*International Journal of Latest Trends in Engineering and Technology*
*Vol.(10)Issue(1), pp.202-206*
*DOI: http://dx.doi.org/10.21172/1.101.35*
*e-ISSN:2278-621X*

# SMART TEST PATHOPTIMIZATIONFROM STRUCTURAL UML USING GENETICALGORITHMS

Sukhjinder Kaur[1], Neha Sikka[2], Dishant Khosla[3]

**Abstract:Software Testing has been significantly changed in these days. Testing is the process of checking a system or its modules (m) to find whether it satisfying the specific requirements or not. Testing is executing a system in order to identify any gaps, errors, or missingrequirements in contrary to the actual requirements. Our paper will aim at improving the real time techniques which are applied by testing engineers to increase the quality of software systems. We will increase the quality of test cases which are applied to check the accuracy of software. Our technique will automatically generate the test cases from unified modeling language constructs.**
**Keywords:Umbrello, GA (Genetic Algorithm), Test Case, UML Sequence Diagram, Test Scripts, Test Suite**

## 1. INTRODUCTION

Testing has become the most expensive task of a software project. As per the survey, the testing process took over 40% of the project cost. Testing is also mainly related to costs related to poor quality, as malfunctioning programs and errors generate bigger expenses to software engineers. Therefore increasing the reliability of software and effectively of the testing process could be considered as a right way to decrease software costs for the longer time. Another way for improving the quality of software testing had been applying auto generation to the procedure of the whole testing plan. In our approach, testing engineers will concentrate on the time crucial software features. In testing cost of employees and time limitation of the project are the two major constraints so it sounds like a future benefit to do automation of testing to generate full path coverage.

We will make it easy to use human resources more efficiently, which again may contribute to more detail testing. This will save the overall cost of developing and testing of the software product.As per the market research,different software companies worldwide did the investment of 931 million dollars in software testing tools in 1999 for auto generation of test cases.

So due to the blunder of software testing failures test automation is going to be considered as a main factor of the test process development by the software organizations. The testing procedure is basically divided into manual testing and automated testing.

Automation has been mostly applied to running repetitive tasks such as unit testing or regression testing, where test cases will be executed every time changes will be updated [7]. Main tasks of test automation systems will include the development and running of test programs and verification of test outputs. If we compare with manual testing, automated testing will not be suitable for modules in which there will be small repetition. For those activities manual testing will be more preferable, as building automation is an extensive task and it will be feasible only if the case is run for the many times. The testing of the use cases of the software is a biggest concern for the testing engineers, because each piece of code will be made poorly enough to be impossible to test it, therefore making it not capable for automation.

### *1.1 UML Modeling 2.0*

It was mandatory to differtiate between the UML model and the various other flowcharts of a software. A flowchart is a partial graphic representation of software's model. The combination of various diagrams will not completely cover the model and removing a diagram will not change the model. The system may also contain documentation that will run the model elements and diagram and various use cases..

UML diagrams gives two different views of a software model:

- Structural part: focuses on the static structure of the system using objects, attributes, operations and relationships. It will include class diagrams and structure diagrams.
- Behavioral view: focuses on the dynamic behavior of the software by displaying collaborations among objects and changes to the internal states of objects. This view will use sequence diagrams, activity diagrams and state machine diagrams.

UML models will be be exchanged to UML tools by using the XML Metadata Interchange (XMI) format.

[1] Assistant Professor, CSE, CGC COE, Mohali
[2] Assistant Professor, CSE, CGC COE, Mohali
[3] Assistant Professor, CSE, CGC COE, Mohali

In UML, one of the key tools for behavior modelling is the use-case model. Use cases are a way of specifying requirements of a system. they will be used to capture the requirements of a system, that is, what a system is supposed to do.

### *1.2. Genetic Algorithm*

The classical evolutionary algorithms, includes genetic algorithms, are successfully accepted in various kinds of optimization problems. Genetic algorithm is one of the most commonly used evolutionary algorithms in the past which was firstly given in the 1970s. GA has beeninspired from the process of searching and selection process which will lead to the survival of the fittest individuals. Researchers in the literature have demonstrated the efficiency of adopting GA in solving various optimization problems such as data mining and network traffic control problems. In fact, GA has been adopted in the past to solve some problems and washybridized with other techniques for better enhanced results. GA will be hybridized with local and heuristic search techniques such as Tabu search. various researchers hybridized it with other evolutionary approaches like dynamic programming. Our paper will try to fill these gaps and present a genetic algorithm (GA) to solve the test generation problems.

