Automated GUI Test Cases Generation with Optimization Algorithm using a Model Driven Approach

Prabhu.J[1]
[1]Research Scholar, CSE Dept, Sathyabama University
Chennai, Tamil Nadu, India
jprabhuit@gmail.com

Dr.N.Malmurugan[2]
[2]Director, Sri Ranganathar Institute of Engineering and Technology,
Coimbatore, Tamil Nadu, India
n.malmurugan@gmail.com

Abstract

The Automated GUI testing improves the Performance of GUI application with the design frameworks across different projects ,products and Automated testing tools which can assist Software developer, The feasibility Study of the GUI application are complex to make simpler test suite are generated based on the budget constraints using regression testing. Software tester and business consultant to solve the above problem. We propose, the writing the oracle is time-consuming factor that, moreover in manual testing in most cases and comparing of expected output and its comparison with the actual output using Model driven approach with the Bee Colony Optimization algorithm for the fault detection in test suite. The quality of the proposed oracle was measured by assessing its accuracy, precision, misclassification error and practicality. In the natural bee colony, there are of two types of worker bees; Forward bees and Reverse bees, who are responsible for the development and maintenance of the colony. The BCO algorithm developed for the fault coverage regression test suite is based on the behavior of these two bees.

Keywords: *Software testing, Fault detection, Test oracle, BCO algorithm, Optimization.*

1. Introduction

The test strategy design should remove most testers from the complexities of the Design-Driven framework**.** Sometimes our software testers are not even professional testers. Sometimes they are application developers need the technical skills for software development. The bulk of our testers can concentrate on test design and test plan only. It is the automation design framework testing will focus on the tools and utility to automate test data. Software development reduces the time needed to obtain the software project.

In Model-Driven Engineering are treated as first class entities that are used to generate code automatically. Since manual testing is time consuming and error prone. Model-Based Testing (MBT) provides techniques for the automatic generation of test cases using models extracted from software artifacts [1].The test oracle is a mechanism to determine whether an application is executed correctly. It is a reliable source of how the System under Test must operate [2]. It is so Expected to provide correct results for any inputs that are specified by the software specifications, and a comparator to verify the actual behavior [3].The test oracles are used as a complete and reliable source of expected outputs and a tool to verify the correctness of actual outputs. Usually, the verifier makes a comparison between actual and expected outputs. The process of finding correct and reliable expected outputs is called oracle problem [4]. Oracle information and oracle procedure are building blocks of each test oracle. The former is a source of expected results, and the last is the comparator. Thus, we concluded that the activities to provide an oracle and verify test cases could be as follows [5]:

1. Generate expected outputs.

2. Execute the test cases.

3. Map the input domain to the expected outputs and fetch the corresponding output for the executed test case.

4. Compare expected and actual outputs.

5. Decide whether there is a fault or not.

The quality of the proposed approach was assessed by measuring the following parameters:

1. Response time of event. 2. Time consuming execution of the test cases. 3. Consistency. 4. Accuracy.

The inclusion of the oracle in the test case is essential for finding errors in the system under test (SUT), which is main goal of testing [6]

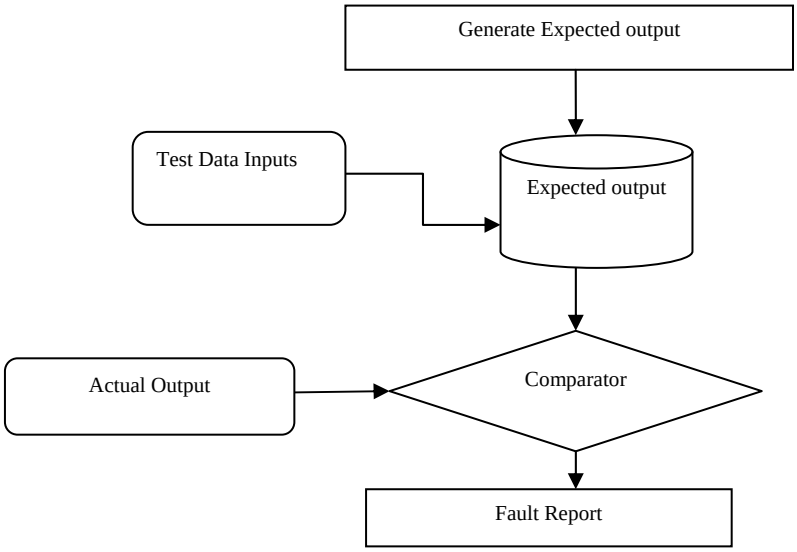

Fig 1. Test Oracle Process. [5]

