Volume 2, No. 5, Sept-Oct 2011

ISSN No. 0976-5697

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

An Approach for Test Case Generation Using UML State chart Diagram

Veenu Makker*
Lecturer
Department of information technology
Haryana College of technology & management
Kaithal, Haryana, INDIA
Veenu.makker63@gmail.com

Vikram Singh
Professor
Department of CSA
Chaudhary Devi LalUniversity
SIRSA, Haryana, INDIA
vikramsinghkuk@yahoo.com

Abstract: Software testing accounts for consumption of the largest chunk of life cycle resources. Within the software testing test case generation is a paramount step. In this paper, we propose an approach to generate the test cases using UML state chart diagrams. Firstly, state chart diagram is transformed into a labeled graph which in turn is transformed into what has been called an intermediate testable model – a representation suitable for deriving system level test cases.

Keywords: Software testing, State chart diagrams, Labeled graph, Intermediate testable model, test cases.

I. INTRODUCTION

Owing to the spiraling size and complexity of software systems, the specification and design of such systems has assumed much importance. Software quality assurance and testing is aimed at ensuring that the software meets user requirements and its specifications. However, the field of software testing has a number of underlying issues like effective generation of test cases, prioritization of test cases etc which need to be tackled. In a typical software development project, around 60 % of development effort is typically spent on testing. Both test case design and test case execution are time consuming and labor intensive. Hence test case generation based on design specification is important.

Test case generation from design specification has the added advantage of allowing test cases to be available early in the software development cycle, thereby making test planning more effective [1],[2]. It is therefore advantageous to generate test cases from the software design or analysis documents rather than the code .The unified modeling language is a visual modeling language that comprises nine types of graphics, called diagrams [3],[5]. This paper presents test cases that are generated from the specification based on UML state chart diagrams. Our objective is to develop a testable model that can manage the test case generation process from a state chart diagram.

This paper aim at providing methodological support for automating the test case generation process from a state chart diagram using a two-phase approach. The control flow analysis of state chart diagram is accomplished in the first phase. In order to accomplish control flow analysis, it is desirable to view a state chart diagram in terms of unit of states such as in control flow analysis of programs. A directed graph representation known as labeled graph is presented as an outcome of the first phase. Generating test paths from a labeled graph constitutes the second phase in our approach. To do this, our approach considers each node of a labeled graph as a single entry, single exit region that can be characterized by a single entry node to the region and a single exit node from the region. Since a labeled graph is identified in terms of regions rather than nodes, exits paths can be associated to the respective regions. The complex structure of a state chart diagram is transformed into a well formed hierarchical structure which has been called as an intermediate testable model.

This paper is organized as follows. Section 2, describe about basic. This is followed by our proposed approach for test case generation. Section 4 describes a case study for our proposed approach. Finally conclusion and future direction are discussed in section 5.

II. BASIC TERMINOLOGY

This section presents the description and definitions of terms and concepts that come across in this paper. A brief review of the UML State chart diagram and labeled graph and finally, the concept of testable model known as Intermediate Testable Model (ITM) have been discussed.

State chart diagram is a pictorial representation of a state machine, emphasizing the flow of control from state to state. A state chart "G" can be described as a couplet <V, E> where

- V is the set of nodes of G
- E is the set of edges in G. Nodes represent states and edges represent transitions between states.

A Labeled graph is a labeled graph is a directed graph, G=<A, E, IN, F>. Here, in denotes the initial node such that A is a set of nodes composed of BN and CN, where BN is a set of block nodes, and CN is a set of control nodes like DN, MN, FN, JN where DN is a set of decision nodes, MN is a set of merge nodes, FN is a set of fork nodes and JN is a set of join nodes denotes a set of control edges [1].

– An initial node denotes the beginning of a labeled graph.

– A block node denotes a sequence of states.

– A decision node denotes a conditional expression such as Boolean expression that need to be satisfied for selection among different execution.

– A merge node denotes an exit from the selection behavior of same states.

– A fork node is an entry into parallel execution part.

– A join node denotes an exit from a parallel execution part.

– A final node denotes an exit of a Labeled graph.

Intermediate Testable model In order to manage the complexity of a labeled graph, an intermediate representation of the labeled graph called the intermediate testable model (ITM) has been proposed. With ITM representation; one can analyze each region independently. In other words, a region in the labeled graph has been mapped to a special node in ITM termed as compressed into a composite node [1]. An ITM can finally be described as a chain of nodes as follows:

Gk=<A, E, in, f> is a chain of nodes where

- "A" is a finite set of A_b and A_c node.