### *1.3. Automatic Test path optimization*

Test cases involve a set of steps, conditions, and inputs that will be used while performing testing tasks. The main purpose of this activity is to ensure whether a software passes or fails in terms of its functionality and other parameters. There are many types of test cases such as functional, negative, error, logical test cases, physical test cases, UI test cases, etc. Furthermore, test cases will be written to keep track of the testing coverage of a software. An Optimization will not only test linear test sequences. All the test sequence generation and running, tests several inputs with the SUT.It will compute the output of the SUT for a whole set of input vectors and will check whether each of the correspondingly running test sequences will be right. This can be possible, since the SUT will be required to provide one method to generate the next output of the SUT without making changes its state. If an error is found, the test sequence which provoked the error will be returned along with a fail verdict and will terminate. If no error finds out, then the test sequence will be continued with one of the already tested inputs.

### *1.4 Umbrella*

Umbrello is a UML modeling program for the KED that supports a wide range of diagrams, code export, and reverse engineering. It is an open source modeling tool that is based on the technology of KDE. It lets the diagram makers and designers create the diagrams for the software, applications for other systems in a standard format document support system to the designing own independent structure for their programs.

## 2. PROPOSED OBJECTIVES

In this paper we will focus to find out the solution of threetypes of testing scenarios. The testing procedure generates only those test cases that will have three objectives.The prime focus isto cover completepath coverage of error coverage and the second is to take minimum time to find bugs with the focus to decrease the effort and cost. The problem will be handled by employing meta-heuristic inspired genetic algorithm optimization.

## 3. RESEARCH PROBLEM FORMULATION

The test case auto generation process is the process that automatically generates a new set of test cases from two elements:(i)a test suite composed of an initial set of test cases (those corresponding to the use cases that represent the behavior of the GUI);(ii)an special test case called Annotation Test Case which contains all the annotations corresponding to the widgets of a GUI. The process will follows these steps:(1) The process is based on an initial test suite and an Annotation Test Case. Both together will make up the initial Annotated Test Suite.(2)The test case autogeneration process explores all the base use cases. For each use case, it will output all the possible variations depending on the values previously given in the annotations.

Enriching the research area and using the combination i.e both model based testing and genetic algorithm will give a new technique for smart generation of test cases.

## 4. RESEARCH METHDOLOGY

The proposed algorithm is implemented with java program. XMI representation of structural UML will be given to the program and corresponding test cases are generated with the help of proposed algorithm. Figure 1 shows the Flowchart of Research Methodology.

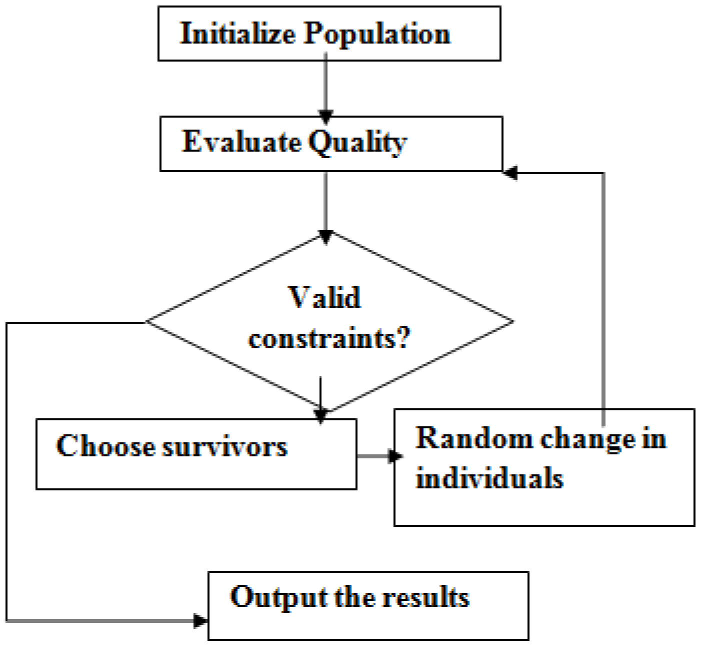

Fig 1. Operation of Test Case Generation

.

