Volume 6, Issue 1, January 2016 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Auto Test Generation from UML Use Case State Chart Diagrams

Anbunathan R*
Test Manager and Research Scholar
Bharathiar University, Tamilnadu,
India

Anirban Basu
Professor, Department of CSE
APS College of Engineering, Bangalore,
India

Abstract— Due to increasing use of OOAD techniques, UML-based testing has been gaining attention for Functional Testing. In this paper, a novel method to generate test cases from UML State diagrams is presented. Use case State chart diagram is parsed to extract information about States and Transitions. Using this information, LCSAJ test cases and MC/DC test cases are generated automatically. The application of the method is illustrated with a case study. The advantages of the proposed method are also discussed.

Keywords— UML diagram, UML testing; Test case generation; State chart diagram; Model Based Testing; Test automation.

I. INTRODUCTION

Today UML is being widely used for designing systems and UML State chart diagram is playing a major role in modelling the dynamic behaviour of an application or of an embedded system. Product development environment is tightly coupled with UML tools for designing system behaviour. Test Engineers need to create test cases from State chart diagrams to test the behaviour of the system, by inputting different combinations of test data.

Many methods have been proposed for generating test cases from UML State chart diagram. The method proposed in this paper discusses test case generation from State chart diagram and generates Multiple Conditions/Decision coverage (MC/DC) test cases and LCSAJ based test cases. The method is more effective than others with effectiveness measured in terms of state coverage, transition coverage, and path coverage. The test cases generated by this method help to achieve 100% test coverage without spending much effort.

II. RELATED WORK

This section discusses other methods that have been proposed for UML State chart based testing. In [4], Samuel et al. proposed a method to generate test cases from UML State diagram. This approach can handle events, guards and transitions. Test data also generated automatically using function minimization technique.

In [18], Ranjitha et al. proposed a method, to convert UML State diagram to Extended Finite State Machine (EFSM) Graph, which is used for generating test cases by using a tool.

There are some other methods [29][32][37] that generate test cases from UML diagrams using a similar approach. But none of the methods discuss generation of LCSAJ test cases.

In [13], Offutt and Abdurazik proposed a method to generate system test cases from State-based formal specifications. Test cases are generated automatically from UML specifications using UMLTest tool.

In [17], Kim et al. proposed a method for generating test cases for class testing using UML State chart diagrams. State charts are transformed to extended FSMs (EFSM) and flow graphs, and then conventional data flow analysis techniques are applied to generate test cases.

In [33], Wang et al. proposed a method for converting UML diagrams into FSM diagrams. XMI files are obtained from these FSMs, which are used for automatic generation of test cases.

III. ILLUSTRATION OF THE METHOD

This section illustrates test case creation from State chart diagram with a case study.

3.1 Case study

Purchase Online System (POS) is taken as an example and as shown below State chart diagram is drawn for POS in Figure 1.

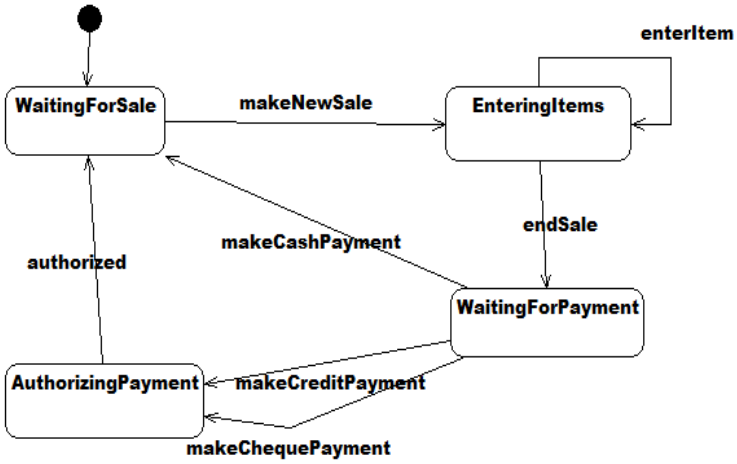

Figure 1. State chart diagram for POS system.

The corresponding Control Flow Graph (CFG) is drawn as shown in Figure 2.

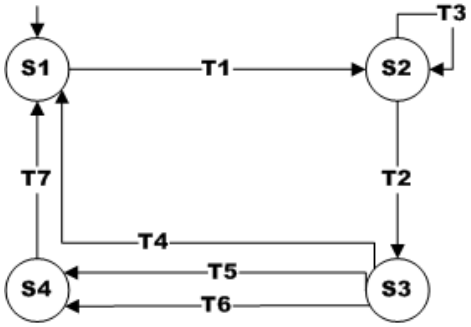

Figure 2. Control Flow Graph (CFG) derived from POS State chart diagram.

3.1.1 Adjacency/Incidence Matrices

The Adjacency matrix and Incidence matrix for CFG are shown in Tables 1 and 2 respectively. These matrices are useful to traverse through all States and Transitions.

Table 1. Adjacency matrix of CFG

	S1	*S2*	*S3*	*S4*
S1	0	1	0	0
S2	0	1	1	0
S3	1	0	0	2
S4	1	0	0	0

Table 2. Incidence matrix of CFG

	T1	*T2*	*T3*	*T4*	*T5*	*T6*	*T7*
S1	1	0	0	1	0	0	1
S2	1	1	1	0	0	0	0
S3	0	1	0	1	1	1	0
S4	0	0	0	0	1	1	1

3.2 LCSAJ based test case generation method

Linear Code Sequence and Jump (LCSAJ) is a linear sequence of executable code commencing either at the start of the program or at a point to which control flow may jump. Same principle is applied in UML State diagram, where linear control flow from one State to another is realized through Transitions. In case of code, each statement in the code is considered as node. Analogous to this, each State is considered as one node, in the case of State diagram based representation. Basically an LCSAJ consists of a body of code through which the flow of control proceeds sequentially and then terminated by a jump in the control flow. In the proposed approach, an LCSAJ represents linear control flow from one State to another State and then the control flow is terminated by a jump to the third State through a transition or jump to second State itself in case of self transition and so on. Each LCSAJ yields one test case. The Start State, Transition and jump to the State of each LCSAJ constitute precondition, test description and expected result of the corresponding test case.

3.2.1 LCSAJ table from POS State chart diagram

LCSAJ table is created by traversing from Initial State to other States through transitions. LCSAJ is formed when control is transferred to new State, if a transition is encountered. Table 3 shows LCSAJ table derived from POS State diagram as shown in Figure2. In LCSAJ1 in the Table 1, control is transferred to S2 from S1, when transition T1 occurs and then jump to State 2 happens through Transition 3. Every jump to a new State formulates one LCSAJ.

Table 3. LCSAJ table derived from POS State diagram

LCSAJ Number	*Start State*	*Finish State*	*Jump To State*	*Transitions*
1	S1	S2	S2	T1, T3
2	S2	S3	S4	T2,T5
3	S2	S3	S4	T2,T6
4	S2	S3	S1	T2,T4
5	S4	S1		T7
6	S1	S2	S3	T1,T2
7	S3	S4	S1	T6,T7
8	S3	S4	S1	T5,T7
9	S3	S1		T4

3.2.2 Converting LCSAJ Table to test cases

LCSAJ can be converted to test cases as shown in Table 4. Start State in LCSAJ table is mapped with Pre-condition in test case. Similarly Finish State is mapped with Expected Result. Transitions traced through LCSAJ algorithm are mapped with Description in test case.

Table 4. Test cases generated from LCSAJ table

LCSAJ Test cases			
Sl.no	*Precondition*	*Description*	*Expected result*
1	S3	T4	S1
2	S4	T7	S1
3	S1	T1,T2	S3
4	S1	T1,T3	S2
5	S2	T2,T4	S1
6	S2	T2,T5	S4
7	S2	T2,T6	S4
8	S3	T5,T7	S1
9	S3	T6,T7	S1

3.2.3 Metrics from LCSAJ

Calculation of Test Effectiveness Ratio (TER):

TER1 = Number of States covered by test data/total number of States

TER2 = Number of Transitions covered by the test data/total number of transitions

TER3 = Number of LCSAJs executed by the test data/total number of LCSAJs

Advantage:

When TER3 = 100% has been achieved it follows that TER2 = 100% and TER1 = 100% have also been achieved.

3.3 Decision table based test case generation method

Table 5 shows decision table derived from POS State diagram. The States constitute variables in the decision table. The transitions constitute values for these variables.

