Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 2, Issue. 4, April 2013, pg. 491 – 494

RESEARCH ARTICLE

Coverage Analysis and Chinese Postman Algorithm for Efficient Model-Based Test Generation

Parampreet Kaur[1], Gaurav Gupta[2]
[1]Computer Science Engineering, Punjabi University Patiala, India
[2]Computer Science Engineering, Punjabi University Patiala, India

[1] *paramnagpal16@gmail.com;* [2] *gaurav_shakti@yahoo.com*

Abstract— The software testing activity aims to check whether an implementation behaves according to the specified system requirements. Model-based testing is an important black-box testing approach based on formalisms to specify critical and reactive systems. In a black-box approach the internal structure of an implementation candidate is unknown by the test designer. Thus, an implementation receives stimuli from the environment and produces autonomous outputs. Good software testers cannot avoid models. They construct mental models whenever they test an application. They update their mental model of the application and apply new tests according to the model. MBT calls for explicit definition of the model, either in advance or throughout the testing endeavour. However, software testers of today have a difficult time planning such a modelling effort. They are victims of the ad hoc nature of the development process where Requirements change abruptly. Testers who have only a few hours or days to test will most often opt to maintain their models in their head and perform testing manually. Today, the scene seems to be changing. Modelling in general seems to be gaining favour in fields where quality is essential and low quality software is not an option. Henceforth, we have introduced Extensive coverage analysis methods and a Guided Chinese Postman Algorithm for generating Test Cases.

Key Terms: - Model Coverage Analysis; MBT; Chinese Postman Algorithm; State charts; extended MBT

I. INTRODUCTION

Coverage criterion Analysis is the basic and traditional empirical means to measure the fault detection potential of test suites. They are also used to guide and abort the test generation process. Testing is often incomplete, i.e. cannot cover all possible system behaviours. They can be applied to anything from requirements via models to machine code. There are different kinds of coverage criterions which may describe parts of the system behaviour that must be covered by tests. They may not only address single instruction or value assignments but also long sequences of instructions or data flow paths.

Most of the criteria used for selecting test cases is classified according to the following three criteria:

1) The test case specification is a description of a structural criterion which needs to be covered.

2) The test case specification is a description of functional aspects, also called scenarios, which needs to be covered.

3) The test case specification contains random information about different aspects of the implementation.

Common belief however seems to be that random testing should systematically complement coverage based approaches. Coverage based approaches should be used to detect specific faults while random approaches aim at providing confidence about programs reliability. A test case is defined as a sequence of stimuli from the environment to exercise the system structure. The resulting sequence is named observable and it gives the output behaviour.

II. MODEL COVERAGE ANALYSIS CRITERIA

A model of software is a depiction of its behaviour. Behaviour can be described in terms of the input sequences accepted by the system, the actions, conditions, and output logic, or the flow of data through the application's modules and routines.

A. Types of Coverage Criteria

i) Transition Based coverage Criteria
ii) Control Flow Based Coverage Criteria
iii) Data Flow based Coverage Criteria
iv) Boundary Based Coverage Criteria.

Transition Based coverage Criteria includes All-States, All Transitions, All-Paths coverage.

All-States: A test suite that satisfies the coverage criterion All-States on a state machine must visit all states of the state machine.

All-Transitions: Satisfying the coverage criterion All-Transitions requires to traverse all transition sequences.

All-Paths: This coverage criterion is satisfied if all paths of the state machine are traversed.

Control Flow Based Coverage Criteria

Decision Coverage: To satisfy Decision Coverage, a test suite must cover the positive and negative assessment of all guard conditions of a state machine.

Condition Coverage: Condition Coverage is satisfied if all Boolean conditions of each guard are assessed to true and false at least once.

For state machines, coverage is expressed in terms of inputs, states, and transitions. For example, there is the criterion of covering all states in Fig 1. Finite state machines are equivalent to directed graphs structurally. Since tests generated based on a model are simply paths in that graph, graph theory can help by supplying traversal algorithms and achieving graph-structure coverage criteria for testing.

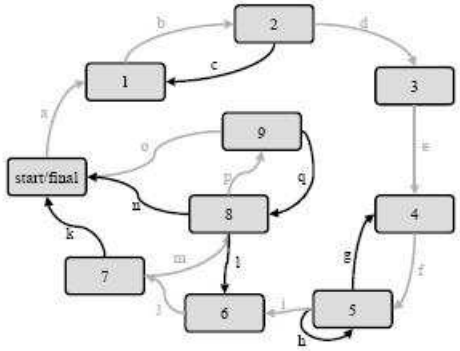

Fig 1 A state machine test path that covers all states.

III. TEST GENERATION

The difficulty of test generation from a model depends on the nature of the model. Models that are useful for testing usually possess properties that make test generation easy and often automatable. For some models, all that is required is to go through combinations of conditions described in the model, requiring simple knowledge of combinatory. In the case of finite state machines, it is as simple as implementing an algorithm that randomly traverses the state transition diagram. The sequences of arc labels along the generated paths are, by definition called tests. For example, in the state transition diagram below, the sequence of inputs {a, b, d, e, f, i, j, k} qualifies as a test of the represented system.

