. Specifications can be in the form of UML diagrams, formal language specifications or natural language description. Software testing efficiency is optimized by identifying critical path clusters [8]. The test case scenarios are derived from activity diagram. The activity diagram is converted into CFG where each node represents an activity and the edges of the flow graph depict the control flow of the activities. It may be very tedious expensive and time consuming to achieve this goal due to various reasons. For example, there can exist infinite paths when a CFG has loops. GA has been applied in many optimization problems for generating test plans for functionality testing, feasible test cases and in many other areas [9], [10]. GA has also been used in model based test case generation [3], [10]. Various techniques have been proposed for generating test data/test cases automatically using GA in structural testing [2], [5]. GA has also been applied in the regression testing, object oriented unit testing as well as in the black box testing for the automatic generation of the test cases [13].

2.1. Existing System:

2.1.1. Basic Scheme

Several approaches to design test cases and application of Genetic Algorithm on software testing have been proposed by researchers. These approaches include generation of test cases from requirements specification i.e. black box testing or from code i.e. white box testing or from model-based specification [11]. Test case generation solely based on requirements specification completely ignores system implementation aspects. Further, it cannot reveal whether the software performed some tasks which are not specified in the requirement specifications. On the other hand, test case design from program code is cumbersome and difficult to automate. Besides, it also cannot reveal missing functionalities [2].

Further, the traditional system testing techniques black box as well as white box testing, achieves poor state coverage in many systems. The reason being that the system state information is very difficult to identify either from the requirement specifications or from the code [3]. An automated model-driven test case framework is therefore desirable. At the same time we have studied on optimization of generated test cases.

3. Problem Statement

Model Based testing approach for both test case generation and test case optimization to achieve the goals, described below:

- To propose a generalized technique for optimized test case generation for object oriented software using UML behavior model.
- To implement the proposed methods to evaluate its efficiency and effectiveness using the test coverage criteria of white box testing.

3. 1 Conclusion from Literature Survey:

Model-based testing which uses UML design specifications for test case generation overcomes these short comings and has emerged as a promising testing method. Further, models being simplified representations of systems are more amenable for use in automated test case generation . Automation of test case design process can result in significant reductions in time and effort, and at the same time it can help in achieving an increased reliability of the software through increased test coverage**.**

3.2. Necessity of Proposed System:

- Lacking in generalization and automation. These are two basic pit falls of present procedure to software testing. When it being automated it fails to generalize only for that particular application it fits.
- The open problem is to generate an automatic and optimized test case with existing approaches.
- Optimized test cases is not only helpful in quick the testing process but also cost saving. It is also essential to differentiate among the various test cases. It defines the clear cut objective in front of the tester. No needs to go for different test cases, which may serve different objectives, in turn save the time and money.

4. Methodology:

Our proposed approach to generate test case generation from UML behavior model and then, we optimize test covarge while minimizing time and cost. GA is a search technique based on natural genetic and evolution mechanisms which can be used to solve many categories of problems in machine learning and function optimization. . Transition coverage is used as

test adequacy criteria in this approach. The schematic representation of our approach is shown in Figure.1.Our proposed methodology involves the following steps.

Step 1.Analyze the real system which is to be tested and accepted by User.

Step 2.Construct the UML dynamic diagram using rational rose software and store with .mdl extension input to parser.

Step 3: Convert the given diagram into an intermediate graph using parser by analyzing and collecting all information about the object state, action and transition.

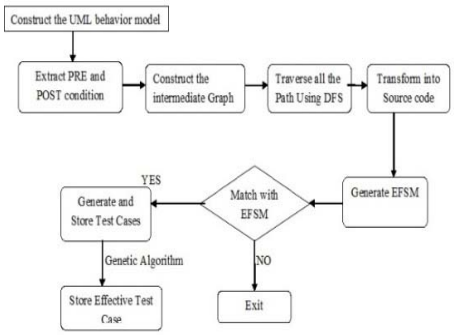
Step 5: Graph is transformed into the source code in java. Hence the design is implemented using java swing.

Step 6: Generate EFSM for graphical representation as graph where states are represented as node and transition as directed edge between states.

Step 7: Starting from first node, traverse remaining node, using DFS in order to form test sequence.

Step 8: Obtain all the valid and invalid sequences of the application until final edge is reached.

Step 9: Minimization of test cases by calculating node coverage for each test sequences to design model and program under testing.



.

6. Expected Result:

1) To generate test cases would be optimal and efficient.

2) To address testability, coverage criteria and automation issues in order to fully support system testing activities.

7. References:

[1] M.Prasanna, S.N.Sivanandam, Venkatesan, R.Sundarrajan, 15, 2005,"A SURVEY ON AUTOMATIC TEST CASE GENERATION", Academic Open Internet Journal.

[2] M.Prasanna, K.R. Chandran, ” Automatic Test Case Generation for UML Object diagrams using Genetic Algorithm”, Int. J. Advance. Soft Comput. Appl., Vol. 1, No. 1, July 2009

[3] Baikuntha Narayan Biswal, Pragyan Nanda, Durga Prasad Mohapatra, 2008 IEEE, "A Novel Approach for Scenario-Based Test Case Generation”, International Conference on Information Technology.

[4] Sangeeta Sabharwal, RituSibal, Chayanika Sharma, “ Applying Genetic Algorithm for Prioritization of Test Case Scenarios Derived from UML Diagrams”, International Journal of Computer Science Issues, vol.8, issue 3, no.2, May 2011, pp. 433-444.

[5] Ashalatha Nayak, Debasis Samanta: "Automatic Test Data Synthesis using UML Sequence Diagrams", in Journal of Object Technology, vol. 09, no. 2, March{April 2010, pp. 75

[6] Li Bao-Lin, Li Zhi-shu, Li Qing, Chen Yan Hong ," Test Case automate Generation from UML Sequence diagram and OCL Expression", International Conference on Computational Intelligence and Security 2007, pp 1048-52.

[7] Monalisa Sarma , Debasish Kundu Rajib Mall, "Automatic Test Case Generation from UML Sequence Diagrams", 15th International Conference on Advanced Computing and Communications 2007, pp 60-65.

[8] P. Samuel, R. Mall, A.K. Bothra,2008 "Automatic test case generation using unified modeling language (UML) state diagrams ", Published in IET Software.

[9] Z. Michalewicz. Genetic algorithms + data structures = evolution programs (3rd ed.). Springer-Verlag, London, UK, 1996.

[10] M. Mitchell. An Introduction to Genetic Algorithms. MIT Press, Cambridge, MA, USA, 1998.

[11]Abdul Rauf et. al., "Automated GUI test coverage analysis using GA", Seventh international conference on information technology. IEEE, 2010, pp. 1057-1063.

[12]Bo Zhang, Chen Wang, "Automatic generation of test data for path testing by adaptive genetic simulated annealing algorithm", IEEE, 2011, pp. 38 – 42.

[13]Chartchai Doungsa et. al., "An automatic test data generation from UML state diagram using genetic algorithm",http://eastwest.inf.brad.ac.uk/document/public ation/Doungsa-ard-SKIMA.pdf, Accessed on 25.10.2012.

[14]D.J Berndt, A. Watkins, "High volume software testing using genetic algorithms", Proceedings of the 38t h International Conference on system sciences (9), IEEE, 2005, pp. 1- 9.