Table 5. Decision table derived from POS State diagram

S1	*S2*	*S3*	*S4*
T1	T2	T4	T7
	T3+T2	T5	
		T6	

3.3.1 Converting Decision table to test cases

Decision table can be converted to MC/DC test cases as shown in Table 6, using Pairwise testing tool (Allpairs tool) [38].

Table 6. Test cases generated from decision table

Test case	*S1*	*S2*	*S3*	*S4*	*Expected result*
1	T1	T2	T4	T7	S2,S3,S1
2	T1	T3+T2	T5	T7	S2,S2,S3,S4,S1
3	T1	T2	T6	T7	S2,S3,S4,S1
4	~T1	T3+T2	T4	~T7	S2,S2,S3,S1
5	~T1	T2	T5	~T7	S2,S3,S4,S1
6	~T1	T3+T2	T6	~T7	S2,S2,S3,S4,S1

3.4 Automatic Generation of Test Cases

The detailed steps for generating test cases from State chart diagram are given in section 3.4.1 through 3.4.3.

3.4.1 Process Flow Diagram from State chart diagram

Figure 3 illustrates the steps involved in generating test cases automatically from State chart diagram.

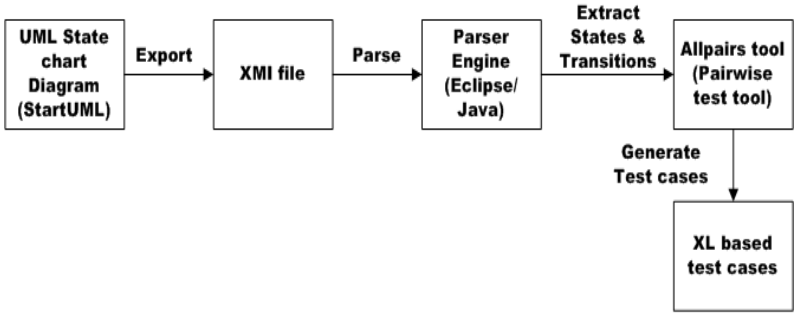

Figure 3. Process steps to generate test cases from State chart.

3.4.2 Automatic generation of LCSAJ Test cases from State chart diagram

XMI file [11] is exported from corresponding State chart diagram in StarUML [6] tool environment. From XMI file, States, incoming Transitions and outgoing Transitions are identified. Using Incidence matrix, Transition traversal table is created as shown in Table 7. In this table, Transitions and their corresponding Start State and End State are tabulated.

Table 7. Transition travel table for POS CFG

Transition	*Start State*	*End State*
T1	S1	S2
T2	S2	S3
T3	S2	S2
T4	S3	S1
T5	S3	S4
T6	S3	S4
T7	S4	S1

From Transition travel table, LCSAJ table is constructed. Start States and End States are directly taken from Transition travel table to LCSAJ table. 'Jump to' States are identified by searing End State in Start State column of Transition travel table. 'Jump to' State is other than Start and End States, in case of non-self transitions. In case of self transitions, 'Jump to' State is same as End State. If End State is Initial State, there is no 'Jump to' State. LCSAJ table for POS State diagram is automatically generated as shown in Table 8.

Table 8. Auto generated LCSAJ table from POS State diagram

Start state	*Finish State*	*Jump to state*	*Transitions*
WaitingForSale	EnteringItems	WaitingForPayment	makeNewSale,endSale
WaitingForSale	EnteringItems	EnteringItems	makeNewSale,enterItem
EnteringItems	WaitingForPayment	WaitingForSale	endSale,makeCashPayment
EnteringItems	WaitingForPayment	AuthorizingPayment	endSale,makeCreditPayment
EnteringItems	WaitingForPayment	AuthorizingPayment	endSale,makeChequePayment
AuthorizingPayment	WaitingForSale	WaitingForSale	authorized
WaitingForPayment	WaitingForSale	WaitingForSale	makeCashPayment
WaitingForPayment	AuthorizingPayment	WaitingForSale	makeCreditPayment,authorized
WaitingForPayment	AuthorizingPayment	WaitingForSale	makeChequePayment,authorized

From LCSAJ table, LCSAJ test cases are generated as shown in Table 9.

Table 9. Generated test cases from LCSAJ table

Test Case ID	*Pre Condition*	*Description*	*Expected result*
TC1	WaitingForSale	makeNewSale,endSale	WaitingForPayment
TC2	WaitingForSale	makeNewSale,enterItem	EnteringItems
TC3	EnteringItems	endSale,makeCashPayment	WaitingForSale
TC4	EnteringItems	endSale,makeCreditPayment	AuthorizingPayment
TC5	EnteringItems	endSale,makeChequePayment	AuthorizingPayment
TC6	AuthorizingPayment	Authorized	WaitingForSale
TC7	WaitingForPayment	makeCashPayment	WaitingForSale
TC8	WaitingForPayment	makeCreditPayment,authorized	WaitingForSale
TC9	WaitingForPayment	makeChequePayment,authorized	WaitingForSale

The algorithm for generating LCSAJ test cases is shown in Figure 4.

Algorithm for automatic LCSAJ test case generation

1. Start State = Initial State
2. Push all outgoing Transitions to Stack.
3. If Stack is empty terminate algorithm.
4. If Stack is non-empty, pop one Transition. Find Start/End States for this Transition using Transition traversal table
5. Search End State in Start State column of Transition traversal table and find corresponding 'Jump to' State and 'Jump through' Transition.
6. If 'Jump through' Transition is not self Transition, then 'Jump to' State is other than End State. If it is self Transition, 'Jump to' State is same as End State. If End State is Initial State, then no 'Jump to' State.
7. Start State = 'Jump to' State
8. Repeat from Step 2.

Figure 4. Algorithm to generate LCSAJ test cases.

3.4.3 Automatic generation of MC/DC test cases from State chart diagram

Decision table is created using States and their corresponding outgoing Transitions. States constitute variables in the decision table, and Transitions constitute values for each variable. Table 10 shows Decision table generated from POS State diagram.

Table 10. Auto generated Decision table for POS

WaitingForSale	*Entering-Items*	*WaitingForPayment*	*Authorizing Payment*
Make-NewSale	endSale	makeCash-Payment	authorized
	enterItem+endSale	makeCredit-Payment	
		makeChequePayment	

Using 'All pairs' tool, MC/DC test cases are generated from the Decision table as shown in Table 11.

Table 11. Generated MC/DC test cases by 'Allpairs' tool

TC ID	*Waiting ForSale*	*Entering Items*	*Waiting ForPayment*	*Authorizing Payment*	*Expected Result*
1	makeNewSale	endSale	makeCashPayment	authorized	1.EnteringItems 2.WaitingForPayment 3.WaitingForSale 4.WaitingForSale
2	makeNewSale	enterItem	makeCreditPayment	authorized	1.EnteringItems 2.EnteringItems 3.AuthorizingPayment 4.WaitingForSale

3	makeNewSale	endSale	makeChequePayment	authorized	1.EnteringItems 2.WaitingForPayment 3.AuthorizingPayment 4.WaitingForSale
4	~makeNewSale	enterItem	makeCashPayment	~authorized	1.EnteringItems 2.EnteringItems 3.WaitingForSale 4.WaitingForSale
5	~makeNewSale	endSale	makeCreditPayment	~authorized	1.EnteringItems 2.WaitingForPayment 3.AuthorizingPayment 4.WaitingForSale
6	~makeNewSale	enterItem	makeChequePayment	~authorized	1.EnteringItems 2.EnteringItems 3.AuthorizingPayment 4.WaitingForSale

IV. COMPARISON WITH OTHER METHODS

There are others who have considered a State diagram as input for test case generation. An example can be seen in [28], where test suites can be automatically generated from State charts. This is done by mapping State chart elements to the STRIPS planning language. The application of the State of the art planning tool graph plan yields the different test cases as solutions to a planning problem. This method has following limitations:

1. The expected system responses have to be added to the test sequence manually to yield complete test cases.
2. Test case coverage is not ensured.
3. Test cases are not optimized.

The proposed method has the following advantages:

1. Expected Results are automatically added to ensure completeness of test cases.
2. Test case coverage is ensured by LCSAJ algorithm.
3. Test cases are optimized by generating MC/DC test cases using Pairwise test approach.

There are several research projects [4][18][33] proposing concepts for UML based test tools. However, most of them generate exhaustive test cases, which in turn significantly lower the chance of those concepts being accepted in industry projects. The proposed method addresses generating Multiple Conditions/Decisions Coverage (MC/DC) test cases from State diagram, which are optimized in number, at the same time ensuring 100% transition and State coverage. The usage of 'Allpairs' tool ensures reducing number of test cases being generated.
In Agile environment, it is recommended to use LCSAJ test cases during developmental stage and use MC/DC test cases for regression testing, once software is stabilized.