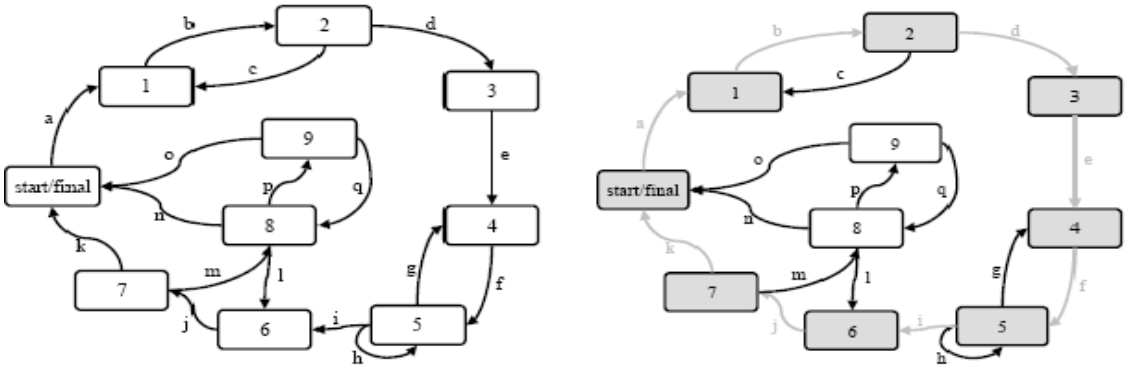

Fig 2 A state machine (left) and a test path in the state machine (right)

IV. DIRECTED CHINESE POSTMAN ALGORITHM

In order to realize the extended-model based testing, we use Chinese postman algorithm. Though there could be several approaches to decide route of operation (Fig 3, such as random walk and using probability [4], we use Chinese postman algorithm in this research because it can generate the shortest path to cover all routes/edges efficiently .Chinese postman problem is well known and was established by a Chinese mathematician, Mei-Ku Kuan. Chinese postman problem is that a postman delivers letters following routes as short as possible. The definition is:

A postman tour in a graph G is a closed walk that uses each edge of G at least once.

When Kuan developed Chinese postman problem, only undirected connection was considered, which is called undirected Chinese postman problem. To adjust this Chinese postman problem into the advanced model-based testing, the undirected Chinese postman problem is extended into directed Chinese postman problem. In general, most edges for software operation have directed edges (each edge is associated with an ordered pair of vertices). For example, in Figure 3, there is a path: {Start Invoke, Choose function. However there is no feasible alternative way for: {Choose function, Invoke, Start}.

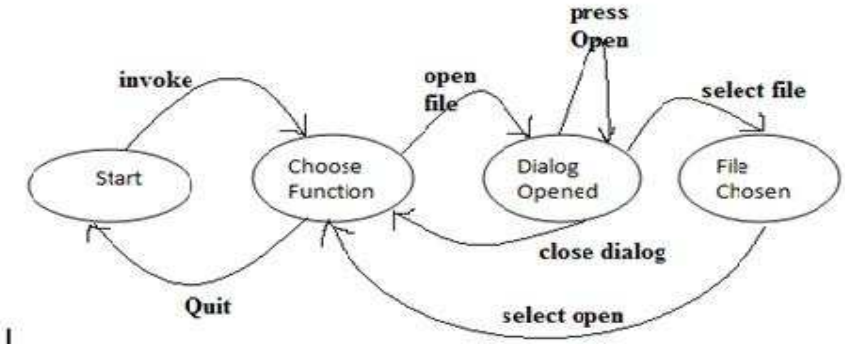

Fig 3 Transition Diagram G of Ms Notepad.

V. CONCLUSIONS

The application of coverage techniques at the model level seems to be a promising approach. These techniques allow easier test selection from executable models while ensuring the coverage of targeted behaviours of the model .However, criteria has to be adapted to specification formalisms. Most of the usual criteria can be easily applied to models, since model's executable aspect makes them look like programs. Moreover, approaches based on model coverage may be adapted to perform functional testing. This can be done through property coverage by model checking approaches. All these properties make model-coverage-based-testing methods good candidates to detect specific faults at the earliest design level. The strength of coverage approaches relies mainly on their ability to explore in a systematic manner missing logic faults or bad treatment of bounds etc. These approaches are the only one to tackle the problem of detecting catastrophic failure causing inputs. However, one must keep in mind that these properties are not sufficient to ensure reliability. In particular,

there is no scientific evidence that coverage based testing is better than random testing to reach this purpose. To gain more insight, a further analysis of what is a "typical program under test" is needed.

REFERENCES

[1] Ibrahim K. El-Far and James A. Whittaker: Model-Based Software Testing.
[2] Christophe Gaston ,Dirk Seifert, Evaluating Coverage Based Testing, North Carolina,2005
[3] Robinson, H. Finite state model-based testing on a shoestring. International Conference on Software Testing Analysis and Review, San Jose, California, USA, 1999.
[4] Takahashi, J., and Yoshiaki, K.,"Extended Model based testing by directed Chinese postman algorithm" 7th IEEE International Symposium on High Assurance Systems Engineering (HASE'02).
[5] T. Quatrani. Visual Modeling with Rational Rose and UML. Addison Wesley Longman, 1998.
[6] Guan, Graphic Programming using Odd and even points, Chinese Maths, 1992.
[7] Blackburn, M., Busser, R. & Nauman, A. Why Model-Based Test Automation is Different and What You Should Know to Get Started. International Conference on Practical Software Quality and Testing, Washington, USA, 2004.
[8] Safford, E. Test Automation Framework, State-based and Signal Flow Examples.
[9] Mikko Aleksi Mäkinen Model Based Approach to Testing.
[10] Mark Utting and Bruno Legeard Practical ModelBased Testing a tool approach.