## 5. RESULTS AND ANALYSIS

Analysis shows that the technique will find out the exact bugs of mutant procedure.The results of the methodology of generating test cases for the given structural diagram and optimization using genetic algorithm are given below:

### *5.1. Structual Diagram*

Various types of tools had been usedfor drawing UMLs. We will use umbrello to draw UML diagram.

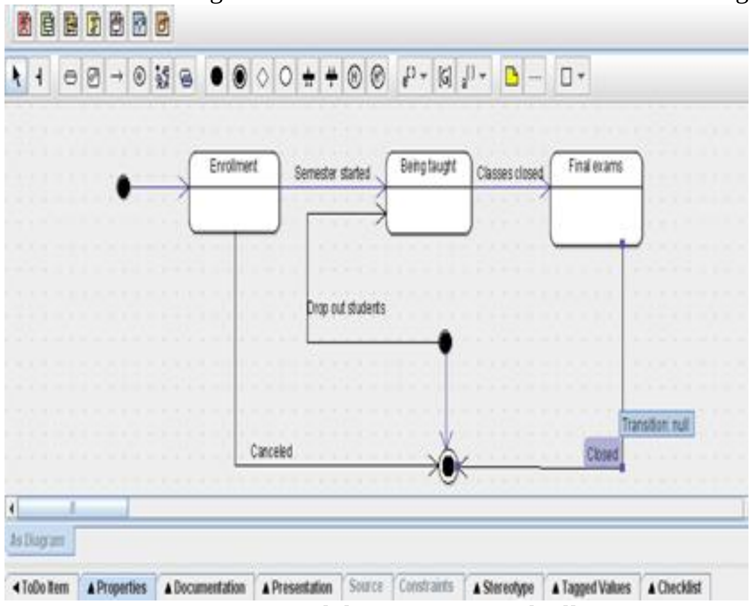

Fig 2.Structural diagram using umbrello

### *5.2. Conversion into XMI*

UML models are usually exported from UML modeling tools in XMI representation. Unfortunately, construction of control flow graph from XMI representation of interaction diagram is not straightforward. The factors contributing to the complexity of construction of control flow graph are: (i) model elements of interaction diagrams are captured in XMI by means of values of attributes of multiple tagged elements, (ii) correlations among tagged elements are not explicitly specified , and (iii) control flow information is modeled by means of different types of fragments that can be nested fragment with arbitrary nesting depth.

### *5.3. Calculating the Quality Factor*

These use cases (modeled as a set of initial test cases) are annotated by the tester to indicate interesting variations in widget values (ranges, valid or invalid values) and validation rules with expected results. Once the use cases are annotated, this approach uses the new defined values and validation rules to automatically generate new test cases and validation points, easily expanding the test coverage. Also, the process allows narrowing the GUI model testing to precisely identify the set of GUI elements, interactions, and values the tester is interested in.

Table 2: Path coverage after first GA

| INPUT | Quality factor for first GA | Quality factor for 2nd GA |
|---|---|---|
| 1,2000,1,1000,101,4 | 6.4 | 6.4 |
| 2,3000,1,2000,100,5 | 8.1 | 7.4 |
| 1,1000,1,-1500,100,3 | 7.4 | 7.2 |
| 2,2000,1,-1500,101,5 | 6.4 | 6.2 |
| 1,2000,2,2000,100,5 | 7.2 | 7.4 |
| 2,3000,2,2000,101,5 | 5.4 | 5.6 |

The test case autogeneration process will seen, in a test case level, in the form of the construction of a tree. This tree will initially represent a test case composed of a series of test cases from which a new branch will be added for each new value defined in the annotations. The validation rules are incorporated later as validation points.