V. EXPERIMENTAL RESULTS

The proposed approach is deployed in few applications and results are obtained. The following applications are considered for experimentation:

a. Account system
b. Borrow book
c. Currency controller
d. Ice vending machine
e. Safe home system
f. Simple ATM (SATM)
g. Triangle program
h. Wiper controller

5.1 Description

The brief descriptions of all applications are given in the following section:

5.1.1 Account system

An Account system helps user to open 'new' account. Once account is created user can do various transactions such as balance checking, debit money, credit etc. If the balance is maintained less than 0, then the status is changed to 'overdrawn'. If the account is not accessed for more than 5 years, then the status is changed to 'locked'. Also Account system allows user to close the account. The State diagram of the Account system is shown in Figure A-1 and its corresponding automatically generated LCSAJ and MC/DC test cases are given in Table A-1 and A-2 respectively in Appendix A.

5.1.2 Borrow book

The Borrow book application allows user to search book in the database. If book is found in the database, user can reserve the book in his name. The Borrow book application has login feature and checks authentication of the user. The

State diagram of the Borrow book is shown in Figure B-1 and its corresponding automatically generated LCSAJ and MC/DC test cases are given in Table B-1 and B-2 respectively in Appendix B.

5.1.3 Currency converter

The Currency converter application allows user to convert currency from USD or Indian rupees to equivalent other country currencies. It allows user to enter input value and select target country. It throws error, if either input value not entered or target country is not selected. The State diagram of the Currency converter is shown in Figure C-1 and its corresponding automatically generated LCSAJ and MC/DC test cases are given in Table C-1 and C-2 respectively in Appendix C.

5.1.4 Ice cream vending machine

The Ice cream vending machine allows user to purchase ice creams automatically. It allows user to select different flavour of ice creams such as Vanilla, Chocolate, Strawberry and Butterscotch etc. It calculates money based on selected flavour and number of ice creams ordered. When user inserts money into the slot, it calculates balance amount and returns back. The State diagram of the Ice vending machine is shown in Figure D-1 and its corresponding automatically generated LCSAJ and MC/DC test cases are given in Table D-1 and D-2 respectively in Appendix D.

5.1.5 Safe home system

The Safe home system is a security system that helps user to monitor home. It alerts home owner in case of any intruder entering home, through various mechanisms such as sending SMS, making emergency call, activating alarm, video recording and blinking control panel. The State diagram of the Safe home system is shown in Figure E-1 and its corresponding automatically generated LCSAJ and MC/DC test cases are given in Table E-1 and E-2 respectively in Appendix E.

5.1.6 Simple ATM system

The Simple ATM system provides banking transactions such as withdraw money, deposit money, balance checking, print mini statement etc. User requires a valid debit card and need to enter valid PIN number to avail banking services. The State diagram of the Simple ATM system is shown in Figure F-1 and its corresponding automatically generated LCSAJ and MC/DC test cases are given in Table F-1 and F-2 respectively in Appendix F.

5.1.7 Triangle program

The Triangle program displays triangle type such as Isosceles, Scalene, Equilateral based on the values of the sides a, b, c. It displays an error message, in case of invalid entry. The State diagram of the Triangle program is shown in Figure G-1 and its corresponding automatically generated LCSAJ and MC/DC test cases are given in Table G-1 and G-2 respectively in Appendix G.

5.1.8 Wiper controller

The Wiper controller allows user to set different wiper speed by changing position of lever and dial. The lever position can be changed to off, inter, low and high. When lever position is set to inter, dial positions can be changed to 1, 2 and 3. The State diagram of the Wiper controller is shown in Figure H-1 and its corresponding automatically generated LCSAJ and MC/DC test cases are given in Table H-1 and H-2 respectively in Appendix H.

5.2 Summary

Table 12 shows summary of LCSAJ and MC/DC test cases derived from State diagrams of various applications. Column B shows MC/DC test cases possible to derive from State diagram. Column C shows optimized number of test cases using 'Allpairs' tool.

Table 12. Summary of LCSAJ and MC/DC test cases

Project name	*LCSAJ Test cases (A)*	*MC/DC Test cases*		*Total Test cases (A+C)*
		Before optimization (B)	*After optimization (C)*	
Account system	12	112	28	40
Borrow book	7	4	4	11
Currency converter	19	32	10	29
Ice vending machine	11	8	6	17
Safe home	17	16	8	25
Simple ATM	22	32	10	32
Triangle program	13	5	5	18
Wiper controller	16	10	10	26

5.3 Mutation analysis

The effectiveness of generated test cases for Triangle program is checked using fault injection technique called mutation analysis [39][40]. Mutants are created from Triangle program after injecting errors in program to make program faulty. If

test case set is capable of capturing these errors, then mutants are killed by tests. The mutation analysis report after creating mutants for Triangle program is shown in Table 13.

Totally 18 test cases failed because of 5 mutants. Mutants 3, 4, 5 each have 4 test cases failed out of 18 test cases executed. This shows even though test cases are generated from design, any defect injected by developer is easily captured by these cases.

Table 13. Mutation analysis for Triangle program test cases

Mutation number	*Change in code*		*Number of test cases failed in Mutants*		*Total Fails per Mutant*
	Correct	*Buggy*	*LCSAJ*	*MC/DC*	
1	!(a>0 && b>0 && c>0)	!(a>0 || b>0 || c>0)	2	1	3
2	(a<b+c)&&(b<c+a)&&(c<a+b)	(a<b+c)||(b<c+a)||(c<a+b)	2	1	3
3	a==b && b==c && c==a	a==b || b==c || c==a	3	1	4
4	(a==b && b<>c)||(a==c && a<>b)||(b==c && a<>c)	(a==b && b<>c) &&(a==c && a<>b) &&(b==c && a<>c)	3	1	4
5	a<>b && b<>c && c<>a	a==b && b<>c && c<>a	3	1	4
	Total fails category wise		13	5	18

VI. CONCLUSIONS

In this paper, a method has been proposed to generate functional test cases from LCSAJ table and MC/DC test cases from Decision table. These two tables are automatically derived from UML Use case State chart diagram. As State chart diagram is developed early in the development cycle, early generation of test cases is possible by the proposed method. Besides, the proposed method is more prone to automation and reduces effort for writing exhaustive test cases.

The case study discussed here has shown that the proposed method can be used for applications in both PC based and in embedded environments. As State chart diagram represent dynamic behaviour of the system, and generating MC/DC test cases helps to test system behaviour under various input conditions. LCSAJ based test generation approach ensures 100% test coverage.

REFERENCES

[1] P. Jorgensen, Software Testing: A Craftsman's Approach. CRC Press,2002.

[2] B. Beizer, Software Testing Techniques, 2nd ed. Van Nostrand Reinhold, 1990.

[3] Craig Larman, "Applying UML and patterns ", Addison Wesley, 2000.

[4] P. Samuel R. Mall A.K. Bothra, "Automatic test case generation using unified modeling language (UML) state diagrams", The Institution of Engineering and Technology, 2008.

[5] Qaisar A. Malik, Dragos¸ Trus¸can, Johan Lilius,"Using UML Models and Formal Verification in Model-Based Testing", 17th IEEE International Conference and Workshops on Engineering of Computer-Based Systems 2010.

[6] Star UML Tool. http://staruml.sourceforge.net/en/, Jul. 2011.

[7] Padma Iyenghar1, Elke Pulvermueller1, Clemens Westerkamp,"Towards Model-Based Test Automation for Embedded Systems Using UML and UTP", IEEE ETFA 2011.

[8] Object Constraint Language 2.0 is available from Object Mangement Group's web site http://www.omg.org/

[9] Vinaya Sawant, Dragos¸ Ketan Shah,"Automatic Generation of Test Cases from UML Models", International Conference on Technology Systems and Management 2011.

[10] G.J. Myers, C.sandler, T.Badgett, and T.M.Thomas. "The art of software Testing" , 2nd Edition.Wiley,2004

[11] OMG, "XML Metadata Interchange (XMI),v2.1",2004.

[12] I. K. El-Far and J. A. Whittaker, "Model-based software testing," Encyclopedia on Software Engineering, 2001.

