Test Case Generation Using Graph Theory in Object Oriented Systems

M. Prasanna[1] S.N. Sivanandam[2] R. Venkatesan[3]

ABSTRACT

A new approach for automated test case generation in object-oriented systems has been presented. The UML models are used as the basis for functional testing. The scope of the paper is been limited to the state-chart diagrams. This paper considers the open state chart diagram for a real time example. The models are transformed into usage models to describe the system behavior. Graph Theory technique has been employed to bring out all possible test cases of a given state chart diagram.

KEYWORDS : Software testing, Unified Modeling Language, Euler Graph, State-chart diagrams, Graph theory.

1. INTRODUCTION

Software testing includes executing a program on a set of test cases and computing the actual results with the expected results. A test case is a general software artifact that includes test case input values, expected results for the test cases, and any other inputs that are necessary to put the software system into a state that is appropriate for the test input values. Testing is an important but expensive part of software development. Improvements in software testing technique are needed to address the increasingly complex applications that are implemented by today's software systems. To test the functionality of a system, its behaviour must be understood. This can be done by creating a suitable model of the system. Object oriented modeling techniques using the UML play an important role in commercial software development .The test cases are derived by analyzing the dynamic behavior of the objects. Comprehensive testing is a trivial prerequisite for high software quality.

The use of models for the purpose of test case generation is feasible due to its extensive nature. UML can be used as the basis for functional testing, as it is an inherently discrete language that emphasizes representation of dynamic behavior of systems. Clearly, UML is a language for specifying, constructing, visualizing and documenting the artifacts of software-intensive systems. The UML diagrams are used widely to visually depict the static structure and more importantly for us, the dynamic behavior of such applications. This provides us with an excellent tool to automatically generate tests early on, during the software development life cycle.

The distinct advantage of using the UML is that they provide a convenient basis for selecting tests. UML state charts are based on finite state machines used to represent the behavior of the object. Coupled with Graph Theory, UML state charts provide a basis for test case generation.

The rest of the paper is organized as follows:

The related works done in this area are highlighted in Section 2. Our proposed methodology is outlined in Section 3. Section 4 presents an illustrative case study on Automatic Teller Machine, for which test cases have been generated using our approach followed by discussion and conclusion in sections.

[1]Research Scholar, [2]Professor & Head, [3]Professor, Department of Computer Science and Engineering, PSG College of Technology, Coimbatore, India.
E-mail : mp_psg@rediffmail.com[1]

2. Related Works

Aynur Abdurazik and Jeff Offutt [1] offer a method to develop test cases using UML specifications but their work has been limited to statecharts. Frohlich [3] has discussed about how to automate test case generation using dynamic models but input test data has to be specified manually. Harry Robinson [6] offers an introduction to model based testing and its efficiency with usage of graph theory techniques but graph traversals are manual. Jeff Offutt [7] provides a method for test data generation from state based specifications but does not guarantee full coverage.

We have proposed a simplified approach of generating test cases covering all possible cases using Euler's graph.

3. Proposed Methodology

The proposed methodologies involves the following steps:

- Analysing the real system which is to be tested and accepted by the user
- Constructing finite state machines of the components and sub components of the system
- Capturing the behaviour of components with the help of the corresponding finite states.
- Representing the state transitions as a graph which will be the basis for generating test cases
- Eulerization of the graph to close the circuits.
- Generating possible test cases from the Eulerized graph using two length pair associations.

A Flow chart of the above procedure is shown in Figure 1.

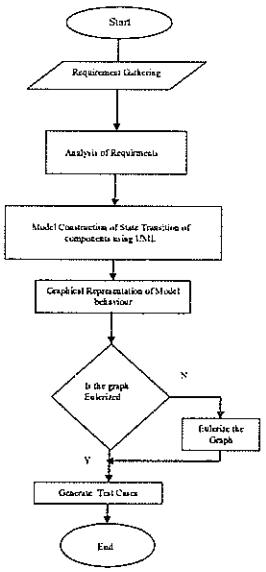


Figure 1. Flowchart of the proposed methodology

4. Case Study

Problem Statement (ATM System)

Consider the case where an user is using an Automatic Teller Machine (ATM) for executing a transaction. Initially the cursor is placed either in the Card number field where the user may type his Card number or in PIN field (i.e if the card number is read by inserting the card). While Getting PIN the user can either type his PIN value or he can move to the Card number field by pressing [tab] key. User can submit his transaction either from the Card Number field or from the PIN field. User request is validated by checking with the database, If a match occurs then the user is allowed to do transaction. If it is invalid, the user is allowed to retry operation. Retry is to be allowed for three times. If the user still submits invalid request, then the user is invalidated and not allowed any further transactions. Authenticated users proceed for further transactions, as indicated in a menu.

