ISSN 2250-0987
Shanthi et al, UNIASCIT, Vol 2 (2), 2012, 222-224

A NOVEL APPROACH FOR GENERATION OF TEST CASES USING TABU SEARCH ALGORITHM

A.V.KShanthi[#1], Dr. G. Mohan kumar[*2]

[#] *Research Scholar, Sathyabama University, Chennai, TamilNadu, India*
[*] *Principal, Park College of Engineering, Coimbatore, India*

***Abstract*— This Test case generation based on design specifications is an important part of testing processes. Improvements in software testing techniques are needed to address the increasingly complex applications that are implemented by today's software systems. The Unified Modelling Language (UML) is gaining wide acceptance in industry, and they are seeking ways in which UML diagrams can be used as the basis for functional testing. Our approach focuses on developing effective technique and tool for test case generation and coupling them with suitable execution tool for unit, integration and system testing. Unified Modelling Language - Class diagrams are used as design specifications. By setting up several tests adequacy criteria with respect to class diagrams, an automatic approach is presented to generate test cases for Object oriented programs using Data mining searching techniques. In this approach, we first instrument a Java program under testing according to its class diagram model, and generate abundant test cases using Tabu Search. Then, by running the instrumented program we obtain the corresponding program execution traces. Finally, by matching these traces with the behaviour of the class diagram, a reduced set of test cases are selected according to the given test adequacy criterion. This approach can also be used to check the consistency between the program execution traces and the behaviour of class diagrams.**

***Keywords*— Test Cases, Class Diagram, UML (Unified Modelling Language), Data Mining, Tabu search Algorithm.**

I. INTRODUCTION

Improvements in software testing techniques are needed to address the increasingly complex applications that are implemented by today's software systems. The UML and its diagrams are widely used to visually depict the static structure and more importantly for us, the dynamic behaviour of such applications. This trend provides us with an excellent opportunity to meld our proven test generation technology (TDE) with the UML and give developers and testers the capability of automatically generating black-box conformance tests early on. Testing is an important part of quality assurance in the software life-cycle. As the complexity and the size of software systems grow, more and more time and manpower are required for testing. Manual testing is so labour-intensive and error-prone that it is necessary to employ automatic testing techniques in some circumstances.

Data mining techniques support automatic exploration of data and attempts to source out patterns and trends in the data and also infer rules from these patterns which will help the user to support review and examine decisions in some related business or scientific area.

In this paper, we use UML Class diagrams as design specifications, and consider test case generation for Java programs. The state of arts of UML model-based test case generation mainly focuses on generating test cases directly from various UML models by applying specification-based testing and/or code-based testing approaches. However, those direct approaches are hardly implemented in a fully automatic fashion because of the following reasons. First, only abstract test cases can directly be generated from the models, which specify the system under constructing. They cannot be used directly in program testing without manual concretization. Second, for dynamic models, the test cases are generated by traversing the paths of models to derive the test scenarios. The loops and branches in the models make the path conditions very complicated. It leads to algorithms with high complexity, even undesirable problems. In this paper, For deriving test cases from UML Class diagrams, we present a novel approach which selects test cases from a set of generated test cases according to a given coverage criterion concerning the Class diagram specification.

II. OBJECTIVE

New model based approach for automated generation of test cases in object oriented systems has been analysed. The test cases are derived by analysing the dynamic behaviour of the class due to internal and external stimuli. The UML and its diagrams are widely used to visually depict the static structure and more importantly for us, the dynamic behaviour of such applications gives developers and testers the capability of automatically generating black-box conformance tests early on. The scope of the Tool has been limited to the class diagrams taken from the Unified Modelling Language model of the system. Class diagrams are the backbone of almost every object oriented method, including UML. They describe the static structure of a system. Evolutionary Tabu search Algorithm and Neighbourhood search method has been proposed to bring out all possible test cases of a given class diagram. Specification-based testing uses information derived from a specification to assist testing as well as to develop program. Testing activities consist

of designing test cases that are a sequence of inputs, executing the program with test cases, and examining the results produced by this execution. Testing can be created earlier in the development process so the developer will often find inconsistency and ambiguity in the specification and so the specification can be improved before the program is written.

III. METHODOLOGY

Construct a class diagram for any software to developed using Rational rose software. UML Model file will be saving with an extension .MDL. Write Parser in java to extract all the possible information from file. From the extracted information, group the attributes and methods for their respective class and also look for the relationship between the classes to form a tree structure. If a class contains more than one aggregated, associated, specialized, dependent class, draw tree structure for one set of associated class, then move to other such class for forming new tree. Tree structure can be formed such way the class name occupies in next to start node and left side with its attributes and right with its method. Apply Tabu search Algorithm to all the obtained trees. Neighbourhood search method used to obtain the fittest tree. Since the tree structure is not in the form of binary, covert the obtained tree to binary tree by moving class of present class in the tree to occupy its last method present in its present class. Then apply Distant Allocation Searching method to all the binary trees formed, to generate all possible and valid test cases. Efficiency of test cases can be analysed with help of Mutation Analysis.