[13] J. Offutt and A. Abdurazik, "Generating Tests from UML Specifications", Second International Conference on the Unified Modeling Language, Springer, New York 1999, pp.416-429

[14] W. M. Ho, J.-M. Jquel, A. L. Guennec, and F. Pennaneac'h, "UMLAUT: An extendible UML transformation framework," in Automated Software Engineering, 1999, pp. 275–278. [Online]. Available: citeseer.ist.psu.edu/ho99umlaut.html

[15] T. J´eron and P. Morel, "Test generation derived from model-checking," in CAV '99: Proceedings of the 11th International Conference on Computer Aided Verification. London, UK: Springer-Verlag, 1999, pp. 108–121.

[16] L. Bousquet, H. Martin, and J. Jzquel, "Conformance testing from uml specifications." [Online]. Available: citeseer.ist.psu.edu/683853.html

[17] KIM Y.G., HONG H.S., BAE D.H., ET AL.: 'Test cases generation from UML state diagram', Proc. Softw., 1999, 146, (4),pp. 187–192

[18] Ranjita Swain, Vikas Panthi and Durga Prasad Mohapatra, "Automatic Test case Generation From UML State Chart Diagram", International Journal of Computer Applications (0975 – 8887) Volume 42– No.7, March 2012.

[19] Gnesi Stefania, Latella, Diego, and Massink Mieke. 2004. Formal test-case generation for UML statecharts, Proceedings of the Ninth IEEE International Conference on Engineering Complex Computer Systems Navigating Complexity in the e-Engineering Age, 2004, pp.75 – 84.

[20] Joanne M. Atlee and John Gannon, "State-Based Model Checking of Event-Driven System Requirements", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 19, NO. 1, JANUARY 1993.

[21] John Joseph Chilenski and Steven P. Miller, "Applicability of modified conditioddecision coverage to software testing", Software Engineering Journal, September 1994.

[22] Luqi, Hongji Yang and Xiaodong Zhang, "Constructing an Automated Testing Oracle: An Effort to Produce Reliable Software", Computer Software and Applications Conference, 1994.

[23] Apfelbaum and Larry, " Automated functional test generation", AUTOTESTCON '95.

[24] Mark Stephenson, Tom Lynch and Steve Walters, "Using Advanced Tools to Automate the Design, Generation and Execution of Formal Qualification Testing", AUTOTESTCON '96.

[25] Peter Savage, Steve Waiters and Mark Stephenson, "Automated Test Methodology for Operational Flight Programs", Aerospace Conference, 1997. Proceedings.

[26] T. Savor and R.E. Seviora, "An Approach to Automatic Detection of Software Failures in Real-Time Systems", Real-Time Technology and Applications Symposium, 1997. Proceedings.

[27] A. Jeerson Outt, Yiwei Xiong and Shaoying Liu, "Criteria for Generating Specication-based Tests", Proceedings Fifth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'99).

[28] Peter Fröhlich and Johannes Link, "Automated Test Case Generation from Dynamic Models", Proceedings ECOOP, 2000.

[29] Diego Latella and Mieke Massink, "A Formal Testing Framework for UML Statechart Diagrams Behaviours: From Theory to Automatic Verification", Proceedings of the 6th IEEE International Symposium on High Assurance Systems Engineering, 2001.

[30] Philippe Chevalley and Pascale Thevenod-Fosse, "An Empirical Evaluation of Statistical Testing Designed from UML State Diagrams: the Flight Guidance System Case Study", Proceedings of 12th International Symposium on Software Reliability Engineering, 2001.

[31] Khaled El-Fakih, Anton Kolomeez, Svetlana Prokopenko and Nina Yevtushenko, "Extended Finite State Machine Based Test Derivation Driven By User Defined Faults", International Conference on Software Testing, Verification, and Validation, 2008.

[32] TSUN S. CHOW, "Testing Software Design Modeled by Finite-State Machines", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. SE-4, NO. 3, MAY 1978.

[33] Xi Wang, Liang Guo and Huaikou Miao, "An Approach to Transforming UML Model to FSM Model for Automatic Testing", International Conference on Computer Science and Software Engineering, 2008.

[34] Qurat-ul-ann Farooq, Muhammad Zohaib Z.Iqbal, Zafar I Malik and Zafar I Malik, "A Model-Based Regression Testing Approach for Evolving Software Systems with Flexible Tool Support", 17th IEEE International Conference and Workshops on Engineering of Computer-Based Systems, 2010.

[35] Reinhard Hametner, Dietmar Winkler, Thomas Östreicher, Natascha Surnic and Stefan Biffl, "Selecting UML Models for Test-Driven Development along the Automation Systems Engineering Process", IEEE Conference on Emerging Technologies and Factory Automation, 2010.

[36] Reinhard Hametner, Benjamin Kormann, Birgit Vogel-Heuser, Dietmar Winkler and Alois Zoitl, "Test Case Generation Approach for Industrial Automation Systems", 5th International Conference on Automation, Robotics and Applications, 2011.

[37] Manuj Aggarwal and Sangeeta Sabharwal, "Test Case Generation from UML State Machine Diagram: A Survey", Third International Conference on Computer and Communication Technology, 2012.

[38] ALL PAIRS Tool. http://www.satisfice.com/tools.shtml.

[39] S. Kansomkeat and W. Rivepiboon, "Automated-generating test case using UML statechart diagrams", Proc. SAICSIT 2003, ACM 2003 pp. 296 – 300, 2003.

[40] Demillo, Lipton and Sayward, "Hints on Test Data Selection:Help for the Practicing Programmer", IEEE, 1978.

APPENDIX

This section contains State diagrams of applications taken for experiment and their corresponding LCSAJ and MC/DC test cases.

Appendix A: Account system

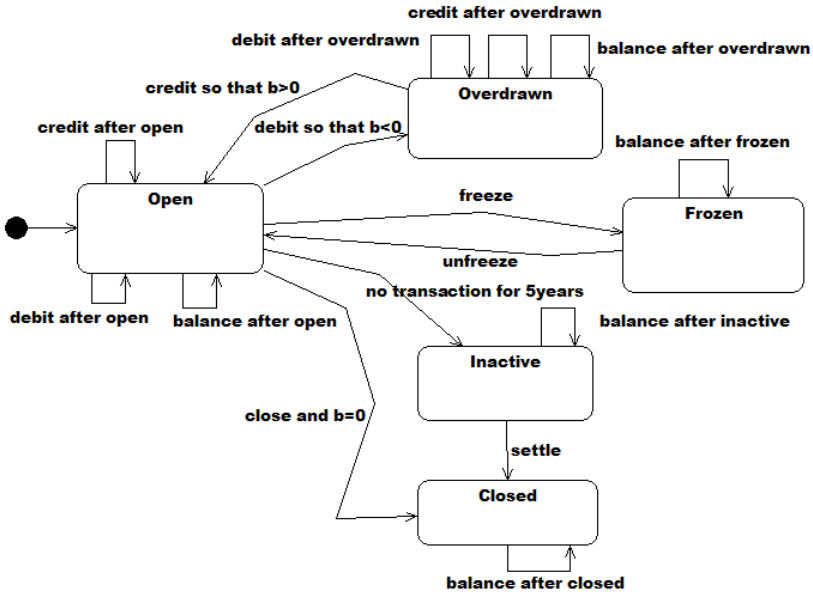

Figure A-1: State diagram for Account system

Table A-1. LCSAJ test cases from Account System State diagram

TestCase ID	*Pre Condition*	*Description*	*Expected result*
TC1	Open	freeze,unfreeze	Open
TC2	Open	freeze,balance after frozen	Frozen
TC3	Frozen	unfreeze	Open
TC4	Open	debit so that b<0,credit so that b>0	Open
TC5	Open	debit so that b<0,debit after overdrawn	Overdrawn
TC6	Open	debit so that b<0,credit after overdrawn	Overdrawn
TC7	Open	debit so that b<0,balance after overdrawn	Overdrawn
TC8	Overdrawn	credit so that b>0	Open
TC9	Open	no transaction for 5years,settle	Closed
TC10	Open	no transaction for 5years,balance after inactive	Inactive
TC11	Open	close and b=0,balance after closed	Closed
TC12	Inactive	settle,balance after closed	Closed

Table A-2. MC/DC test cases from Account System State diagram

TC ID	Open	Over drawn	Frozen	Inactive	Closed	Expected Result
1	credit after open	credit so that b>0	unfreeze	settle	balance after closed	1.Open 2.Open 3.Open 4.Closed 5.Closed
2	debit after open	debit after overdrawn	balance after frozen	balance after inactive	balance after closed	1.Open 2.Overdrawn 3.Frozen 4.Inactive 5.Closed
3	balance after open	credit after overdrawn	unfreeze	balance after inactive	balance after closed	1.Open 2.Overdrawn 3.Open 4.Inactive 5.Closed
4	freeze	balance after overdrawn	balance after frozen	settle	balance after closed	1.Frozen 2.Overdrawn 3.Frozen 4.Closed 5.Closed
5	debit so that b<0	credit so that b>0	balance after frozen	balance after inactive	balance after closed	1.Overdrawn 2.Open 3.Frozen 4.Inactive 5.Closed

