International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 3 Issue 4, April - 2014

Correctness of Implementation in Class Level Testing by Attribute Equivalence

M. Malarvizhi
Department of Computer Science
Sri Manakula Vinayagar Engineering College
Puducherry, India

R. Sathya
Department of Computer Science
Sri Manakula Vinayagar Engineering College
Puducherry, India

***Abstract*—In Software testing, the testing of object oriented software is indispensable in recent years. In that, class level testing is the mostly focused part among object oriented abstractions. Since the oracle problem the main objective of class level testing is to test the correctness of implementation of operations. In the existing system, the test cases are tested by observation using canonical specification. It covers each other. Since, testing of these test cases increases the test inputs. To an exhaustive level, these issue can be eliminated by considering random testing in the place of above testing. In our proposed system, instead of random testing we employ Adaptive Random Testing (ART) algorithms and techniques that have been used for more effectiveness. We intend a progressive conviction of Adaptive Random Testing Algorithm in class level testing of object oriented software. The new finding technique to ensure the testing of attribute equivalence of operation can maximized the test coverage.**

Keywords—Software testing; Object Oriented Software; Adaptive Random testing; Test case generation.

IJERT

I. INTRODUCTION

Testing is a natural process that should be performed throughout the whole development process. Software testing is a significant technique for estimating the clarity of a software product. Into the Lifecycle of software development software testing is also a time consuming and high cost activity. The goal of testing is to detect software failures so that defects may be determined and corrected. The aim of software testing often entail examination of the code as well as execution of that code in various environments and weather as well as analyzing the aspects of code: does it do what it is supposed to do and do what it needs to perform. In the current culture of software development, a testing organization may be displace from the development team. The basic difficulty in testing is finding a test set that will uncover the faults in the program. Exhaustively testing all realistic input/output unit is excessively expensive. The number of test cases increases exponentially with the number of input/output variables. Subdivide the input domain into comparable classes. The traditional approach and object oriented approach is the two ways to improve the projects of Software engineering development. The traditional approach used in the development of procedural programming. Another one used for object oriented projects such that object oriented programming like c++ and Java. In dealing with complexity the object oriented approach to software development has a decided advantage over the traditional approach. Object-oriented programming consists of several different levels of abstraction; namely the algorithmic, class, cluster, and system level. The testing of object-oriented and conventional programming is similar at the algorithmic level and system level. Testing at the class level and the cluster levels presents new challenges. The class level is composed of the interactions of methods and data that are encapsulated within a given class. The object oriented paradigm is founded on several important concepts such as inheritance, encapsulation, dynamic binding, polymorphism etc. These concepts lead to complex relationships among various program elements. The specific aspect and properties of an object-oriented approach extend resulting software systems more authenticate, maintainable, and reusable. However, an object-oriented testing also exhibits new challenges to software testing, as a software system is now dignified of classes of objects and has specific features not found in other programming paradigms.

New testing problems arise from the following facts: (1) Programs in an object-oriented system are not necessarily executed in a predefined order; the sequence of invocation of methods in a class is not specified explicitly; and there are more variations in combining methods in the same class or across different classes. (2) Furthermore, it is mandatory to derive an algorithm for determining the observational equivalence of the output objects so as to judge the correctness of implementations based on class level testing [6] [3]. Formal specifications are mathematically based techniques whose purpose are to help with the implementation of systems and software. That are used to describe a system, to measure its behavior, and to aid in its design by verifying key properties of interest through rigorous and effective reasoning tools As major formal method for defining the functional requirements of object oriented software, the random testing is very important with many benefits, including improvements in the automation and effectiveness of test case generation. Black-box testing methods, such as random testing and boundary value analysis, can produce test data with high speed and low cost. Random testing is a naïve method for generating test data, and has been widely adopted by most popular testing tools. A test case d is an element of input domain $d \in D$. A test case gives a valuation for all the input variables of the program, test cases are chosen randomly until a stopping condition is

met. This might be the detection of a failure, the completion of a predefined number of tests, or the expiration of a set time limit. Respective random testing techniques and algorithms select test inputs using a uniform distribution, while others employ a non-uniform dispersion