In our approach, modeling the GUI and the application behavior will not involve making a model and using all the GUI elements and producing a large amount of test cases coveringall the possible event modules. It will work by defining a set of test cases and representing the most crucial GUI elements to include both interesting values and a set of validation rules so that we can support the test case auto generation process. It is also not necessary to manually verify, fix, or complete any model in this approach, which removes this error-prone process from the GUI Testing process and makes the work easy for the testing team. These features will help to improve the maintenance of the software system.

### *5.4. The execution and validation process*

is the process by which the test cases (auto-generated in the last step) are executed over the target GUI and the validation rules are asserted to check whether the constraints are met. The test case execution process executes all the test cases in order. It is very important that for each test case is going to be executed, the GUI must be reset to its initial state in order to ensure that all the test cases are launched and executed under the same conditions.

Table 6: Mutants Generated for the Program

| Section of the program | Mutant generated | Alive | Dead |
|---|---|---|---|
| Declaration section | 721 | 687 | 34 |
| Conditional section | 114 | 60 | 54 |
| Typecasting | 54 | 43 | 11 |
| Input and output | 48 | 46 | 2 |
| Validations | 165 | 160 | 5 |

## 6. CONCLUSIONS AND FUTURE WORK

This approach can effectively identify the test path that must be tested first. The advantages of the proposed approach are as follows: firstly, test scenarios can be generated early in the development process. Secondly, it helps us to find out many problems in design phase, that is, before the program is implemented. Thirdly, test scenarios are prioritized in such a way that it can increase the probability of early fault detection.

## 7. REFERENCES

1. S. Rajesh Kumar, Ojha. Deeptimantaand Mohapatra. Durgaprasad, "Automated test case generation and Optimization", International Journal of Computer Science and IT, Vol. 8 Issue. 5, 2016.
2. Rogstad, Erik and Briand, Lionel, “Cost effective strategies for the regression testing of database applications: Case study and lessons learned,” Journal of Systems and Software, vol. 113, pp 257-274, 2016.
3. Chengying Mao, Lichuan Xiao, Xinxin Yu, and Jinfu Chen, "Adapting Ant Colony Optimization to Generate Test Data For Software Structural Testing," Swarm and Evolutionary Computation, vol. 20, pp. 23-36, 2015.

4. Michael Felderer and Andrea Herrmann, "Manual test Case Derivation from UML Activity Diagram and State Machines: A Controlled Experiment," Information Software Technology, vol. 61, pp. 1-15, 2015.
5. M. G. Epitropakis, S. Yoo, M. Harman, and E. K. Burke, “Empirical Evaluation of Pareto Efficient Multi-objective Regression Test Case Prioritisation,” in Proceedings of the 2015 International Symposium on Software Testing and Analysis, ser. ISSTA 2015. New York, NY, USA: ACM, pp. 234– 245, 2015.
6. P. S. Kochhar, F. Thung, and D. Lo, “Code coverage and test suite effectiveness: Empirical study with real bugs in large systems,” in 22nd IEEE International Conference on Software Analysis, Evolution, and Reengineering, SANER 2015, Montreal, QC, Canada, March 2-6, 2015, 2015, pp. 560–564.
7. J. Hass, A. Mette,” A guide to Advanced Software Testing”, Artech House Publications, 2014.
8. A. Orso and G. Rothermel, “Software Testing: A Research Travelogue (2000–2014),” in Proceedings of the on Future of Software Engineering, ser. FOSE 2014. New York, NY, USA: ACM, pp. 117–132, 2014.
9. R. Gopinath, C. Jensen, and A. Groce, “Code Coverage for Suite Evaluation by Developers,” in Proceedings of the 36th International Conference on Software Engineering, ser. ICSE 2014. New York, NY, USA: ACM, pp. 72–82, 2014.
10. Q. Luo, F. Hariri, L. Eloussi, and D. Marinov, “An Empirical Analysis of Flaky Tests,” in Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering, ser. FSE 2014. New York, NY, USA: ACM, pp. 643–653, 2014.
11. D. Gong and Y. Zhang, "Generating test data for both path coverage and fault detection using genetic algorithms", Frontier Computer Science, vol. 7, pp. 822–837, 2013.
12. "Introduction to ArgoUML" available at URL:http://argouml.tigris.org.
13. "UML State chart Diagram" available at URL:
http://www.conceptdraw.com/examples/uml-state chart-diagram.