6	no transaction for 5years	debit after overdrawn	unfreeze	settle	balance after closed	1.Inactive 2.Overdrawn 3.Open 4.Closed 5.Closed
7	close and b=0	credit after overdrawn	balance after frozen	settle	balance after closed	1.Closed 2.Overdrawn 3.Frozen 4.Closed 5.Closed
8	credit after open	balance after overdrawn	unfreeze	balance after inactive	~balance after closed	1.Open 2.Overdrawn 3.Open 4.Inactive 5.Closed
9	debit after open	credit so that b>0	unfreeze	settle	~balance after closed	1.Open 2.Open 3.Open 4.Closed 5.Closed
10	balance after open	debit after overdrawn	balance after frozen	settle	~balance after closed	1.Open 2.Overdrawn 3.Frozen 4.Closed 5.Closed
11	freeze	credit after overdrawn	unfreeze	balance after inactive	~balance after closed	1.Frozen 2.Overdrawn 3.Open 4.Inactive 5.Closed
12	debit so that b<0	balance after overdrawn	unfreeze	settle	~balance after closed	1.Overdrawn 2.Overdrawn 3.Open 4.Closed 5.Closed
13	no transaction for 5years	credit so that b>0	balance after frozen	balance after inactive	~balance after closed	1.Inactive 2.Open 3.Frozen 4.Inactive 5.Closed
14	close and b=0	debit after overdrawn	unfreeze	balance after inactive	~balance after closed	1.Closed 2.Overdrawn 3.Open 4.Inactive 5.Closed
15	credit after open	credit after overdrawn	balance after frozen	~settle	~balance after closed	1.Open 2.Overdrawn 3.Frozen 4.Closed 5.Closed
16	debit after open	balance after overdrawn	~balance after frozen	~balance after inactive	~balance after closed	1.Open 2.Overdrawn 3.Frozen 4.Inactive 5.Closed
17	balance after open	credit so that b>0	~unfreeze	~settle	~balance after closed	1.Open 2.Open 3.Open 4.Closed 5.Closed
18	freeze	debit after overdrawn	~balance after frozen	~balance after inactive	~balance after closed	1.Frozen 2.Overdrawn 3.Frozen 4.Inactive 5.Closed
19	debit so that b<0	credit after overdrawn	~unfreeze	~balance after inactive	~balance after closed	1.Overdrawn 2.Overdrawn 3.Open 4.Inactive 5.Closed
20	no transaction for 5years	balance after overdrawn	~balance after frozen	~settle	~balance after closed	1.Inactive 2.Overdrawn 3.Frozen 4.Closed 5.Closed
21	close and b=0	credit so that b>0	~balance after frozen	~balance after inactive	~balance after closed	1.Closed 2.Open 3.Frozen 4.Inactive 5.Closed
22	credit after open	debit after overdrawn	~balance after frozen	~balance after inactive	~balance after closed	1.Open 2.Overdrawn 3.Frozen 4.Inactive 5.Closed
23	debit after open	credit after overdrawn	~unfreeze	~settle	~balance after closed	1.Open 2.Overdrawn 3.Open 4.Closed 5.Closed
24	balance after open	balance after overdrawn	~balance after frozen	~balance after inactive	~balance after closed	1.Open 2.Overdrawn 3.Frozen 4.Inactive 5.Closed
25	freeze	credit so that b>0	~unfreeze	~settle	~balance after closed	1.Frozen 2.Open 3.Open 4.Closed 5.Closed
26	debit so that b<0	debit after overdrawn	~balance after frozen	~settle	~balance after closed	1.Overdrawn 2.Overdrawn 3.Frozen 4.Closed 5.Closed
27	no transaction for 5years	credit after overdrawn	~unfreeze	~balance after inactive	~balance after closed	1.Inactive 2.Overdrawn 3.Open 4.Inactive 5.Closed
28	close and b=0	balance after overdrawn	~unfreeze	~settle	~balance after closed	1.Closed 2.Overdrawn 3.Open 4.Closed 5.Closed

Appendix B: Borrow book

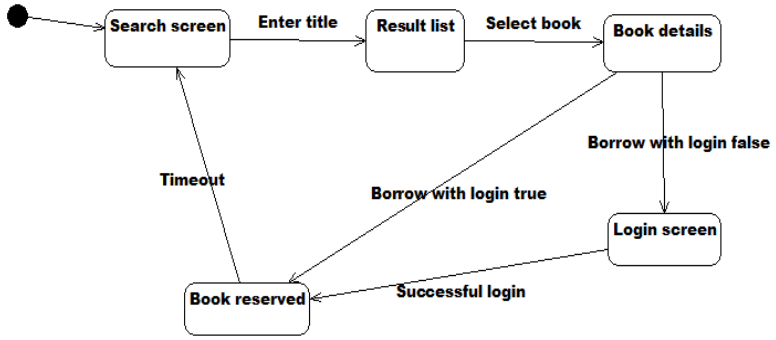

Figure B-1: State diagram for Borrow book

Table B-1. LCSAJ test cases from Borrow book State diagram

TestCaseID	*PreCondition*	*Description*	*Expected result*
TC1	Search screen	Enter title,Select book	Book details
TC2	Result list	Select book,Borrow with login false	Login screen
TC3	Result list	Select book,Borrow with login true	Book reserved
TC4	Book details	Borrow with login false,Successful login	Book reserved
TC5	Book reserved	Timeout	Search screen
TC6	Login screen	Successful login,Timeout	Search screen
TC7	Book details	Borrow with login true,Timeout	Search screen

Table B-2. MC/DC test cases from Borrow book State diagram

TC ID	*Search screen*	*Result list*	*Book details*	*Login screen*	*Book reserved*	*Expected Result*
1	Enter title	Select book	Borrow with login false	Successful login	Timeout	1.Result list 2.Book details 3.Login screen 4.Book reserved 5.Search screen
2	Enter title	Select book	Borrow with login true	Successful login	Timeout	1.Result list 2.Book details 3.Book reserved 4.Book reserved 5.Search screen