II. RELATED WORKS

The random testing, which is one of the most used automated testing techniques in practice. The general purpose of random testing is to generate as many test cases as possible in such a way that they help uncover as many faults or hit as many coverage targets as possible. The idea behind Random Testing is to send random input to the system under test (SUT) [15]. The input is generated from some distribution over the input domain. The output is verified with an oracle that determines if the system under test is acting as specified and expected. The impossibility of exhaustiveness for any non-trivial program, requiring testers to come up with strategies for selecting inputs to be tested in the time available. One possible strategy is random testing. It has several advantages: comprehensive practical lack of bias, applicability, relieve of enforcement in an automatic testing tool, no suspended for selecting inputs out of the set of all inputs. Several other strategies for input generation have been proposed (symbolic execution, object fields, genetic algorithm etc.), but none of these strategies reaches the level of applicability and the speed of execution of random testing [2].

A. Object Oriented Testing

The testing process for object-oriented software is decisive because these languages have been generally used in developing progressive software systems. Numerous efficient test input selection methods for object-oriented software have been suggested state-of-the-art algorithms yield very poor code coverage (e.g., less than 50%) on prominence software. Therefore, one significant and yet dispute problem is to generate expected input entities for recipients and arguments that can reach better code coverage (such as branch analysis) or help unveil faults. The Capture-based Automated Test Input Generation for Objected-Oriented Unit Testing (CAPTIG) [12]. It guided input and method selection that increase code coverage. We anticipate our approach can achieve higher code coverage with a reduced duration of time with smaller amount of test input. Three input selection approaches that collectively help achieve higher code coverage with a small set of test cases. a) Simplified distance-based Selection collects the farthest test input from the used values with lower computation cost than ARTOO [9]. b) On-demand Input Creation immediately generates needed inputs. c) Type-based Selection. It capture class usage patterns that reveal method invocation orders only applicable for capture-based input generation technique. It need to be applied to other input generation technique. The testing of object-oriented systems with emphasis on developing a preliminary taxonomy of faults it identifies a set of candidate testing methods [11]. The testing process for object-oriented programs is compared and contrasted with the traditional advent of unit testing and integration testing. The change of insistence for testing from the practice themselves, to the testing of the interaction between practices via the data-members of a class is accomplished by the employment of a state-based technique which individually tests the description and practices of the data-members [13]. The class testing, that is, the complication of extracting test cases for desirably implementing interactions among gather of classes. This technique uses data-flow analysis for deriving a suitable set of test case specifications for interclass testing and automatically generate feasible test cases that satisfy the derived specifications using symbolic execution and automated deduction. To improve the implemented prototype to reduce some of its limitations and on identifying additional systems to be used as subjects for experimentation [16].

B. Test case generation

The test case generation process provides the subtype association to tolerate testing instruction on base classes to be inherited by derived classes. The subtype relation is specified along the formal specification and test model, and enhances on the efficiency of testing using the class inheritance hierarchy [14]. The object-oriented test tools based on the technique cannot be described clearly, and various real time applications not evaluated clearly. For generating random test cases that have been experimentally demonstrated to have greater fault-detection capacity than simple random testing. A very low failure rate may not be effectively detected. It's not suitable for complex data structures as input [5]. ART is based on various empirical observations showing that many program faults result in failures in contiguous areas of the input domain, known as failure patterns. The relationships between the information available to the software tester is not cleared. The effectiveness of testing strategies is simple [4]. The impossibility of exhaustiveness for any non-trivial program, requiring testers to come up with strategies for selecting inputs to be tested in the time available. ARTOO reduces the number of tests generated until the first fault is found [8]. It also uncovers faults that the random strategy does not find in the time allotted. But the less performance changes to the object distance calculation would affect ARTOO's fault finding ability. To select the test data with high fault-revealing capability is a critical problem in the field of software testing. This is addressed by the Two Point Partitioning algorithm [7]. This method usually reveals the potential faults with the large amount of test inputs, so its cost-benefit is not very sound. The Centroidal Voronoi Tessellations proposed for better test case coverage of the input field [1].