Step 1:

The state chart diagram for the Authentication Component of ATM system is shown in Figure 2. it represents the dynamic behavior of the ATM system during user validation.

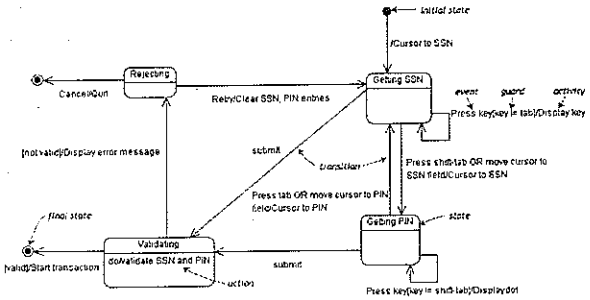


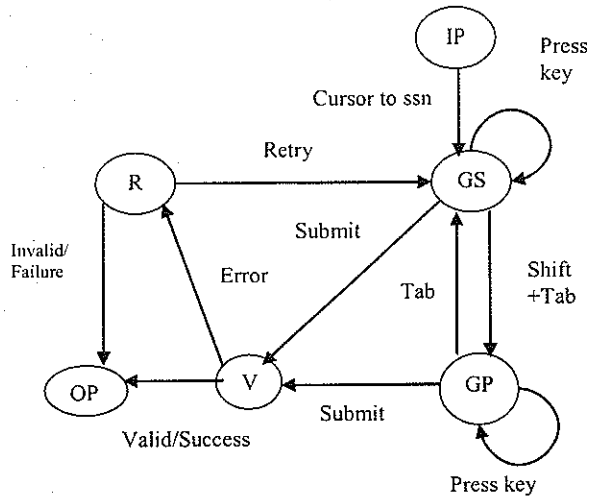
Figure 2. State chart diagram for ATM

Step 2:

To represent this in a graphical form as in Figure 3. we need to make the following modifications.

a. States are represented as nodes.

b. Actions are represented as edges.



3. Graphical representation of ATM system

For our example problem, the states and their respective nodes of the state chart diagram are shown in Table 1.

Table 1. States and their respective Nodes

States of State chart Diagram	Nodes of the Graph
Getting ssn	GS
Getting pin	GP
Validate	V
Reject	R
Input	IP
Output	OP

Step 3a :

We can see that the graph in Figure 3. is not an Euler's graph, since the in-degree of the IP node is not one less than its out-degree and the in-degree of the OP node is not one greater than its out-degree and nodes GS,GP and R have un-equal number of in-degrees and out-degrees. To trace out all possible combination of actions, we need to convert it into an Euler's graph. In-degrees and out-degrees of the graph represented in Figure 3 are shown in Table 2.

Table 2. Degree of the nodes before Eulerizing

Nodes	In-degree	Out-degree
IP	0	1
GS	4	3
GP	2	3
V	2	2
R	1	2
OP	2	0

Step 3b:

The graph in Figure 3 is not an Euler graph. To make it an Euler's graph we need to add unaffecting edges. Eulerized graph is shown in the Figure 4, where edge 'a' is added without affecting the functioning of the system. The actions and the corresponding edges of eulerized graph is shown in Table 3.

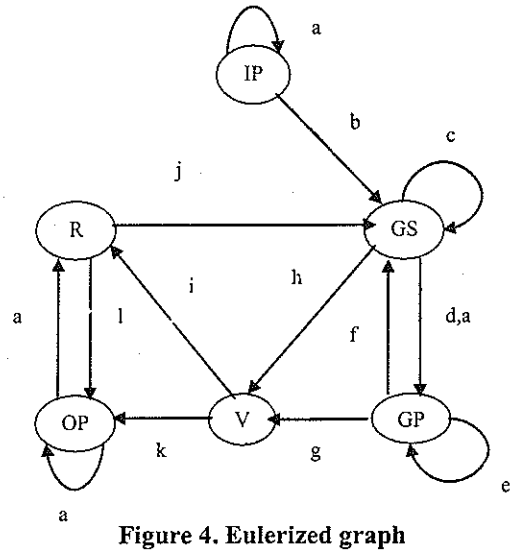


Figure 4. Eulerized graph

Table 3. Actions and their respective Edges

Actions of State chart Diagram	Edges of the Graph
Wait 1 sec	a
Cursor to SSN	b
Press key[key!=tab]/display key	c
Press shift-tab or move cursor to PIN	d
Press key[key!=shift-tab]/display asterisk	e
Press tab or move cursor to SSN/	f
Submit2	g
Submit1	h
Error/[invalid]/display error message	i
Retry/clear SSN , PIN entries	j
Correct[valid]/start transaction	k
Incorrect[invalid]/cancel	l

The in-degree and out-degree of the eulerized graph is shown in Table 4.

