AI Based Framework for Automatic Test Data Generation

C Doungsa-ard, K Dahal, and A Hossain, Member IEEE

School of Informatics, University of Bradford, Bradford BD7 1DP, UK
C. Doungsa-ard; K.P.dahal; M.A.Hossain1{@Bradford.ac.uk}

***Abstract*— Software testing is a labor intensive and takes very much effort in software development process. There are many aspects in software testing to assurance quality of software. Gray-box testing is a combination of white-box testing and black-box testing. In gray-box testing, test data is generated from the specification which describes the behavior and data structure of the system. In this paper, a frame work for automatically generating test data from gray-box method is proposed. The proposed framework is pluggable; therefore it can be used in many test generation approaches. Artificial intelligent is contemplated in this research, especially genetic algorithm. UML state diagram is a specification for generating test data because it is similar to control flow graph. Anyway, the proposed framework still needs many further research and investigation.**

***Index Terms*— test data generation, gray-box testing, artificial intelligence, genetic algorithm**

I. Introduction

Software testing is an important activity to assure the quality of software. Unfortunately, software testing is very labor intensive and very expensive. It take about 50 percents of total cost in software developing process[1]. Autonomous testing mechanisms are evolving to reduce tester effort. Due to the major problems of automated testing is identification of data set to use for testing. Without test data, automated test frame work could not be done.

In this paper, a frame work for automatic generated test data mechanism has been proposed. The framework uses UML State diagram to generate test data. Artificial intelligence is applied for generating an appropriate test data set. The paper is organized as follows. In section 2, a review of testing problem and some automated test data generation techniques is discussed. A hort description of UML state diagram which is use as a source of AI based gray-box test data generation is discussed and the proposed frame work for generating test data is proposed in section 3. Finally, section 4 presents conclusion and future work.

II. Review on Testing and Test Data Generation

Quality of test data is one issue for automated test framework. The approaches for generating test data require metric to measure the quality of generated test data. In White-box testing, test data is design for program coverage. That means all paths of program should be executed at least once. Source program can be transformed to control flow graph[2], then it can be instrumented for the path executing each test run. There are three main types of coverage criteria; statement coverage, branch coverage, and path coverage. Many automate test generation approaches chose branch coverage as their mile stone because it is not too weak as statement coverage, and not hard to compute as path coverage.

Random test generated test data may not give a good test data set. GADGET[3] and TGEN[4] used genetic algorithm to improve quality of their result. GADGET computes fitness values by assigning a function to each condition. GADGET has been used to generate test cases for a part of autopilot system, b737. Test data from GADGET covered more than 93 % while random testing achieved around 55 %.

TGEN transforms control flow graph to a control dependency graph (CDG) and use CDG to calculate fitness of generated test data. Each path of CDG represents the smallest set of predicate to traverse every node in control flow graph. TGEN can generate test cases for statement and condition coverage.

Another example of using AI to generate test data set is demonstrated in [5]. The proposed approach uses Tabu searching to find an appropriate test set. Each iteration of generating test data, the proposed approach establishes the sub goal nodes from the best known test as a target for the search.

Black-box testing is completely opposite as compared to the White-box testing. Black-box testing does not need any information about how the program was written[6]. It generates test from software specification to ensure that software work properly.

In Object-orient context, the structure of software is more complicated then the structural one. Conventional test approaches may not enough for testing. The combination of those two traditional approaches is called Gray-box testing. In Gray-box testing, test data generates based on the high level design which specifies the expected structure and behavior of system. Gray-box testing investigates the coverage criteria of white-box method and finds all possible coverage paths. In addition, the generated test case should be satisfied with the black-box testing criteria. Linzhang[7] proposed the automatic test data generation mechanism using gray-box testing. Activity diagram is selected as specification in that research, since it models system by the flow of system. Moreover, activity diagram also represents clear expected structure by its

states, decisions, swim lane, forks, join, and signal sender and receiver. The algorithm for extracting test scenario from the diagram has been proposed. The test scenario is a sequence of possible path in activities diagram. From these paths or scenarios, the executing sequence of program has been explored. A generated test set should cover all possible test scenarios.

III. AI Based Automate Test Data Generation Framework

Initially, our framework focuses on generating test data from UML state chart diagram using genetic algorithm technique. Unified Modeling Language (UML)[8] is a semi formal specification which depicts system requirement into type of graphics called diagram. The UML design uses many diagram, each of which describes a different aspect of system. State diagram present the most interesting aspect from testing point of view. State diagram characterizes the behavior of objects. Object should have their status called state. One object can have one state at a time. The status of object can be changed if there is a trigger that relevant to the current state of object, the object can be changed its state. Guard condition is a key to define the next state to change according to current state and trigger. While the object is changing its state, the object may update some attribute. This depend on the effect define in the transition.