The RBCVT method cannot be considered as an independent approach since it requires an initial set of input test cases and it leads cost effective system. A canonical description of a class with actual implications, an absolute execution, meets all the empirically comparable test cases if and only if it fulfills all the empirically non-comparable test cases [10]. An impossible task in software testing because of the need to verify an infinite number of behavioral outcomes even for one single test case. In this new work we proposed the new Adaptive random Testing algorithm for test case generation.

IJERT

III. PROPOSED SYSTEM

In this work deals with existing system which describes the object oriented testing. It poses the challenging issue of test case generation for large scale input domain and checking the correctness of implementation in the class level testing. It described as an oracle problem. This includes 2 aspects that are testing the equivalent and non-equivalent operations. The equivalence of terms can be tested for checking of interactions among various operations. In the class level testing of object oriented software describes in the previous work the testing of all equivalent test cases if and only if testing of all non-equivalent test cases. The canonical specification of a class with proper imports a complete implementation, the testing of observationally equivalent test cases and the testing of observationally non-equivalent test cases cover each other. It has been pointed out, therefore, that the testing of observationally non-equivalent test cases is necessary and cannot be ignored even after exhaustive testing of observationally equivalent test cases. The canonical specification contains specified preconditions, it cannot cover all the possible test inputs. Therefore it leads to the minimized test coverage. The testing of attributively equivalent test cases and attributively non-equivalent test cases cover each other which can be described in Fig.1. The set of attributes in any class is finite and simple .The class level testing utilizes the adaptive random testing for the effective test case generation. The randomized test case algorithm can be applied.

The class level testing of object oriented software is an oracle problem. It describes the testing of correctness of implementation. The testing includes 2 aspects that are testing the equivalent and non-equivalent ground terms (sequence of operations). Testing of equivalent terms checking of interactions among operations in the terms

In this section, the class can be analyzed from the source code. Here we consider the object oriented or c++ source code. The analyzing information should repository to reduce the redundancy. The control flow graph of the analyzed class can be resolved by the data-flow analyser and symbolic executor. The data flow analyser consist of definitions which using association. Symbolic executor comprised the execution precondition. From these processes the test case generator sequence can be identified. Then test cases can be generated which can be shown in Figure 2.

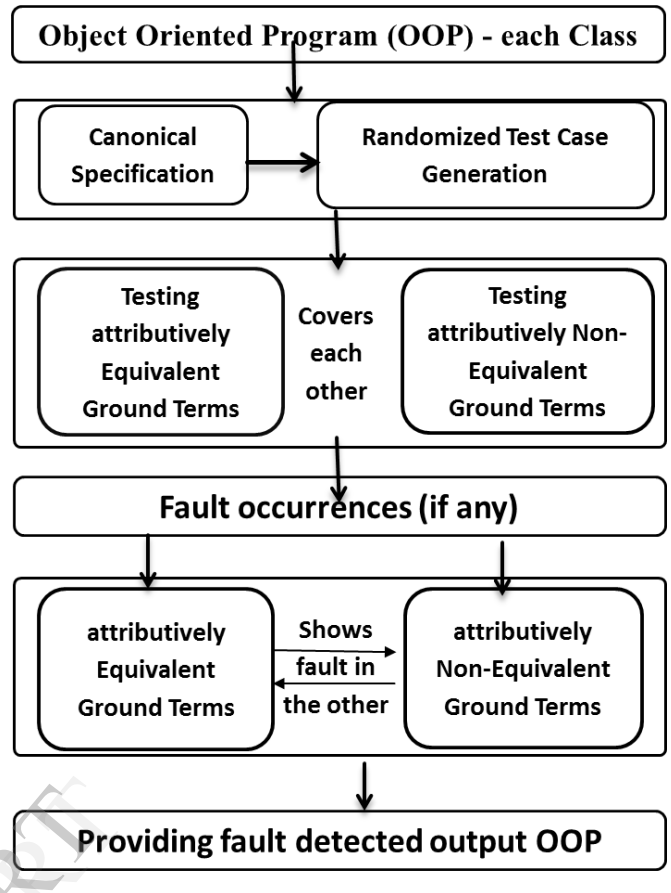

Fig.1: testing of equivalence and non-equivalence cover each other.