Table 4. Degree of the nodes after Eulerization

Nodes	In-degree	Out-degree
IP	1	2
GS	4	4
GP	3	3
V	2	2
R	2	2
OP	3	2

Step 4:

We use all the two length pairs of adjacent edges to check out all possible combination of actions from the Eulerized graph. To derive the adjacency matrix we would first note the adjacent nodes associated with each node in the graph. Adjacency matrix is often referred to as length pair association matrix. Two length adjacent edges of all the edges are shown in the Table 5.

Table 5. Two Length Pair Adjacent Edges

Edges	Adjacent Edges
a	b
b	a,c,d,h
c	a,d,h
d	e,g,f
e	f,g
f	a,c,d,h
g	i,k
h	i,k
i	j,l
j	a,c,d,h
k	a
l	a

Step 5:

The adjacency matrix is traversed and the test cases are generated. With the help of the test cases we build a test matrix that covers all the valid and invalid actions possible in the system. Test matrix is shown in the Table 6.

In the Test matrix,

V - Valid action in the sequence

I - Invalid action in the sequence

With the above case study we observe that all possible test cases can be generated by modeling the state chart diagram in to graphs.

5. Discussion and Conclusion

It has been established that UML models can be effectively used to derive test cases. This paper suggests a model based approach in dealing with behavioral aspect of the system and deriving test cases based on graph theory concepts. This approach will help software developer and tester to commence the testing process sufficiently early in the software development cycle. This approach also provides requirements traceability throughout the life cycle, as the models form the basic building blocks of system design.

State chart diagrams are used as input to trace the behaviour of the system to generate test cases. This provides an easy way to automatically generate test cases to keep pace with applications that are constantly changing and evolving. Graph theory techniques allow us to use the behavioral information stored in state chart diagrams to generate appropriate and adequate test cases. Our methodology has been illustrated with an case study of a real world system. This methodology can be extended for nested state chart diagrams and for other dynamic models. We have concentrated on open state models, using graph theory concepts.

6. References

[1] Aynur Abdurazik and Jeff Offutt. *"Generating test cases from UML specifications"*. Technical report ISE-TR-99-105, Department of Information and Software Engineering, George Mason University, Fairfax VA, 1999.

[2] Rational Software Corporation.Rational Rose 98: Using Rational Rose.Rational Rose Corporation, Cupertina CA,1998.

[3] Frohlich, P., Link, J. *"Automated test case generation from dynamic models"*. In: Proc.ECOOP 2000, LNCS 1850, Springer(2000) 427-491.

[4] Harel, D. Statecharts *"A visual formalism for complex systems. Science of Computer Programming"*, 8 (1987) 231-274

[5] Alessandar Cavarra, Jim Davies. *"Using UML for Automatic Test Generation"*. Oxford University, Computing Laboratory.

[6] Harry Robinson, 1999, *"Graph Theory techniques in Model Based testing"*, Presented at international conference on Testing computer software.

[7] Jeff Offutt, Shaoying Liu, Aynur Abdurazik, Paul Ammann, March 2003, *"Generating Test data from State based Specifications"*, The Journal of Software Testing, Verification and Reliability, 13(1):25-53.

[8] Mark Priestley, 2001, *"Practical Object-Oriented Design with UML"*, Tata McGraw Hill, pp. 195-202.

[9] Ibrahim K. El-far, 1995, *"Automated construction of Software Behavioral models"*, American University of Beirut.

[10] Ibrahim K. El-far, 2001, *"Enjoying the Perks of Model based Testing"*, STARWEST.

[11] ED Adams, Sam Guckenheimer, 27 April 2004 , *"Achieving Quality by Design partII using UML"*, The Rational Edge, IBM.

[12]Jeff Offutt, Shaoying Liu, Aynur Abdurazik, Paul Ammann, March 2003, "Generating Test data from State based Specifications", The Journal of Software Testing, Verification and Reliability, 13(1):25-53.

[13] Soumen Maity, Amiya Nayak, Marzia Zaman, Nita Bansal, Alka Srivastava, 2003, "An Improved Test Generation Algorithm for Pair-wise testing", ISSRE.