Appendix C: Currency Converter

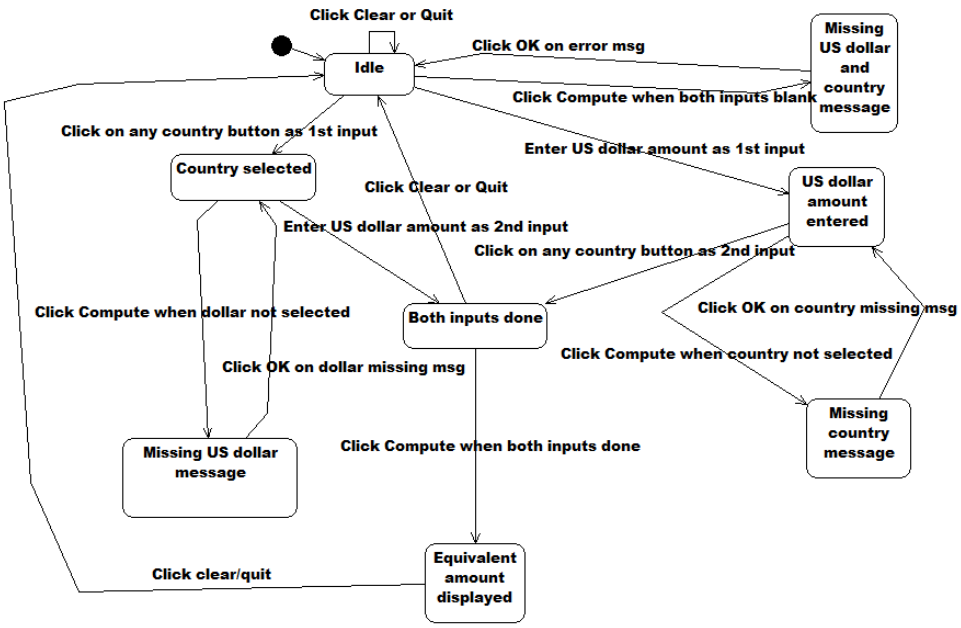

Figure C-1: State diagram for Currency converter

Table C-1. LCSAJ test cases from Currency Converter State diagram

TestCase ID	*Pre Condition*	*Description*	*Expected result*
TC1	Idle	Click Compute when both inputs blank,Click OK on error msg	Idle
TC2	Missing US dollar and country message	Click OK on error msg	Idle
TC3	Idle	Click on any country button as 1st input,Click Compute when dollar not selected	Missing US dollar message
TC4	Idle	Click on any country button as 1st input,Enter US dollar amount as 2nd input	Both inputs done
TC5	Idle	Enter US dollar amount as 1st input,Click Compute when country not selected	Missing country message
TC6	Idle	Enter US dollar amount as 1st input,Click on any country button as 2nd input	Both inputs done
TC7	Country selected	Click Compute when dollar not selected,Click OK on dollar missing msg	Country selected
TC8	Missing US dollar message	Click OK on dollar missing msg,Click Compute when dollar not selected	Missing US dollar message
TC9	Missing US dollar message	Click OK on dollar missing msg,Enter US dollar amount as 2nd input	Both inputs done
TC10	US dollar amount entered	Click Compute when country not selected,Click OK on country missing msg	US dollar amount entered
TC11	Missing country message	Click OK on country missing msg,Click Compute when country not selected	Missing country message
TC12	Missing country message	Click OK on country missing msg,Click on any country button as 2nd input	Both inputs done
TC13	Both inputs done	Click Compute when both inputs done,Click clear/quit	Idle
TC14	Country selected	Enter US dollar amount as 2nd input,Click Compute when both inputs done	Equivalent amount displayed
TC15	Country selected	Enter US dollar amount as 2nd input,Click Clear or Quit	Idle
TC16	US dollar amount entered	Click on any country button as 2nd input,Click Compute when both inputs done	Equivalent amount displayed
TC17	US dollar amount entered	Click on any country button as 2nd input,Click Clear or Quit	Idle
TC18	Both inputs done	Click Clear or Quit	Idle
TC19	Equivalent amount displayed	Click clear/quit	Idle

Table C-2. MC/DC test cases from Currency Converter State diagram

TC ID	Idle	Country selected	US dollar amount entered	Both inputs done	Equivalent amount displayed	Missing US dollar message	Missing country message	Missing US dollar and country message	Expected Result
1	Click Compute	Click Compute	Click Compute	Click Compute	Click clear/quit	Click OK on	Click OK on	Click OK on	1.Missing US dollar and country message

	when both inputs blank	when dollar not selected	when country not selected	when both inputs done		dollar missing msg	country missing msg	error msg	2.Missing US dollar message 3.Missing country message 4.Equivalent amount displayed 5.Idle 6.Country selected 7.US dollar amount entered 8.Idle
2	Click on any country button as 1st input	Enter US dollar amount as 2nd input	Click on any country button as 2nd input	Click Clear or Quit	Click clear/quit	Click OK on dollar missing msg	Click OK on country missing msg	Click OK on error msg	1.Country selected 2.Both inputs done 3.Both inputs done 4.Idle 4.Idle 5.Idle 6.Country selected 7.US dollar amount entered 8.Idle
3	Enter US dollar amount as 1st input	Click Compute when dollar not selected	Click on any country button as 2nd input	Click Compute when both inputs done	Click clear/quit	Click OK on dollar missing msg	Click OK on country missing msg	Click OK on error msg	1.US dollar amount entered 2.Missing US dollar message 3.Both inputs done 4.Equivalent amount displayed 5.Idle 6.Country selected 7.US dollar amount entered 8.Idle
4	Click Clear or Quit	Enter US dollar amount as 2nd input	Click Compute when country not selected	Click Clear or Quit	Click clear/quit	Click OK on dollar missing msg	Click OK on country missing msg	Click OK on error msg	1.Idle 1.Idle 2.Both inputs done 3.Missing country message 4.Idle 4.Idle 5.Idle 6.Country selected 7.US dollar amount entered 8.Idle
5	Click Compute when both inputs blank	Enter US dollar amount as 2nd input	Click on any country button as 2nd input	Click Compute when both inputs done	~Click clear/quit	~Click OK on dollar missing msg	~Click OK on country missing msg	~Click OK on error msg	1.Missing US dollar and country message 2.Both inputs done 3.Both inputs done 4.Equivalent amount displayed 5.Idle 6.Country selected 7.US dollar amount entered 8.Idle
6	Click on any country button as 1st input	Click Compute when dollar not selected	Click Compute when country not selected	Click Clear or Quit	~Click clear/quit	~Click OK on dollar missing msg	~Click OK on country missing msg	~Click OK on error msg	1.Country selected 2.Missing US dollar message 3.Missing country message 4.Idle 4.Idle 5.Idle 6.Country selected 7.US dollar amount entered 8.Idle
7	Enter US dollar amount as 1st input	Enter US dollar amount as 2nd input	Click Compute when country not selected	Click Clear or Quit	~Click clear/quit	~Click OK on dollar missing msg	~Click OK on country missing msg	~Click OK on error msg	1.US dollar amount entered 2.Both inputs done 3.Missing country message 4.Idle 4.Idle 5.Idle 6.Country selected 7.US dollar amount entered 8.Idle
8	Click Clear or Quit	Click Compute when dollar not selected	Click on any country button as 2nd input	Click Compute when both inputs done	~Click clear/quit	~Click OK on dollar missing msg	~Click OK on country missing msg	~Click OK on error msg	1.Idle 1.Idle 2.Missing US dollar message 3.Both inputs done 4.Equivalent amount displayed 5.Idle 6.Country selected 7.US dollar amount

									entered 8.Idle
9	Click Compute when both inputs blank	~Click Compute when dollar not selected	~Click Compute when country not selected	Click Clear or Quit	~Click clear/quit	~Click OK on dollar missing msg	~Click OK on country missing msg	~Click OK on error msg	1.Missing US dollar and country message 2.Missing US dollar message 3.Missing country message 4.Idle 4.Idle 5.Idle 6.Country selected 7.US dollar amount entered 8.Idle
10	Click on any country button as 1st input	~Enter US dollar amount as 2nd input	~Click on any country button as 2nd input	Click Compute when both inputs done	~Click clear/quit	~Click OK on dollar missing msg	~Click OK on country missing msg	~Click OK on error msg	1.Country selected 2.Both inputs done 3.Both inputs done 4.Equivalent amount displayed 5.Idle 6.Country selected 7.US dollar amount entered 8.Idle

Appendix D: Ice cream Vending Machine

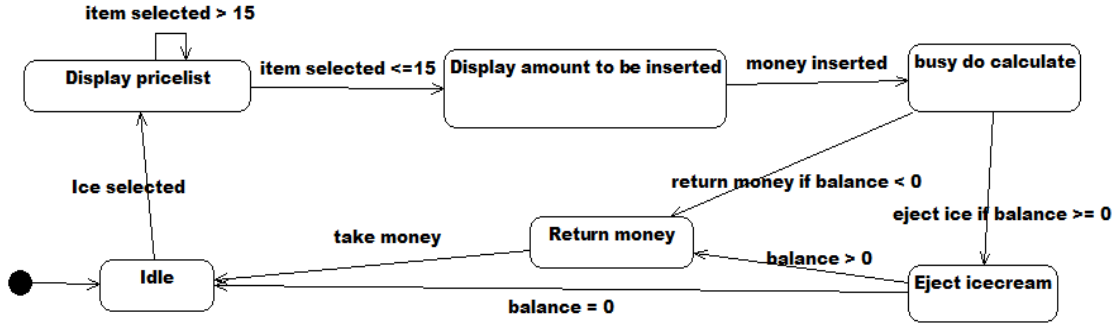

Figure D-1: State diagram for Ice cream vending machine

Table D-1. LCSAJ test cases from Ice cream Vending Machine State diagram

TestCase ID	*Pre Condition*	*Description*	*Expected result*
TC1	Idle	Ice selected,item selected <=15	Display amount to be inserted
TC2	Idle	Ice selected,item selected > 15	Display pricelist
TC3	Display pricelist	item selected <=15,money inserted	busy do calculate
TC4	Display amount to be inserted	money inserted,eject ice if balance >= 0	Eject icecream
TC5	Display amount to be inserted	money inserted,return money if balance < 0	Return money
TC6	busy do calculate	eject ice if balance >= 0,balance = 0	Idle
TC7	busy do calculate	eject ice if balance >= 0,balance > 0	Return money
TC8	Eject icecream	balance = 0	Idle
TC9	Eject icecream	balance > 0,take money	Idle
TC10	busy do calculate	return money if balance < 0,take money	Idle
TC11	Return money	take money	Idle