- "in" is the start node representing an initial node of the state chart diagram.

- "f" is the final node representing without any successor node.

III. TEST CASE GENERATION

Our approach consist of four phases, control Flow analysis, building Intermediate model, Test Scenario generation and test case generation. The control flow analysis phase produces the directed graph representation of given state chart diagram. In the second phase the labeled graph analyzed for finding out the dependencies among the nodes. In the test scenarios generation phase the ITM can be mapped to different scenarios and from these scenarios, the test cases can be constructed. In the final step, the input and output values are determined to form test cases.

A. Control Flow Analysis :

In order to map a state chart diagram it is first analyzed and converted it into a labeled graph consisting of nodes like block node Bi, merge node Mi, decision node Di, fork-Join node Fj and Ji. Each state is represented by a node but the sequence of transition remains same.

B. Building an ITM:

In order to build ITM our objective is to analyze the labeled graph and find out the dependencies among the nodes. Thus a labeled graph can be simplified by identifying the minimal regions and reduce them to their corresponding composite nodes. Such a procedure of replacing minimal regions by composite nodes has been termed as composition. If the regions are nested with other regions, then the compositions are done from the innermost region to the outermost successively. The composition procedure eventually reduces a labeled graph to a single chain of nodes known as ITM.

C. Generating Test Scenarios:

An ITM is a chain from initial node to final node and it is called the base path Bp. This base path contains block nodes and composite nodes. A composite node is made up by grouping the internals path. When a composite node Ci is expanded, then the sequence of nodes from initial to composite is attached with a suffix which is the set of all internals paths.

The ITM can be mapped to different scenarios and these scenarios can be used to construct the test cases. Building a set of test cases can therefore be considered as replacing each of the composite nodes by each of its internal path. This process is to be continuing until every node is expanded.

D. Test case Generation:

In test case generation process our objective is to expand the composite node one by one in the base path and obtain many scenarios. For each scenario the test cases are generated and can be thought of the set of some terms like initial condition, input and output.

IV A case study

In this section, there is a case study to illustrate the test case generation approach. A typical state chart diagram of "ATM System" is shown in figure1. State chart diagram shows all the states of one object under processing.

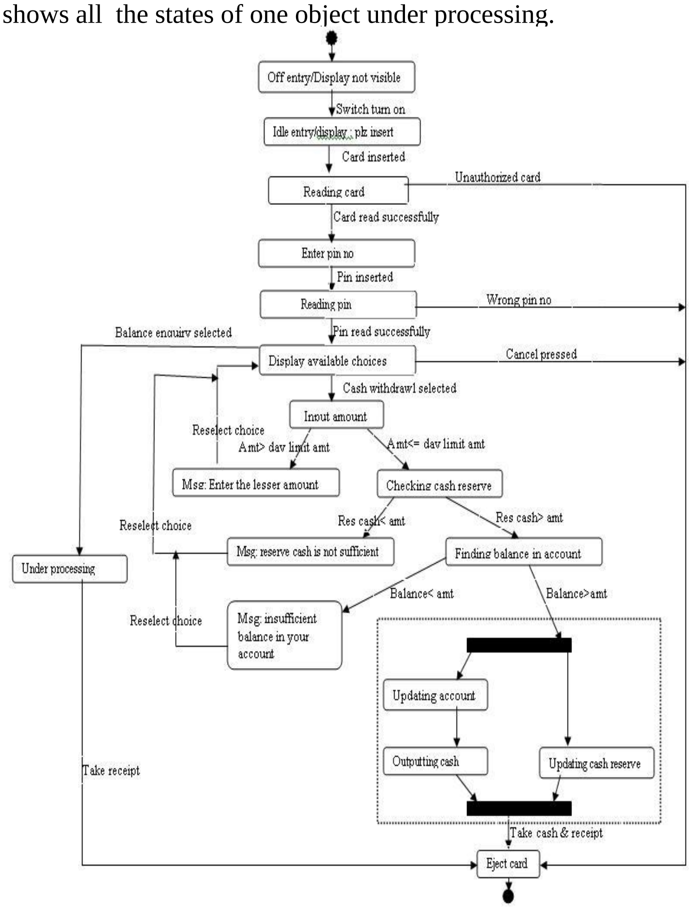

Figure1: State chart Diagram of ATM System

Now convert this state chart diagram into labeled graph as shown in figure 2.In labeled graph, block nodes are represented by oval shapes and Bi is used to denote the same. Where i=1, 2, 3. To represent the fork and join node thick line segments are considered. To denote the decision and merge nodes diamond symbols are used. A filled circle notation is used to denote initial nodes and a small circle notation is used to denote final node. FNi and JNi notations are used for fork and join nodes. Di and Mi notation uses for decision and merge nodes. Labeled graph G is to taken as an input for composition procedure. The minimal regions are identified and replaced with composite nodes. The selection nodes, concurrent node and loop node are labeled as Si, Xi, and Li respectively.