These problems can be addressed by the new Adaptive Random Testing algorithm as follows:

Randomized test case algorithm

Consider the whole input as a test regions, the first test case can be randomly choosed.

i. The current sequence test regions can be partitioned into attributes.

ii. The test case which has the higher probability for furthest away from the previously executed test case can be selected as the next test case.

iii. If the test case is a failure-causing input, report fault detection and terminate.

iv. Otherwise, partition the current test region into equal size.

v. Again the same as (iii).

In this section the proposed system is comprised the following functions.

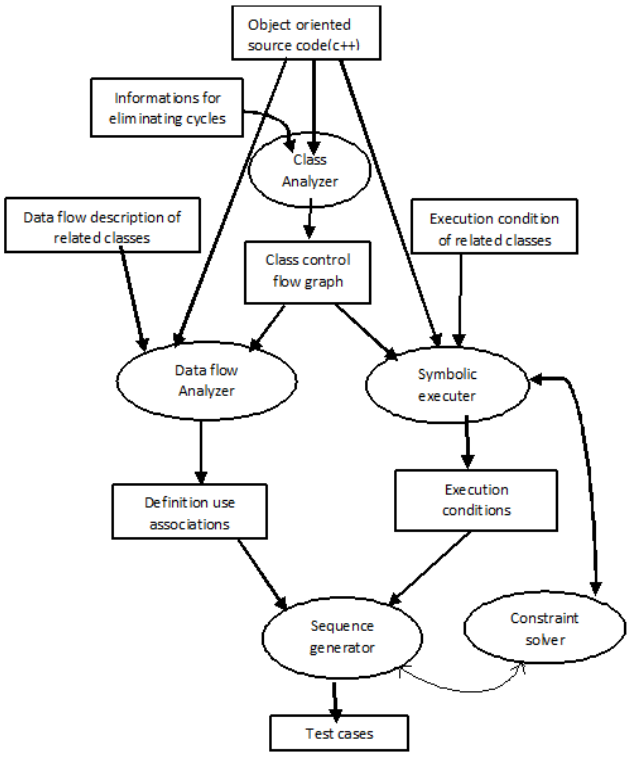

Fig.2: Software architecture for the test case generator

The source code is consider as several classes. The member function and the method sequence of each class is tested. The sub class can be inherited from the each class. From the derived sub class new attributes are identified. Finally the interactions among those attributes to be tested.

The major advantages of proposed system are 1) It reduce the computational complexity while testing the large scale input domain.2) by using the random technique automization can be applied.3) these leads to improve the system efficiency.4) this new adaptive random algorithm should maximize the test coverage.

CONCLUSION

Random testing is an auspicious technology that has been proven to be emphatic, but whose intensive rely upon the conditions of test algorithm condition. In this paper, it have described Randomized Test Case Algorithm for checking the correctness of implementation in Class Level testing of Object Oriented Software by Attributive Equivalence. It have proven that Randomized Test Case Algorithm is capable to accomplish better coverage of complex, real Java application units, while sustaining the most convenient aspect of randomized testing, the proficiency to develop maximized test case coverage directly. It have also shown that were able to optimize the system efficiency and simplify the time consuming. Randomized test case algorithm improves Automization. In this way Adaptive Random Testing increases the system efficiency. An attribute of an object is a visible property of that object. Attributive Equivalence and Attributive non Equivalence of object is tested in class level testing of Object Oriented Software.