Table D-2. MC/DC test cases from Ice cream Vending Machine State diagram

TC ID	*Idle*	*Display pricelist*	*Display amount to be inserted*	*busy do calculate*	*Eject icecream*	*Return money*	*Expected Result*
1	Ice selected	item selected <=15	money inserted	eject ice if balance >= 0	balance = 0	take money	1.Display pricelist 2.Display amount to be inserted 3.busy do calculate 4.Eject icecream

							5.Idle 6.Idle
2	Ice selected	item selected > 15	money inserted	return money if balance < 0	balance > 0	take money	1.Display pricelist 2.Display pricelist 3.busy do calculate 4.Return money 5.Return money 6.Idle
3	~Ice selected	item selected <=15	~money inserted	return money if balance < 0	balance = 0	~take money	1.Display pricelist 2.Display amount to be inserted 3.busy do calculate 4.Return money 5.Idle 6.Idle
4	~Ice selected	item selected > 15	~money inserted	eject ice if balance >= 0	balance > 0	~take money	1.Display pricelist 2.Display pricelist 3.busy do calculate 4.Eject icecream 5.Return money 6.Idle
5	~Ice selected	item selected <=15	~money inserted	~eject ice if balance >= 0	balance > 0	~take money	1.Display pricelist 2.Display amount to be inserted 3.busy do calculate 4.Eject icecream 5.Return money 6.Idle
6	~Ice selected	item selected > 15	~money inserted	~return money if balance < 0	balance = 0	~take money	1.Display pricelist 2.Display pricelist 3.busy do calculate 4.Return money 5.Idle 6.Idle

Appendix E: Safe home system

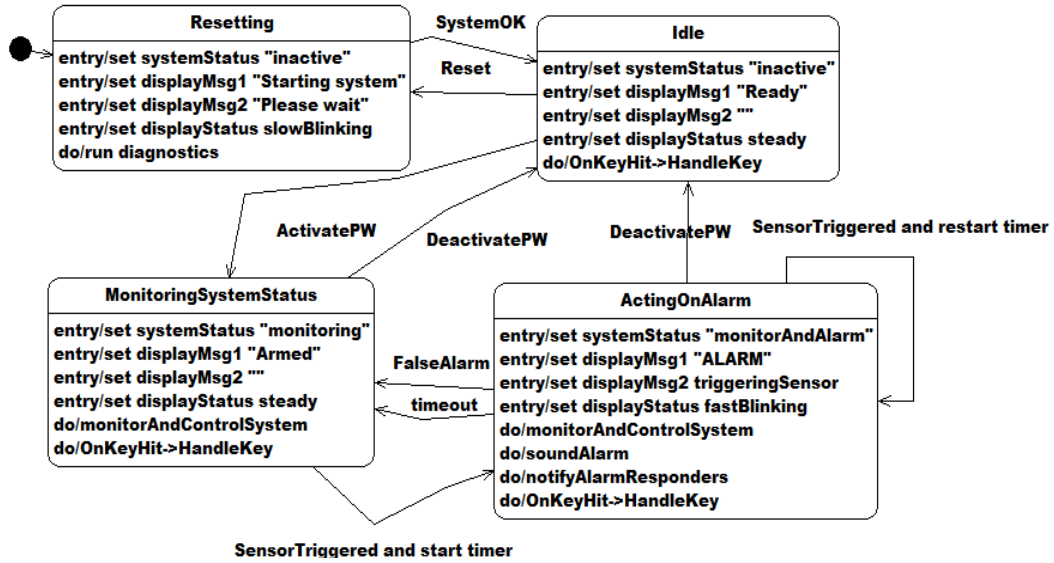

Figure E-1: State diagram for Safe home system

Table E-1. LCSAJ test cases from Safe home system State diagram

TestCase ID	*PreCondition*	*Description*	*Expected result*
TC1	Resetting	SystemOK,Reset	Resetting
TC2	Resetting	SystemOK,ActivatePW	MonitoringSystemStatus
TC3	Idle	Reset	Resetting
TC4	Idle	ActivatePW,DeactivatePW	Idle
TC5	Idle	ActivatePW,SensorTriggered and start timer	ActingOnAlarm
TC6	MonitoringSystemStatus	DeactivatePW,Reset	Resetting
TC7	MonitoringSystemStatus	DeactivatePW,ActivatePW	MonitoringSystemStatus
TC8	ActingOnAlarm	FalseAlarm,DeactivatePW	Idle
TC9	ActingOnAlarm	FalseAlarm,SensorTriggered and start timer	ActingOnAlarm
TC10	ActingOnAlarm	timeout,DeactivatePW	Idle
TC11	ActingOnAlarm	timeout,SensorTriggered and start timer	ActingOnAlarm
TC12	MonitoringSystemStatus	SensorTriggered and start timer,FalseAlarm	MonitoringSystemStatus
TC13	MonitoringSystemStatus	SensorTriggered and start	MonitoringSystemStatus

		timer,timeout	
TC14	MonitoringSystemStatus	SensorTriggered and start timer,DeactivatePW	Idle
TC15	MonitoringSystemStatus	SensorTriggered and start timer,SensorTriggered and restart timer	ActingOnAlarm
TC16	ActingOnAlarm	DeactivatePW,Reset	Resetting
TC17	ActingOnAlarm	DeactivatePW,ActivatePW	MonitoringSystemStatus

Table E-2. MC/DC test cases from Safe home system State diagram

TC ID	*Resetting*	*Idle*	*Acting On Alarm*	*MonitoringSystem Status*	*Expected Result*
1	System OK	Reset	FalseAlarm	DeactivatePW	1.Idle 2.Resetting 3.MonitoringSystemStatus 4.Idle 4.Idle
2	System OK	ActivatePW	timeout	SensorTriggered and start timer	1.Idle 2.MonitoringSystemStatus 3.MonitoringSystemStatus 4.ActingOnAlarm
3	System OK	Reset	DeactivatePW	SensorTriggered and start timer	1.Idle 2.Resetting 3.Idle 3.Idle 4.ActingOnAlarm
4	System OK	ActivatePW	SensorTriggered and restart timer	DeactivatePW	1.Idle 2.MonitoringSystemStatus 3.ActingOnAlarm 4.Idle 4.Idle
5	~System OK	ActivatePW	FalseAlarm	SensorTriggered and start timer	1.Idle 2.MonitoringSystemStatus 3.MonitoringSystemStatus 4.ActingOnAlarm
6	~System OK	Reset	timeout	DeactivatePW	1.Idle 2.Resetting 3.MonitoringSystemStatus 4.Idle 4.Idle
7	~System OK	ActivatePW	DeactivatePW	DeactivatePW	1.Idle 2.MonitoringSystemStatus 3.Idle 3.Idle 4.Idle 4.Idle
8	~System OK	Reset	SensorTriggered and restart timer	SensorTriggered and start timer	1.Idle 2.Resetting 3.ActingOnAlarm 4.ActingOnAlarm

Appendix F: Simple ATM system

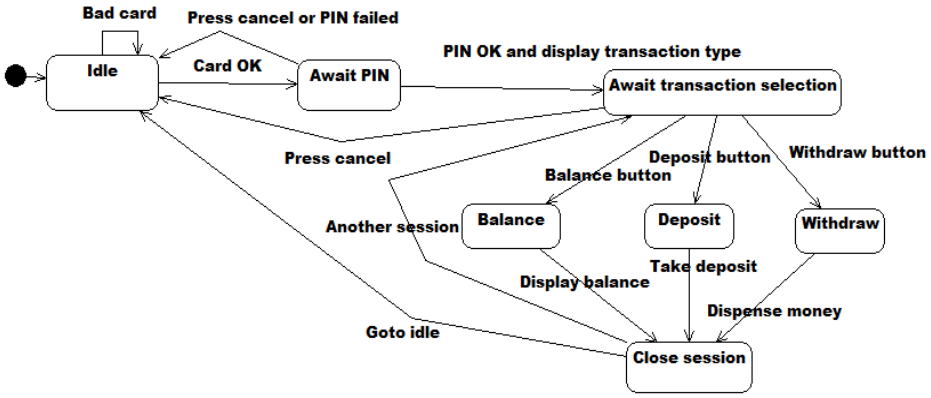

Figure F-1: State diagram for Simple ATM system

Table F-1. LCSAJ test cases from Simple ATM System State diagram

TestCase ID	*Pre Condition*	*Description*	*Expected result*
TC1	Idle	Card OK,PIN OK and display transaction type	Await transaction selection
TC2	Idle	Card OK, Press cancel or PIN failed	Idle

