Data Flow Testing of UML State Machine Using Ant Colony Algorithm (ACO)

Abdul Rauf
College of Computer and Information Sciences Al-Imam Mohammed ibn Saud Islamic University (IMSIU) Riyadh. Saudi Arabia

Abstract
Automatic data flow testing refers to analysis of flow of data within models by using data flow analysis rules. To ensure correct data flow within states we have to consider these data values. The data flow analysis (DFA) forms a source of testing data flow (DFT) by considering defines and uses of the variables. State-based testing examines state changes and its behavior without focusing on the internal details, thus data faults remain uncovered. Empirical studies have shown that existing state-based approaches are not efficient in detecting state based faults. In this paper, an approach is presented to enhance fault detection capability state based approaches. All definition-use paths are used as coverage criterion. We implemented this approach in a tool named data flow generator (DFG). Tool enhances fault detection capability of state based approaches by efficiently detects data flow errors and generates optimal number of feasible test cases are automatically to provide complete def-use paths coverage.

Keywords:
State-Based approaches, Data Flow Testing, Coverage Criteria, Data flow testing

1. Introduction

Unified Modelling Language is de-facto standard defined by OMG for design and specification of object oriented system [21]. UML state diagrams describe the behavior of an object having finite number of states and transitions [4, 5].System change its state, if it receives event trigger affecting the values of invariants used within state [5].To represent the correct behavior of UML state machine, we have to consider these values as data values and analyze the flow of data [5]. State based approaches focused on the control flow information of models without focusing on data flow leading to data flow faults remained uncovered. Hence when state changes occur, alters the values of variable within state that can affect the behaviour of the system.

Testing of data-flow is significant because it supplements control flow information leading to more efficient as well as targeted test cases. Data flow analysis analyses the relationship and association among data objects. Take account of only the information regarding control flow of models in model based testing is not adequate to make sure that flow of data within model is properly flowing throughout the model [5].

Most of the existing state-based approaches focused on the control flow structure and don't examine state changes and its behaviour. No existing state-based testing approaches performing data flow testing provide efficient detection of data faults. However, currently the approaches perform data flow testing using metaheuristic approach don't provide all def-use paths coverage; generate a number of test cases result in redundant and infeasible test cases. Our proposed approach t improves existing state based coverage criteria by enhancing fault detection capability of existing approaches. And provide a mechanism to select the Optimum set of test sequences among alternative while ensuring all definition-use paths complete coverage.

The rest of the paper is ordered as follows; limitations of the existing state based approaches are described in Section 2. Tool is demonstrated in section 3. Evaluation and validation of approach is presented in section 4. Section 5 describes the conclusion.

2. Problem Statement

In traditional data flow testing process of state based approaches, data flow analysis rules are used to analyse invariants to identify definition-use relationship between variables. These rules are used for categorization of variable as define or used in state. These rules identify definition-use pairs in models. But don't generate test cases to cover these definition-use paths and are inefficient in detection of data flow faults. While in other approach relies on mechanism that is partially automated to build feasible sequences of transitions [4]. And definition-use pairs remained undetected due to infeasible paths results in incomplete coverage of all definition-use paths.

Systematic technique should be developed to enhance fault detection capability for insurance of correct flow of data through model. Without considering data flow, changes of data values can affect the overall behaviour of system [5]. In state bases testing, redundant test cases are generated to provide coverage a major problem of state

Manuscript received October 5, 2017
Manuscript revised October 20, 2017

based testing. For generation of infeasible test cases, code/Model techniques are used but still rely on user input to identify them [4]. Only a few techniques focused on both information of control and data flow of system to test the models [5]. Moreover, values of variables alter; affecting the control flow as well as behaviour of system. The complexity of state machine models having a number of states with invariants information aggravates the situation.

So far, a cost-effective strategy is needed that not only identifies all definition-use paths but also generates feasible test cases along with optimal test cases that provide all definition-use paths coverage and makes the data flow testing of state based system easy, uncomplicated and unproblematic [4]. According to [4] definition-use paths that are uncovered are indication of infeasible path and need to address and should focus on definition-use paths that are uncovered. Because due to that data flow faults remain undetected. None of the existing state based technique is efficient enough to detect data flow faults with optimal number of test cases. By seeing all these observations, we propose our approach based on these state based data flow testing approaches to perform data flow testing.

3. Tool

Our proposed approach of data flow testing of UML State Machine is implemented in a tool that is thoroughly explained in this section.