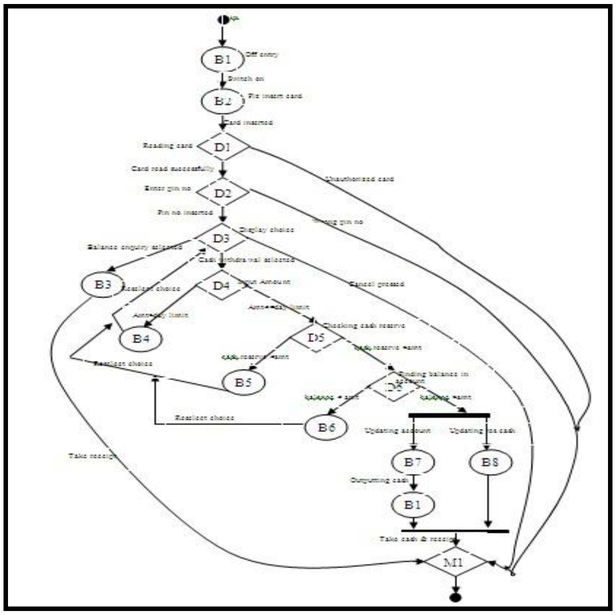

Figure 2: Labeled Graph for State chart diagram of ATM system.

Before starting the composition procedure, the labeled graph is redrawn in form of intermediate model. Thus Labeled graph is obtained as an intermediate diagram shown in figure 3.

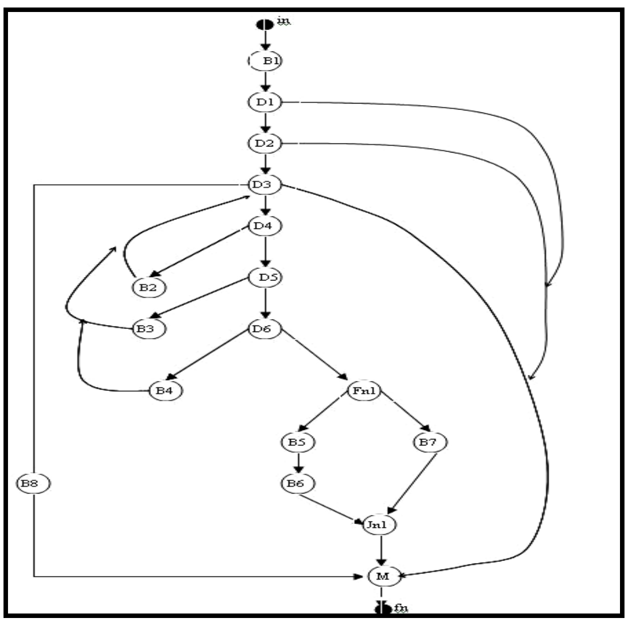

Figure 3(a) ITM1

By having first intermediate diagram, the procedure of composition for obtaining next intermediate model is carried out. The transformed of figure 3(a) is shown in figure 3(b). During the second iteration, procedure is repeated with all minimal regions of previous graph. The resulting graph after second iteration is shown in figure 3(c). After third and fourth iteration, the finally ITM is shown in figure 3(e).