The design of state diagram is evolved from finite state machine. Therefore, it does not have the branch as in a control flow graph. In our proposed approach, the transition coverage is selected as criteria for test data generation. Because branch coverage can not be exist since there is no branch in state chart diagram, while path coverage can produce the problem of path space explosion[9], which causes of loop transition in state diagram.

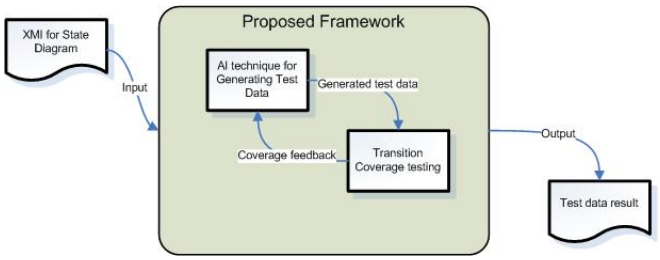

Figure 1 AI based framework for automatic test data

In a proposed framework, we try to adopt the AI-based test data generation for white box testing to generate test data for gray-box testing. The proposed approach is shown in figure 1. The specification used in this gray-box test data generation is State diagram. State diagram is in a form of XMI[10], because XMI is a standard which can be generated from many top modeling programs. From the XMI of state diagram, control flow graph of state diagram is created and store in the "Transition coverage testing" part.

Transition coverage testing part is responsible for creating a graph of state diagram and testing the system using the input test data. Then, all the coverage results from each test cases are recorded and considered whether current test result is suitable for the system under test or not. If generated test data is suitable for some criteria, the frame work will return the current test set. Otherwise, coverage results will send back to create a better test set.

Genetic Algorithm is used for proposed framework at the beginning. The technique used in GADGET is adopted. The fitness of each test data is calculate from the transition which is passed by that test data and the execution of guard statement in the traversed transition. Transition coverage testing part will calculate those fitness values and return it to generating test data part. In order to compare the result of test generation, random test generating technique will also be implemented.

IV. Conclusion and Future work

Currently Object-oriented approach is used for software development. Therefore, traditional testing method approaches may not be suitable for this approach. In this paper, the framework for automatic test data generating mechanism using gray-box approach has been proposed. The proposed design framework is flexible enough to change the generation method easily. The first selected techniques for generating test data are randomize generation and Genetic algorithm.

In our future work, we are trying to tune genetic algorithm for a better result in term of edge coverage and number of generation. Moreover, other AI techniques may be tested to demonstrate the merits of the proposed approach. Generating expected result from specification will also be investigated. Good specification is expected specify expected the correct result of the system. To reduce effort of a software tester, we will investigate to develop an intelligent technique for automatic test data generation and generate expected result from those data.

V. References

[1] Myers, G., The Art of Software Testing. 2 ed. 2004: John Wiley & Son. Inc. 234.
[2] Korel, B., Automated software test data generation. Software Engineering, IEEE Transactions on, 1990. 16(8): pp. 870-879.
[3] Michael, C., G. McGraw, and M.A. Schatz, Generating software test data by evolution. Software Engineering, IEEE Transactions on, 2001. 27(12): pp. 1085-1110.
[4] Roy P. Pargas, Test-data generation using genetic algorithms. Software Testing, Verification and Reliability, 1999. 9(4): pp. 263-282.
[5] Diaz, E., J. Tuya, and R. Blanco. Automated software testing using a metaheuristic technique based on Tabu search. in Automated Software Engineering, 2003. Proceedings. 18th IEEE International Conference on 2003.
[6] Beizer, B., Black-box testing : techniques for functional testing of software and systems. 1995: John Wiley & son Inc. 294.
[7] Wang, L., et al. Generating test cases from UML activity diagram based on Gray-box method. in Software Engineering Conference, 2004. 11th Asia-Pacific 2004.
[8] OMG, OMG Unified Modeling Language Specification version 1.4.2. 2001: OMG.
[9] El-Far, I.K. and J.A. Whittaker, Model-based Software Testing, in Encyclopedia on Software Engineering, J.J. Marciniak, Editor. 2001, Wiley.
[10] OMG, MOF 2.0 / XMI Mapping Specification, v2.1. 2005.