3.1 Min Idea

Figure 1 shows the process flow of our proposed approach and tool is implemented to validate our approach. XML of different state machine models are given as input to tool perform data flow testing. Tool load the XML file and use XML parser to extract relevant information for data flow testing from XML. XML parser organizes the source, target state and data flow information to be used by later components. Tool extract the source and target states as well as transition to create adjacency matrix for showing only feasible states of state machine diagram. Control flow graph is created to show control flow information. Search engine is major component of our tool. Major role of the search engine is to find data flow errors and all definition-use pairs and all definition-use paths of each definition-use pair.

3.2. Test Case Generator

Automated test case generator component of our tool produce feasible and minimal number of test cases that not only provides maximum def-use pair coverage, corresponding to the input source model while ensure maximum fault detection. It does the job by using the optimal solution produced by the search engine component. Automated test case generator takes input of optimal solution generated by search engine. For every input model, it searches the optimal test cases.

3.3 Identification of Definition-Use Pairs

Tool using search engine identify all definition-use pairs within any input state model. Tool search the all the variables exists in model and by analysing each variable, search the definition and use of variable. By searching each usage of defining variable, all definition-use pairs are identified. Identification of all paths is done by search all paths of each variable definition to its use. Def-clear path is path from state where variable is defined to place where that variable is used. Each definition-use pair may have several definition -use paths. One definition-use pairs may have one or more than one def-clear paths. For example, variable cf has deinition-use pair [1, 13] which has six def-clear paths [1-›2-›3-›8-›7-›13], [1-›9-›10-›13], [1-›6-›7-›2-›3-›11-›10-›13], [1-›6-›7-›13], [1-›6-›7-›2-›12-›11-›10-›13], and [1-›2-›12-›8-›7-›13]. After the identification of def-clear paths, test cases are generated to cover these definition-use paths. Each def-clear path is examined and redundant definition-clear paths are removed by tool. For example, as in table 7.1 the def-clear paths identified by tool are [1-›6-›7-›13], [1-›6], [1-›6-›7-›2], [1-›6-›7-›2-›3]. Aforementioned 4 definition-use paths, a single test case can cover these def-clear paths.

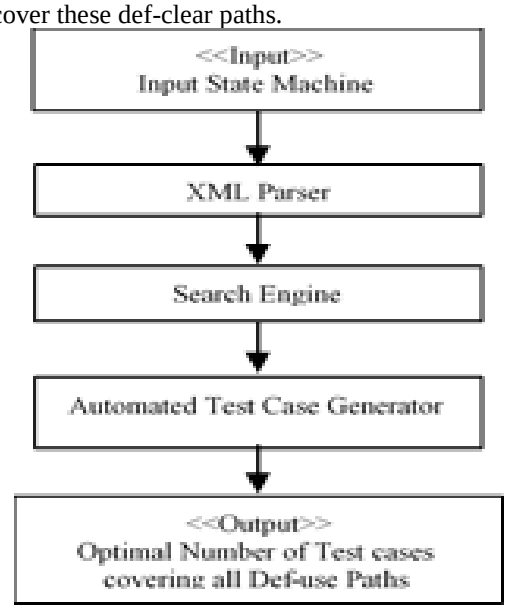

Figure 1. Tool Architecture

That is [1 -› 6 -› 7 -› 2 -› 12 -› 11 -› 10 -› 13] cover the [1-›6-›7-›13], [1-›6], [1-›6-›7-›2], [1-›6-›7-›2-›3]. All these def-clear paths are included in this test case.

3.4 Data flow testing Process

Data flow testing process using the proposed approach consists of following steps.

a) XML of state machine are used to given as an input
b) Control flow graph is created and all definition-use paths are identified
c) Efficient Detection the data flow errors
d) Optimal test cases are generated to fulfil the all definition use paths coverage criteria.

4. Evaluation

Our proposed approach is straightforward and uncomplicated in its application. We applied this approach on a case study of "telephone system". State machine diagram is considered to validate our proposed approach. Figure 2 shows the state machine diagram of system.

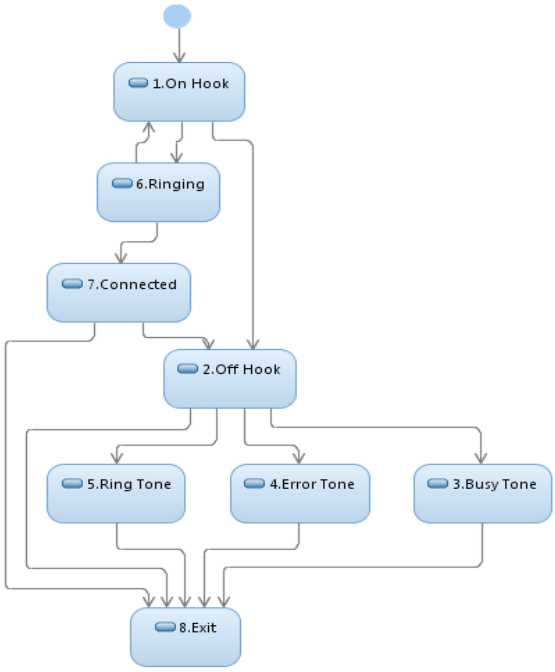

Figure 2 State Machine Diagram of Telephone System

Figure 3 shows the screen shot of input XML file containing the states as well as invariants information about system. Output obtained by applying this proposed approach is shown in figure 4. Results obtained by applying the approach indicates that our approach not only generate non-redundant optimal test cases but is also efficient in detection of data flow faults.