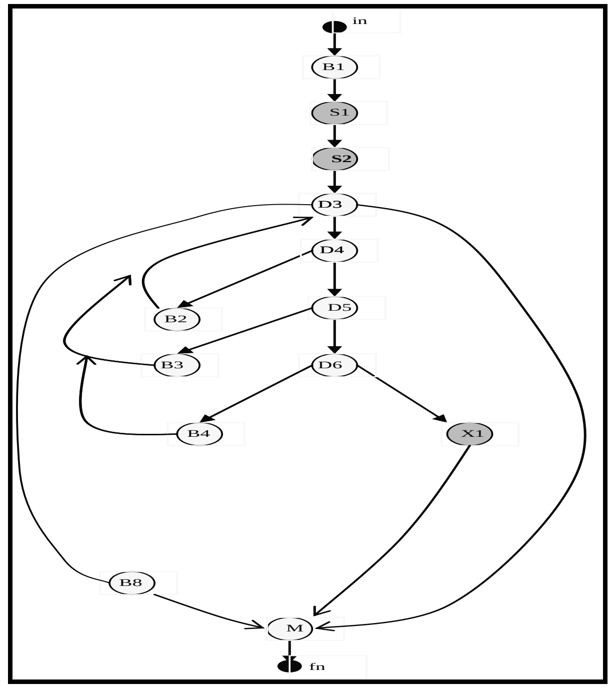

Figure 3(b) ITM2

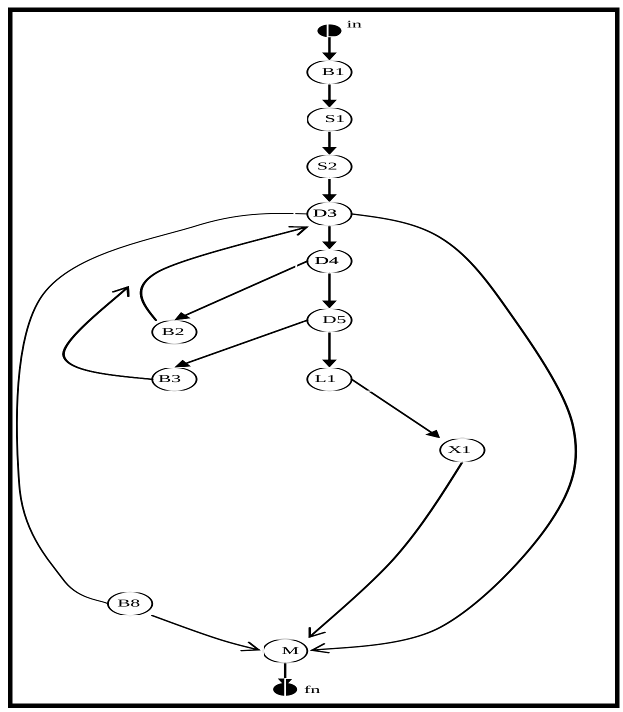

Figure 3(c) ITM3

The final ITM that is obtained is known as base path. In base path, there are many composite nodes. These composite nodes are decomposed and replace by its regions. These regions can be either a loop, a fork-join or decision-merge structures. These node- sequences are known as internal paths.

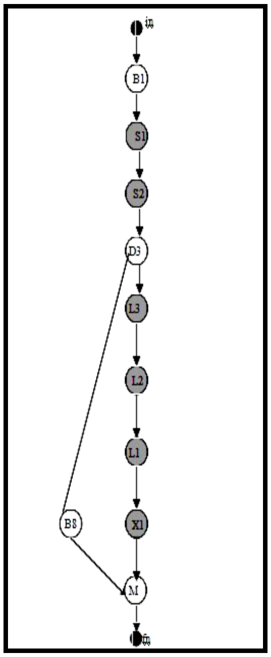

Figure 3(d) ITM4

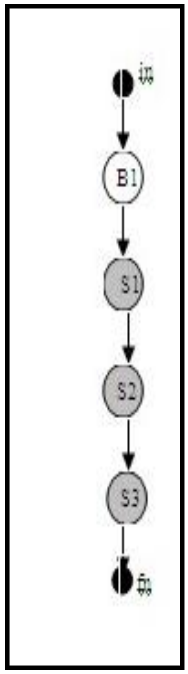

Figure 3(d) ITM5

Hereafter are generated the scenario and test cases: As illustrated in test case generation algorithm, first step is to generate base path, which is B = < in, B1, S1, S2, S3, fn> In next step, one by one composite node is expanded. The first composite is S1 i.e. a selection node, and all the possible scenarios have to be covered. Then S1 is replaced with its internal path, so two paths are generated:{<in,B1,D1,S2,S3,fn>,<in,B1,D1,fn>}Generates the test case from these scenarios. Next the node is S2; Replace S2 with its internal path. Finally after expanding all the composite nodes, following test cases result

Table 1: Test cases for composite node S1

Initial condition/state		ATM card read successfully
	Input	Expected output
1	Insert ATM card	Card read successfully
2		Unauthorized card