2. Bee Colony Optimization (BCO)

The basic idea of designing BCO is to compose the multi-agent system (colony of artificial bees) that will search for good solutions of a variety of combinatorial optimization problems. BCO principles are gathered from natural systems. Artificial bees explore through the search space, looking for the feasible solutions. In order to discover better and better solutions, artificial bees collaborate and exchange information. The BCO search is running in iterations until some pre de-fined stopping criteria is satisfied. Population of agents (artificial bees) consisting of B bees collaboratively searches for the optimal solution. Every artificial bee generates one solution to the problem. There are constructive [7][8][9][10] and improvement version [11] of the BCO algorithm. In this paper we apply the improvement version of the BCO algorithm in the fault detection report. The BCO algorithm parameters whose values need to be set prior the algorithm execution are as follows:

TABLE 1.PROPOSED BOC ALGORITHM

Z – The number of bees involved in the searching,
ITE – The number of different iteration,
BP – The number backward passes in a one iteration,
FP– The number of changes in one forward pass,
SO– The best known solution.
The following is the pseudo code of the BCO algorithm:
procedure BCOm (in Z, ITE, BP, SO)for m = 1 to Z do
Decide an initial solution for the m^{th} bee.
Calculate the solution of the m^{th} bee.
for (j = 1 to ITE) do
for (i = 1 to BP) do
for (k=1 to FP)do
Calculate modified solutions generated by possible changes of the m^{th} bee solution.
Choose one of the modified solutions.
Calculate solution of the m^{th} bee.
for i = 1 to BP do
FDI = (Number of Faults Detected/Execution Time) = (FD/ET)
Best Test Cases

2.1. Proposed GUI Application using an automate test oracle

The System under Test (SUT) refers to a system is been tested to recover bugs. it means the ready to deliver of maturity of the software, the successor of integration test.

The results of the comparator can be divided into two categories:

1. Positive: The actual result and expected are same. Therefore the comparator generates reports “No fault occurred”. It represents successful test cases.

2. Negative: the actual result is different from expected results. Therefore the comparator generates reports “fault occurred”. It represents unsuccessful test cases

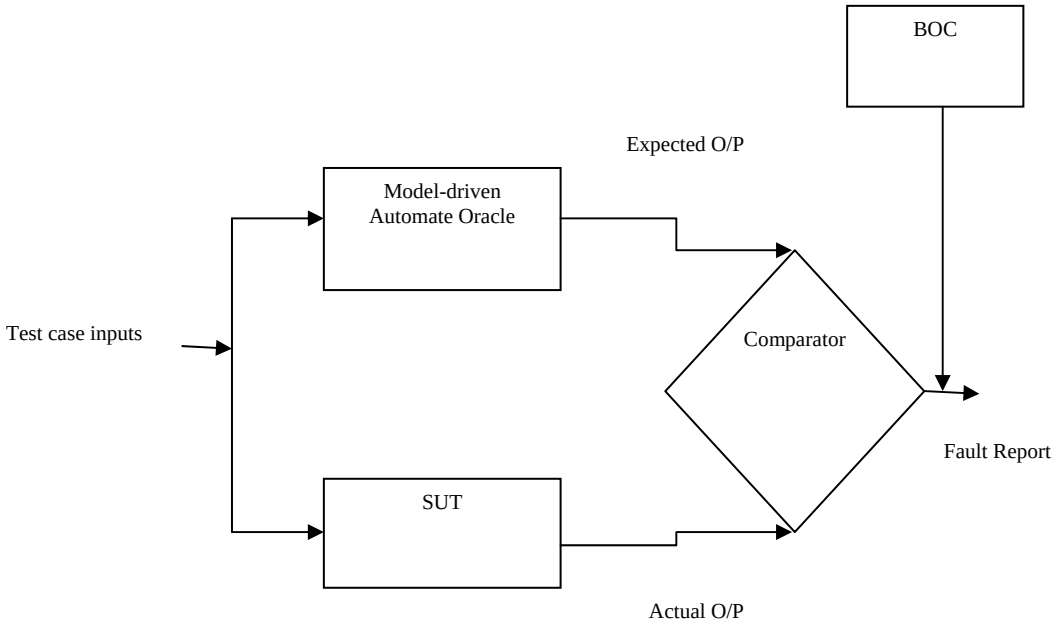

Fig 2. Automate test oracle with BOC algorithm.