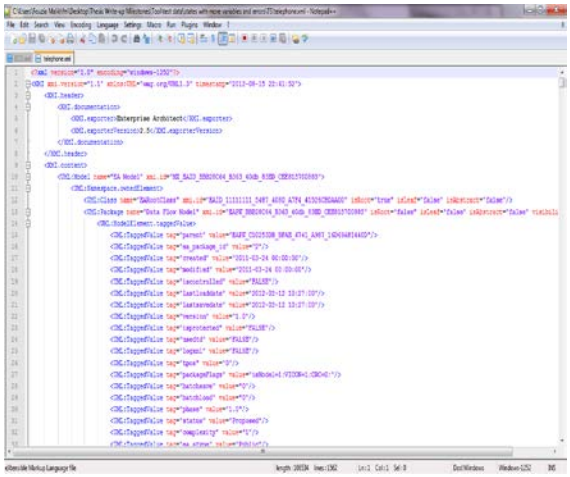

Figure 3. Xml of Input Model

Traditional data flow testing approaches are not efficient in detection of data flow faults. These state based approaches generate redundant test cases to fulfil the coverage criteria and still providing incomplete coverage due to non-detection of infeasible paths. However, our approach use metaheuristic technique ACO that by using heuristic provide efficient detection of data flow faults and automatically detects infeasible paths. As ACO finds good paths through the graph, it not only detects feasible paths but also generate minimal non-redundant test cases in providing all definition-use paths than existing state based approaches. That not only avoids redundant number of test cases while save time consumption. Moreover, our approach detects infeasible paths that are worthless to tester and we don't rely on user input to detect infeasible paths. The XML of state machine don't require any expertise generated using UML modelling tool enterprise architect. Moreover, XML of any state machine model is generated and can be given as an input. Output is generated against each model. In this way our proposed approach detects data flow errors and generates test cases of any of the state machine diagram given as an input.

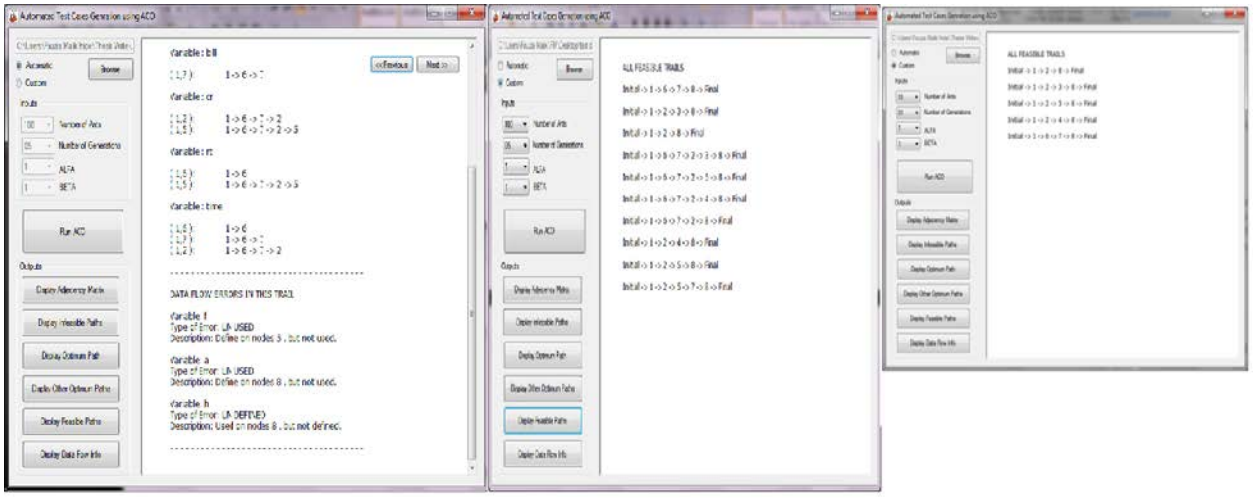

Figure 4. Result: All definition-use paths, feasible test cases & Optimal number of test cases

5. Conclusion

A novel approach is presented in this paper for data flow testing of UML state machine. Main theme is to solve state based approaches problems by using metaheuristic approach. Ant colony algorithm is used to identify all definition-use paths and for the detection of data flow faults to ensure correct flow of data through the model. After that test cases are generated to fulfil the coverage criteria. Validation of this approach is performed on software system. Results obtained indicate the efficient detection of data flow faults. This approach detects infeasible paths to evade the difficulties associated with manual detection.