- Extraction
- Mapping
- Tree Formation
- Binary Tree Formation
- Test Cases Generation
- Test cases Evaluation
- Best Test Case

IV. EXISTING TESTING TECHNIQUE

- ✓ Test Cases are generated with help of Object, Sequence, Activity, Collaboration, State-Chart diagrams. Numerous applications developed with help activity diagram to generate test cases.
- ✓ Lacking in generalization and automation. These are two basic pit falls of present procedure to software testing. Basically testing done manually even for test cases generation. When it is being automated it fails to be generalized only for that particular or some application it fits.
- ✓ Validation of obtained test cases is manual. In case automated system obtained test cases should be validated manually to use such test cases to analysis the software.
- ✓ Test Cases are generated based on code by which the application being developed.
- ✓ Existing system based on code analysis certainly depend on language used and application.
- ✓ Generalization is quite difficult.

V. PROPOSED TESTING TECHNIQUE

- ➢ Since test case generation from design specifications has the added advantage of allowing test cases to be available early in the software development cycle, UML Class diagram is used.
- ➢ Implementation of data mining technique – Tabu search algorithm, to generate optimal test cases.
- ➢ Validation of test cases is automated.
- ➢ Test cases can be easily generated in case of regression testing.
- ➢ Reduce complexity to analysis the code.

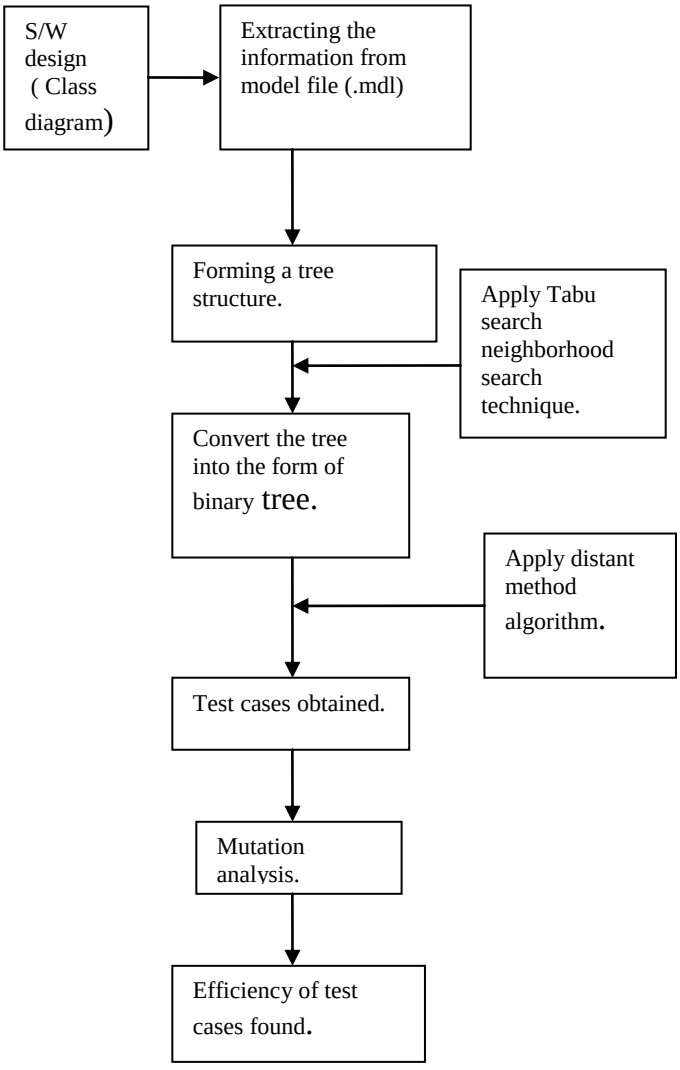

Fig 1. Flow Diagram

VI. MUTATION ANALYSIS

To analysis efficiency of test cases generation mutation process is invoke. Mutation is process of injecting faults to software to analysis the software efficiency. Mutation testing is a process by which faults are injected into the system to verify the efficiency of the test cases. Mutation

testing had reduced its practical use as a method of software testing, but the increased use of objects oriented programming Languages and unit testing frameworks has led to the creation of mutation testing tools for many programming languages as a means to test individual portions of an application.

TABLE I
MUTATION ANLAYSIS

Operator	Faults Injected	Faults Found
Function	10	10
Data Name	10	10
Data Member	10	10
Sub Class	9	9
Total	39	39

Analysis based on above operators our proposed identify 100% bugs based on obtained test cases and Path can be analysed based on obtained test cases.