The Automated oracle requires a simulated model of SUT. In order to provide a reliable oracle, it generate correct expected outputs for every possible inputs specified in the software documentations. The fault report identified and it is optimized. The forward bee and reverse bee is used correct the failed test cases until unless to be a successful project. The proposed Automate test oracle to compare with actual output and expected output, then BOC algorithm to get optimized result

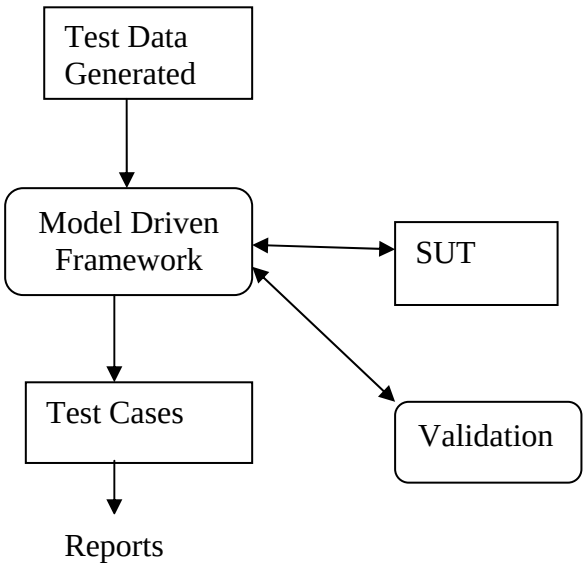

Fig 3. Architecture for Model-Driven Approach

1. Test Data-It contain test data framework. 2.Model-driven Framework-It contain sample model to generate test suite, it provides a set of guidelines for the format of specification. 3. Validation-It is used to validating the SUT driver is correct or wrong.4.SUT- it used to test in a software system for correct operation.

3. Optimization of test cases based on maximum fault coverage

In the proposed BCO technique the test cases are optimized in such a manner to target maximum fault coverage with execution time, that all faults are covered by the test cases selected in the ordered format we use model driven approach. A good testing will detect the change in the program may pass or fail. the proposed algorithm shows the effectiveness is measured using Model-Driven. This is illustrated by examples in section 3.1. are created.

3.1. Example

The problem taken is “Credit Card Management System”. The problem specification is available at website. In this example a test suite has been developed which consisted of 25 test cases. For simplified explanation of the working of the algorithm, a test suite with 10 test cases is considered in it, covering a total of 5 faults. The input test suite contains 10 test cases with default ordering {TC1, TC2, TC3, TC4, TC5, TC6, TC7, TC8, TC9,TC10}, the faults detected (FD) by each test case, the execution time (ET) required by each test case in finding bugs or faults are as shown in Table 2. It is given to the number of faults detected and minimum execution time in selecting the test cases.

TABLE 2. TEST CASES WITH THE FAULTS DETECTED, EXECUTION TIME

Test Cases	FAULTS					Faults detected	Executi on time	Faults identified
	F1	F2	F3	F4	F5	FD	ET	Fdi
TC1		•	•		•	3	.21	14.8
TC2	•	•	•	•		4	.50	8
TC3		•		•	•	3	.64	4.68
TC4			•			1	.7	1.42
TC5	•			•		2	.45	4.44
TC6		•	•			2	.32	6.25
TC7				•	•	2	.41	4.87
TC8	•	•	•			3	.22	13.63
TC9	•			•	•	3	.55	5.45
TC10	•	•	•	•		4	.80	5

4. Results and Discussion

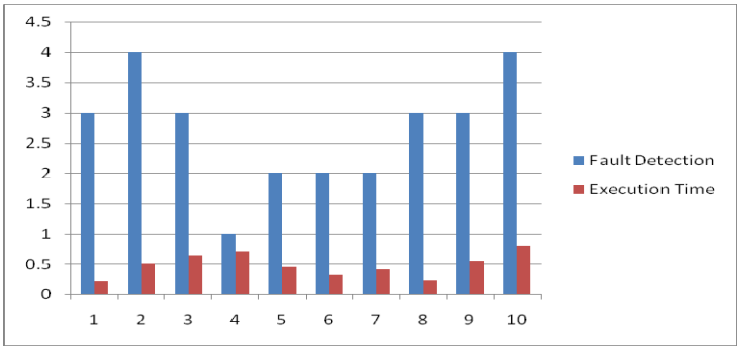

Fig 4.The fault detection occurred before fault identified

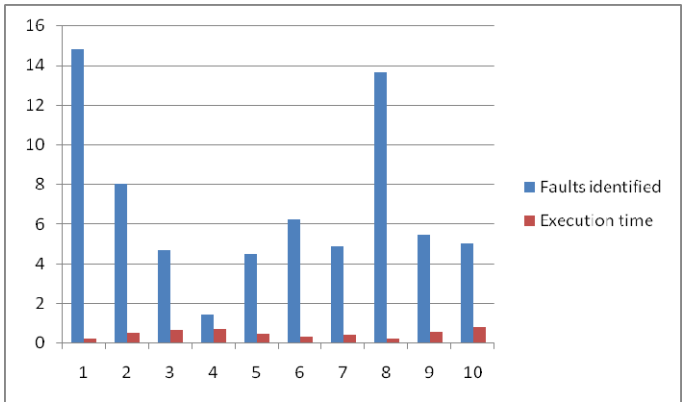

Fig 5.The fault detection occurred maximum after fault identified.