References

[1] Sanjay Singla,"An Automatic Test Data Generation for Data Flow Coverage Using Soft Computing Approach", International Journal of Research and Reviews in Computer Science (IJRRCS), 2011

[2] Mingjie Deng, “Automatic Test Data Generation Model by Combining Dataflow Analysis with Genetic Algorithm”, IEEE, 2009

[3] Jun Hou, " DFTT4CWS: A Testing Tool for Composite Web Services Based on Data-Flow", Sixth Web Information Systems and Applications Conference, 2009

[4] Lionel Briand, "Improving the coverage criteria of UML state machines using data flow analysis",Softw. Test. Verif. Reliab.; 20:177–207, 2009

[5] Tabinda Waheed,"Data Flow Analysis of UML Action Semantics for Executable Models", Springer-Verlag Berlin Heidelberg, 2008

[6] Praveen Ranjan Srivastava, “Optimized Test Sequence Generation from Usage Models using Ant Colony Optimization”, International journal of software engineering, 2010

[7] Bor-Yuan Tsai, “An Automatic Test Case Generator Derived from State-Based Testing”, Department of Information Management, Tamsui Oxford University College,2000

[8] Harsh Kumar Dubey, “Automated Data Flow Testing”, IEEE 978-1-4673-0455-9/12, 2012

[9] XUE Xue-dong, “The Basic Principle and application of Ant Colony Optimization Algorithm”, IEEE 978-1-4244-6936-9/10, 2010

[10] Raluca Lefticaru, “Automatic State-Based Test Generation Using Genetic Algorithms”, IEEE 0-7695-3078-8/08, 2008

[11] Chartchai Doungsa-ard, “Test Data Generation from UML State Machine Diagrams using Gas”, IEEE International Conference on Software Engineering Advances(ICSEA 2007), 0-7695-2937-2/07, 2007

[12] Praveen Ranjan Srivastava, “Structured Testing Using Ant Colony Optimization”, Copyright ACM 978-1-4503-0408-5/10/12, 2010

[13] Hyeon-Jeong Kim, “Deriving Data Depencies from/for UML State Machine Diagrams”, IEEE 978-0-7695-4453-3, 2011

[14] Praveen Ranjan Srivastava, “Automated Software Testing using Metaheuristic Technique Based on Ant Colony Optimization”, IEEE DOI 101.1109/ISED.2010.52, 2010

[15] AhmedS. Ghiduk,"Using Genetic Algorithms to Aid Test-Data Generation for Data-Flow Coverage",IEEE, 2007

[16] L.C. Briand, “Improving Statechart Testing Criteria Using Data Flow Information”, Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering (ISSRE’05), 2005

[17] Moheb R. Girgis, “Automatic Test Data Generation for Data Flow Testing Using a Genetic Algorithm”,Journal of Universal Computer Science, 2005

[18] Huaizhong LI, "An Ant Colony Optimization Approach to Test Sequence Generation for State based Software Testing ",Proceedings of the Fifth International Conference on Quality Software (QSIC’05), 2005

[19] M. R. Girgis, "Automatic test data generation for data flow testing using a genetic algorithm", Journal of Universal computer Science, 2005

[20] Hyoung Seok Hong,"Data Flow Testing as Model Checking", proceedings of the 25th international conference on Software Engineering (ICSE'03) ,2003

[21] “OMG Unified Modeling Language Specification" version 1.3.1, 1st edition 2000
[22] Elaine J. Weyuker, “An Empirical Study of the Complexity of Data Flow Testing”, IEEE 0225-3/88/0000/0188 , 1988
[23] Phyllis ,"An Applicable Family of Data Flow Testing Criteria",IEEE Transactions on Software Engineering, 1998
[24] S. Rapps and E. J. Weyuker, "Data flow analysis techniques for test data selection“, Proceedings of the 6th IEEE-CS International Conference on Software Engineering, 1982
[25] S. Rapps and E. J. Weyuker, "Selecting software test data using data flow information", IEEE Transactions on Software Engineering, 1985
[26] J. Shan, “Research on Formal Description of Data Flow Software Faults”, International Conference on Computer Application and System Modeling (ICCASM 2010), 2010