VII. CONCLUSION

In recent trend Model-Based test case attracts many researchers by using some data mining concept to produce an automated optimal test case. By which human and cost effort are minimized. Our approach using class diagram in UML Model, Comparatively with Evolutionary Tabu search Algorithm yields optimal valid test cases. This paper suggests a model based approach in dealing with object behavioural aspect of the system and deriving test cases based on the Tree structure coupled with Genetic algorithm. From the experimental results, we conclude that our methodology is useful to generate test cases after the completion of the design phase and errors could be detected at an early stage in the software development life cycle.

VIII. OPEN PROBLEM

Our proposed algorithm **(TABU SEARCH ALGORITHM)** could be applied for other UML Diagrams like Use case, Sequence, Collaboration, Activity, State Chart diagrams for generating test cases as a further research in this direction.

REFERENCES

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

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

[3] Chang-ai Sun, 2008 IEEE, "Transformation-based Approach to Generating Scenario-oriented Test Cases from UML Activity Diagrams for Concurrent Applications", Annual IEEE International Computer Software and Applications Conference.

[4] Bin Lei, Linzhang Wang, "Xuandong Li, UML Activity Diagram Based Testing of Java Concurrent Programs for Data Race and Inconsistency ", 2008 International Conference on Software Testing, Verification, and Validation.

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

[6] Emanuela G. Cartaxo, Francisco G. O. Neto and Patr´ıcia D. L. Machado, "Test Case Generation by means of UML Sequence Diagrams and Labeled Transition Systems", IEEE 2007.

[7] Hyungchoul Kim, Sungwon Kang, Jongmoon Baik, Inyoung Ko, "Test Cases Generation from UML Activity Diagrams ", Eighth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing.

[8] Supaporn Kansomkeat and Sanchai Rivepiboon, "Automated-Generating Test Case Using UML Statechart Diagrams ",SAICSIT 2003.

[9] Santosh Kumar Swain, Durga Prasad Mohapatra, and Rajib Mall, "Test Case Generation Based on Use case and Sequence Diagram", Int.J. of Software Engineering, IJSE Vol.3 No.2 July 2010

[10] P. McMinn and M. Holcombe. Evolutionary testing of statebased programs. In *GECCO*, pages 1013–1020, 2005.

[11] B. Meyer. Design by contract. *IEEE Computer*, 25(10):40– 51, 1992.

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

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

[14] J. Offutt and A. Abdurazik. Generating tests from UML specifications. In *UML*, pages 416–429, 1999.

[15] Offutt, S. Liu, A. Abdurazik, and P. Ammann. Generating test data from state-based specifications. *Softw. Test., Verif. Reliab.*, 13(1):25–53, 2003.

[16] R. P. Pargas, M. J. Harrold, and R. Peck. Test-data generation using genetic algorithms. *Softw. Test., Verif. Reliab.*, 9(4):263–282, 1999.

[17] P. Tonella. Evolutionary testing of classes. In *ISSTA*, pages 119–128, 2004.

[18] N. Tracey, J. Clark, and K. Mander. Automated program flaw finding using simulated annealing. In *ISSTA '98*, pages 73–81. ACM Press, 1998.

[19] N. Tracey, J. A. Clark, K. Mander, and J. A. McDermid. An automated framework for structural test-data generation. In *ASE*, pages 285–288, 1998.

[20] N. J. Tracey. *A search-based automated test-data generation framework for safety-critical software*. PhD thesis, University of York, 2000.

[21] Bertolino.A, "Software Testing: Guide to the software engineering body ofknowledge", IEEE Software, Vol. 16, 1999, pp. 35-44.[22] Zhi Quan, Bernhard and Gioranni, "Automated Software Testing and Analysis: Techniques, Practices and Tools", Proc. of Intl Conf. on System Sciences, HICSS'07, 2007, pp 260.

[22] Monalisa Sarma and Rajib Mall, "Automatic Test Case Generation from UML Models," 10th International Conference on Information Technology, 2007, pp. 196-201.

[23] Philip Samuel, R. Mall, and A.K. Bothra ,"Automatic Test Case Generation Using UML State Diagrams", IET Software, 2008, pp. 79-93.

[24] A.V.K. Shanthi, Dr.G.MohanKumar, "Automated Test Case From UML Diagram Using Data Mining Approach", CiiT International Journal of Software Engineering and Technology, Vol3.No3, March 2011.

[25] A.V.K. Shanthi, Dr.G.MohanKumar, "Automated Test Cases Generation For Object Oriented Software", Indian Journal of Computer Science and Engineering, Vol:2, issue 4,Sep2011.