As future work, we will also study the application of ARTOO to select test cases from Attributively Equivalence by defining the object distance of nonequivalent terms with a view to spreading the test cases evenly in the Attributive Equivalence. This will alleviate the users from having to assume the regularity hypothesis and make decisions on the maximum numbers of iterations for cyclic paths.

REFERENCES

[1] Ali Shahbazi, Andrew F. Tappenden, James Miller, "Centroidal Voronoi Tessellations— A New Approach to Random Testing," IEEE TRANSACTIONS SOFTWARE ENGINEERING, VOL. 39, NO. 2, FEBRUARY 2013.
[2] Anand S, Burke E, Chen T.Y, Clark J, Cohen M.B, Grieskamp W, Harman M, Harrold M.J, and McMinn P, "An orchestrated survey on automated software test case generation," Journal of Systems and Software, 2013, doi: 10.1016/j.jss.2013.02.061
[3] Chan W.K, Tse T.H," Oracles are Hardly Attain'd, And Hardly Understood: Confessions of Software Testing Researchers," The Symposium on Engineering Test Harness 2013 co-located with The 13th International Conference on Quality Software (QSIC 2013)
[4] Chen T.Y; Kuo, F.C; Merkel, R.G; Tse, T.H "Adaptive Random Testing: The ART of test case diversity," Journal of Systems and Software, 2010, v. 83 n. 1, p. 60-66.
[5] Chen T.Y, Merkel R.G, Eddy G, and Wong P.K, (2004). "Adaptive Random Testing Through Dynamic Partitioning," Proc. of the 4th Int'l Conference on Quality Software (QSIC'04), IEEE CS Press, Braunschweig, Germany, pp. 79–86.
[6] Chen BH.Y, Tse T.H, Chan F.T, and Chen T.Y, "In black and white: an integrated approach to class-level testing of object-oriented programs," ACM Transactions on Software Engineering and Methodology, vol. 7, no. 3, pp. 250–295, 1998.
[7] Chengying Mao," Adaptive Random Testing Based on Two-Point Partitioning," Informatica 36 (2012) 297–303.
[8] Ciupa I, Leitner A, Oriol M, and Meyer B, "ARTOO: adaptive random testing for object-oriented software," Proceedings of the 30th International Conference on Software Engineering (ICSE '08), pp. 71–80, ACM, 2008.
[9] Ciupa I, Leitner A, Oriole, and Meyer, "Object distance and its application to adaptive random testing of object-oriented programs," In RT '06: Proceedings of the 1st International Workshop on Random Testing (2006), ACM Press, New York, NY, USA, pp. 55–63.
[10] Huo Yan Chen, Tse T.H," Equality to Equals and Unequals: A Revisit of the Equivalence and Nonequivalence Criteria in Class-Level Testing of Object-Oriented Software, IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, TSE-2013-03-0084.R1
[11] Jane Huffman Hayes, "Testing of Object-Oriented Programming Systems (OOPS): A Fault-Based Approach" Science Applications International Corporation.
[12] Jaygarl, Hojun, "Capture-based Automated Test Input Generation" (2010).Graduate Theses and Dissertations.Paper 11894.
[13] Turner C.D, Robson D.J," The Testing of Object-Oriented Programs" Computer Science Division School of Engineering and Computer Science (SECS) University of Durham, Durham, England, Technical Report: TR-13/92.
[14] Tse T.H, Zhinong Xu," Test Case Generation for Class-Level Object-Oriented Testing," Proceedings of the 9th International Software Quality Week (QW '96).
[15] Victor carlsson," Adaptive Random Testing of a Trading System," Master's Thesis in Computer Science (30 ECTS credits) TRITA-CSC-E 2011:067 ISRN-KTH/CSC/E--11/067--SE ISSN-1653-5715
[16] Vincenzo Martena, Alessandro Orso, Mauro Pezze," Interclass Testing of Object Oriented Software".

IJERT