Table 2: Test cases for composite nodes S2

Initial condition/state		ATM card read successfully
	Input	Expected output
3	Insert pin no.	Pin read successfully
4		Wrong pin no

Table 3: Test cases for composite nodes S3

Initial condition/state		Valid card & pin no.
	Input	Expected output
5	Balance enquiry selected	Take receipt
6	Cancel pressed	Eject card
7	Cash withdrawal selected	Input amount

Table 4: Test cases for composite nodes L1, L2

Initial condition/state		Cash withdrawal selected
	Input	Expected output
8	Enter amount	Amt. exceeding day limit amt.& reselect choice
9		Res. Cash is not sufficient & reselect choice
10		Insufficient balance in account & reselect choice
11		Take cash , receipt & eject card

IV. CONCLUSION

This paper has presented a transformation method from UML state chart diagram into labeled graph and than labeled graph is transformed into intermediate model that are used to generate test cases. Existing test cases generation techniques do not include fragments and /or their nesting into test case generation. The labeled graph provides the graphical representation of each state correctly. The hierarchal structure of ITM makes it possible to generate test cases from simple path instead of doing exhaustive graph search technique. It may be underlined that this approach is beneficial when nesting of states occur in state chart diagram. Future work will concentrate on improving the adequacy of test case generation and investigating the development of a prototype tool. Moreover, a plan is afoot to include other diagrams of UML to generate test cases.

V. REFERENCES

[1]. A.Nayak, D.Samanta, “Model Based Test Cases Synthesis using UML Interaction Diagram,”.Vol. 34, March2009, pp. 1-7.

[2]. S.kansomkeat, J.offutt, A.abdyazik “A comparative Evaluation of tests generated from different UML diagrams,” Ninth ACIS International Conference on Software Engineering,A.I,Networking and parallel computing,3 Sep.2008, pp. 867-872.

[3]. S.Swain, D.Mohabatra. “The Test Case generation from UML behavioral UML models,”.International Journal of computer applications,vol.6-no.8,sep.2010,pp. 5-11.

[4]. S.Kansomkeat, W.Rivepiboon “The Automated generating test cases using UML state chart diagrams,” In proceeding of SAICSIT, oct.2003,pp. 296-300.

[5]. Booch,G.,Rumbaugh,J.andJacobson,I. “The Unified Modeling Language User Guide,”(3e),Dorling Kindersely Pvt.Ltd,New Delhi.2009.

[6]. Sapna,P.G.and Mohanty,H. "Automated Senario generation based on UML Activity Diagram", In procedding of International Conference 2008.

[7]. W.Linzhang, Y.Jiesong,"Generating Test cases from UML activity Diagram based on Gray-Box method",In procedding of the 11th Asia Pecific software Engg. Conference,30 Dec2008,pp. 209-214.

[8]. Zhan,x."A formal testing Framework for UML statecharts", Eight ACIS International conference of Software Engineering,13 Aug.2007,pp.882-887.

[9]. T.D.Trong, S.Glosh, "A systematic Approach to generate Inputs to test UML design Model,"17 International Symposium of software reliability Engineering,11 Dec.2006, pp.95-104.

[10]. T.D.Trong, R.France,"A tool supported to testing UML design models", In procedding of 10th IEEE International Conference Of Engineering of Complex Computer System ,11 Sep.2005,pp.1-10..

[11]. H.Kim,and S.kang,"Test cases generation from UML activity diagram". 8th ACIS International Conference on s software Engg, A.I,Networking and parallel computing, 13Aug.2007, pp. 556-561.

[12]. P.Samuel, A.T.Joseph, "Test sequence generation from UML sequence Diagrams".9th ACIS International Conference On software engineering, A.I,Networking and parallel computing ,3 Sep2008, pp. 879-887.

[13]. H.Y.Chen, "Transformation of UML Interaction Diagram into control specification for object oriented testing". In proceeding of National Nature Science Foundation of china,13 Aug.2007,pp.556-561.

[14]. H.Reza, K.Ogaard, "Model based testing Technique to test web Applications using statecharts", 5th International conference on information technology,18-april 2008,pp.183-188

[15]. M.Sarma, D.Kundu. "Automatic test case generation from UML sequence Diagram",15th International conference on advanced computing and communication"2007 pp.60-65.

[16]. X.Fan, Jianshu "Test case generation from UML subactivity and activity diagram,"2nd International Symposium on electronic commerce and security,23 oct.2009, pp. 244-248.