[14]Mark Blackburn, Aaron Nauman, Bob Busser (Software Productivity Consortium) and Bryan Stensvad, 2003, Defect Identification with Model-Based Test Automation, Society of Automotive Engineers; SAE 2003, March 3-6, Detroit MI.

[15]Ralphs, T.K., 8 June 1993, "On the mixed Chinese Postman Problem", School of operation research and Industrial Engineering, Cornell University.

[16]Soumen Maity, Amiya Nayak, Marzia Zaman, Nita Bansal, Alka Srivastava, 2003, "An Improved Test Generation Algorithm for Pair-wise testing", ISSRE.

Table 6 Test Case Matrix

Test-ID	a	b	c	d	e	f	g	h	i	J	k	l	Sequence	Result
1	V	V	V	V	V	V	V	I	I	I	V	I	abcdfaegk	Success
2	V	V	I	V	I	I	V	I	I	V	I	I	abcdfaegij	Retry
3	V	V	I	I	I	I	I	I	I	V	I	I	abcdfaegil	Error
4	V	V	V	I	I	I	V	I	I	I	I	I	abdfaegil	Error
5	V	V	I	V	I	I	V	I	I	I	I	I	abdfaegij	Retry
6	V	V	I	I	I	I	I	I	I	I	I	I	abchij	Retry
7	V	I	V	I	V	I	V	I	V	V	V	V	abchil	Error
8	V	I	I	V	V	I	V	I	V	V	V	V	abdefhij	Retry
9	V	I	I	I	V	I	I	I	V	V	V	V	abdefhil	Error
10	V	I	V	I	V	I	V	I	V	I	V	V	abcdgil	Error
11	V	I	I	V	V	I	V	I	V	I	V	V	abcdgij	Retry
12	V	I	I	I	V	I	I	I	V	I	V	V	abhil	Error
13	V	I	V	I	V	I	I	V	I	I	V	V	abhij	Retry
14	V	I	I	V	V	I	I	V	I	I	V	V	abafcdegil	Error
15	V	I	I	I	V	V	I	I	I	I	V	V	abafcdegk	Success
16	V	I	V	I	V	I	V	I	V	V	I	I	abafcdeil	Error
17	V	I	I	V	V	I	V	I	V	V	I	I	abdefchil	Error
18	V	I	V	I	V	I	V	I	V	I	I	I	abdefchij	Retry
19	V	I	I	V	V	I	V	I	V	I	I	I	abdefchk	Success
20	V	I	I	I	V	I	I	I	V	V	I	I	abcdegij	Retry
21	V	I	I	I	V	I	I	I	V	I	I	I	abcdegil	Error
22	V	I	V	I	V	I	I	V	I	I	I	I	abcdegk	Success
23	V	I	I	V	V	I	I	V	I	I	I	I	abcdefhij	Retry
24	V	I	I	I	V	V	I	I	I	I	I	I	abcdefhk	Success
25	V	V	I	I	I	V	I	I	I	V	I	I	abcdefhil	Error

Authors' Biography :

M.Prasanna is currently working as Lecturer in the department of CSE, PSG College of Technology. He has been in teaching for the past 6 years. His areas of research include Computing and Testing. He is a life member of ISTE and SSI. Currently, he is doing MS [By Research] programme under Anna university.

S.N. Sivanadam, Born in Coimbatore District, TamilNadu state in India. in 1942, received the BE in Electrical and Electronics Engineering from Madras University Chennai, MSc (Engg) from Madras University, Chennai and PhD in Electrical and Electronics Engineering From Madras University, Chennai in 1964, 1966 and 1982 respectively. He is fellow of Institution of Engineers, India. He is life member of ISTE, SSI and CSI. His research interests lie in the areas of Computer Networking, Modelling and Simulation, Network Security, Neural Networks, Genetic Algorithm. He has published 400 Technical papers in National and International Journals and Conferences. He has published seven Technical Books.

R. Venkatesan is currently working as a professor in the department of CSE, PSG College of Technology, Coimbatore. He is a fellow of the Institution of Engineers. He has B.E(Hons) in Mechanical Engineering, ME in Industrial Engineering and MS in Computer and Information Science from University of Michigan, USA. He has over 18 years of Software Industry experience and Information Science from University of Michigan, USA. He has over 18 years of Software Industry experience and 5 years of Teaching Experience. His areas of interest are Software Engineering, Modeling and Simulation , Data Structure and Algorithms. He has published papers in National and International Journals. He is life member of ISTE, SSI and CSI. He has guided projects for UG and PG students.