From the results shown in the Table 2 which also represented in Fig. 4 and 5, it shows the fault detected with execution time the fault detection occurred maximum after fault identified in the GUI application compared to fault detection rate occurred before fault identified. It is identified that the fault detection rate is maximum and it is easy for the software to know maximum fault detection cost of the proposed method are improved than the existing optimizatation method.

5. Conclusion

In this paper, we proposed an Model- driven approach allowing the test oracle to automate two important process: the expected output and its comparison with the actual output .We decide an automated testing framework to support the fault detection report is optimized using BCO Algorithm. This algorithm makes effective use of the Fault detection with Execution time and Fault detection rate covered of Forward process and Reverse process for the test cases. Therefore a GUI Automation tool for the complete usage of the algorithm is being developed. It will also be analyzed on larger projects with large number of test cases and faults.

References

[1] S.R. Dalal, A. Jain, N. Karunanithi, J.M. Leaton, C.M. Lott, G.C. Patton, B.M.Horowitz, Model-based testing in practice, in: Proceedings of the International Conference on Software Engineering, 1999, pp. 285–294.
[2] A.M.J. Hass, Guide to Advance Software Testing, Artech House, Boston, 2008.
[3] J.A. Whittaker, What is software testing? And why is it so hard?, IEEE Software (2000) 70–79
[4] P. Ammann, J. Offutt, Introduction to Software Testing, first ed., Cambridge University Press, New York, 2008.
[5] S.R. Shahamiri, W.M.N.W. Kadir, S.Z. Mohd-Hashim, A comparative study on automated software test oracle methods, in: Proceedings of the 2009 Fourth International Conference on Software Engineering Advances, IEEE Computer Society, Porto, Portugal, 2009, pp. 140–145.
[6] B. Beizier, Black-Box Testing: Techniques for Functional Testing of Software and Systems, John Wiley & Sons, 1995.
[7] Luc ic , P., & Teodorovic , D. (2001). Bee system: Modeling combinatorial optimization transportation engineering problems by swarm intelligence. In Preprints of the TRISTAN IV triennial symposium on transportation analysis (pp. 441–445). Sao Miguel, Azores Islands, Portugal.
[8] Lucic, P., & Teodorovic, D. (2002). Transportation modeling: An artificial life approach. In Proceedings of the 14th IEEE international conference on tools with artificial intelligence (pp. 216–223). Washington, DC.
[9] Luc ic, P., & Teodorovic, D. (2003a). Computing with bees: Attacking complex transportation engineering problems. International Journal on Artificial Intelligence Tools, 12, 375–394.
[10] Luc ic, P., & Teodorovic, D. (2003b). Vehicle routing problem with uncertain demand at nodes: the bee system and fuzzy logic approach. In J. L. Verdegay (Ed.), Fuzzy sets in optimization (pp. 67–82). Heidelberg, Berlin: Springer-Verlag.
[11] Davidovic, T., Ramljak, D., Selmic, M., & Teodorovic´ , D. (2011). Bee colony optimization for the p-center problem. Computers & Operations Research, 38, 1367–1376.

AUTHORS PROFILE

Mr. Prabhu Jayagopal is a Research Scholar in the Faculty of Computer science and Engineering at Sathyabama University, Chennai, Tamil Nadu, India. His Research mostly focuses on software testing, Quality Assurance, Software metrics .He received his Master of Computer Science and Engineering from Sathyabama University in 2007 located at Chennai, Tamil Nadu, India and he received the B.tech degree from the Department of Information Technology, Vellore Engineering College, Affiliated to Madras University in 2004 located at Vellore ,Tamil Nadu, India. He worked as a Lecturer in various Engineering colleges for more than 6 years. Now he is working as Assistant Professor Senior in VIT University, Vellore, Tamil Nadu, India From June 2009 onwards.

Dr. Malmurugan Nagarajan has around Twenty two years of both teaching and Industrial experience. He worked extensively in Audio, Video Codec development for mobile platforms and well versed with Audio and Video standards. He is strong in developing tools & system applications and Modeling & Simulation. Currently he is developing new algorithms and IPs for the components in Wireless Broadband Physical Layer He is an Educationist and held various positions ranging from Lecturer to Principal in various educational organizations. His area of interest includes Wavelet variants based Signal and Image Processing, Multimedia Compression & Watermarking, Signal Processing algorithm, Software Testing, development in Telecom domain. He is Member of IEEE, Fellow of IETE, Fellow of UWA and Editor of Journal of Simulation and Modeling.