TC3	Await PIN	PIN OK and display transaction type,Deposit button	Deposit
TC4	Await PIN	PIN OK and display transaction type,Balance button	Balance
TC5	Await PIN	PIN OK and display transaction type,Withdraw button	Withdraw
TC6	Await PIN	PIN OK and display transaction type,Press cancel	Idle
TC7	Await transaction selection	Deposit button,Take deposit	Close session
TC8	Await transaction selection	Balance button,Display balance	Close session
TC9	Await transaction selection	Withdraw button,Dispense money	Close session
TC10	Balance	Display balance,Another session	Await transaction selection
TC11	Balance	Display balance,Goto idle	Idle
TC12	Deposit	Take deposit,Another session	Await transaction selection
TC13	Deposit	Take deposit,Goto idle	Idle
TC14	Withdraw	Dispense money,Another session	Await transaction selection
TC15	Withdraw	Dispense money,Goto idle	Idle
TC16	Close session	Another session,Deposit button	Deposit
TC17	Close session	Another session,Balance button	Balance
TC18	Close session	Another session,Withdraw button	Withdraw
TC19	Close session	Another session,Press cancel	Idle
TC20	Close session	Goto idle	Idle
TC21	Await transaction selection	Press cancel	Idle
TC22	Await PIN	Press cancel or PIN failed	Idle

Table F-2. MC/DC test cases from Simple ATM System State diagram

TC ID	*Idle*	*Await PIN*	*Await transaction selection*	*Balance*	*Deposit*	*With draw*	*Close session*	*Expected Result*
1	Card OK	PIN OK and display transaction type	Deposit button	Display balance	Take deposit	Dispense money	Another session	1.Await PIN 2.Await transaction selection 3.Deposit 4.Close session 5.Close session 6.Close session 7.Await transaction selection

2	Bad card	Press cancel or PIN failed	Balance button	Display balance	Take deposit	Dispense money	Goto idle	1.Idle 2.Idle 3.Balance 4.Close session 5.Close session 6.Close session 7.Idle
3	Card OK	Press cancel or PIN failed	Withdraw button	Display balance	Take deposit	Dispense money	Another session	1.Await PIN 2.Idle 3.Withdraw 4.Close session 5.Close session 6.Close session 7.Await transaction selection
4	Bad card	PIN OK and display transaction type	Press cancel	Display balance	Take deposit	Dispense money	Goto idle	1.Idle 2.Await transaction selection 3.Idle 4.Close session 5.Close session 6.Close session 7.Idle
5	Bad card	Press cancel or PIN failed	Deposit button	~Display balance	~Take deposit	~Dispense money	Another session	1.Idle 2.Idle 3.Deposit 4.Close session 5.Close session 6.Close session 7.Await transaction selection
6	Card OK	PIN OK and display transaction type	Balance button	~Display balance	~Take deposit	~Dispense money	Goto idle	1.Await PIN 2.Await transaction selection 3.Balance 4.Close session 5.Close session 6.Close session 7.Idle
7	Bad card	PIN OK and display transaction type	Withdraw button	~Display balance	~Take deposit	~Dispense money	Goto idle	1.Idle 2.Await transaction selection 3.Withdraw 4.Close session 5.Close session 6.Close session 7.Idle
8	Card OK	Press cancel or PIN failed	Press cancel	~Display balance	~Take deposit	~Dispense money	Another session	1.Await PIN 2.Idle 3.Idle 4.Close session 5.Close session 6.Close session 7.Await transaction selection
9	~Card OK	~PIN OK and display transaction type	Deposit button	~Display balance	~Take deposit	~Dispense money	Goto idle	1.Await PIN 2.Await transaction selection 3.Deposit 4.Close session 5.Close session 6.Close session

10	~Bad card	~ Press cancel or PIN failed	Balance button	~Display balance	~Take deposit	~Dispense money	Another session	7.Idle 1.Idle 2.Idle 3.Balance 4.Close session 5.Close session 6.Close session 7.Await transaction selection

Appendix G: Triangle program

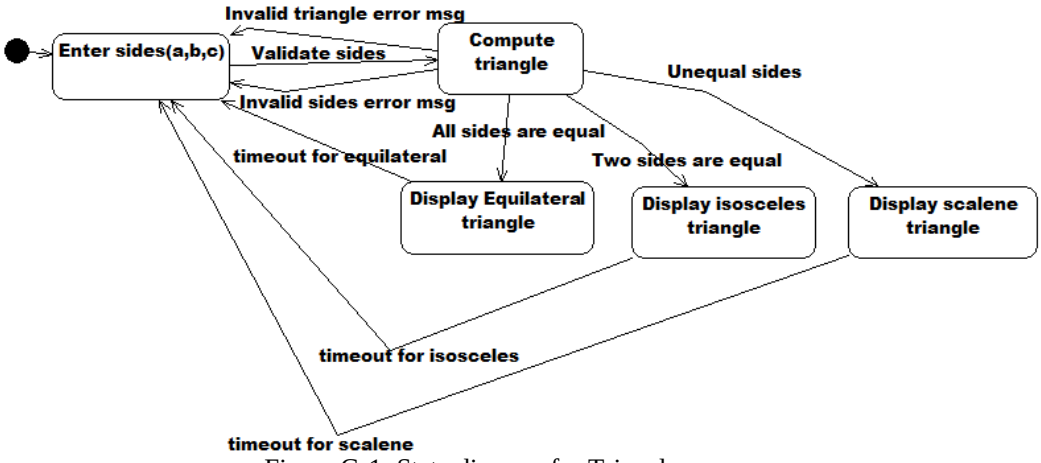

Figure G-1: State diagram for Triangle program

Table G-1. LCSAJ test cases from Triangle program State diagram

TestCase ID	*Pre Condition*	*Description*	*Expected result*
TC1	Enter sides(a,b,c)	Validate sides,All sides are equal	Display Equilateral triangle
TC2	Enter sides(a,b,c)	Validate sides,Two sides are equal	Display isosceles triangle
TC3	Enter sides(a,b,c)	Validate sides,Unequal sides	Display scalene triangle
TC4	Enter sides(a,b,c)	Validate sides,Invalid sides error msg	Enter sides(a,b,c)
TC5	Enter sides(a,b,c)	Validate sides,Invalid triangle error msg	Enter sides(a,b,c)
TC6	Compute triangle	All sides are equal,timeout for equilateral	Enter sides(a,b,c)
TC7	Compute triangle	Two sides are equal,timeout for isosceles	Enter sides(a,b,c)
TC8	Compute triangle	Unequal sides,timeout for scalene	Enter sides(a,b,c)
TC9	Display Equilateral triangle	timeout for equilateral	Enter sides(a,b,c)
TC10	Display isosceles triangle	timeout for isosceles	Enter sides(a,b,c)
TC11	Display scalene triangle	timeout for scalene	Enter sides(a,b,c)
TC12	Compute triangle	Invalid sides error msg	Enter sides(a,b,c)
TC13	Compute triangle	Invalid triangle error msg	Enter sides(a,b,c)

Table G-2. MC/DC test cases from Triangle program State diagram

TC ID	*Enter sides (a,b,c)*	*Compute triangle*	*Display Equilateral triangle*	*Display isosceles triangle*	*Display scalene triangle*	*Expected Result*
1	Validate sides	All sides are equal	timeout for equilateral	timeout for isosceles	timeout for scalene	1.Compute triangle 2.Display Equilateral triangle 3.Enter sides(a,b,c) 4.Enter sides(a,b,c) 5.Enter sides(a,b,c)
2	Validate sides	Two sides are equal	timeout for equilateral	timeout for isosceles	timeout for scalene	1.Compute triangle 2.Display isosceles triangle 3.Enter sides(a,b,c) 4.Enter sides(a,b,c) 5.Enter sides(a,b,c)
3	Validate sides	Unequal sides	timeout for equilateral	timeout for isosceles	timeout for scalene	1.Compute triangle 2.Display scalene triangle 3.Enter sides(a,b,c) 4.Enter sides(a,b,c) 5.Enter sides(a,b,c)
4	Validate sides	Invalid sides error msg	timeout for equilateral	timeout for isosceles	timeout for scalene	1.Compute triangle 2.Enter sides(a,b,c) 3.Enter sides(a,b,c) 4.Enter sides(a,b,c) 5.Enter sides(a,b,c)
5	Validate sides	Invalid triangle error msg	timeout for equilateral	timeout for isosceles	timeout for scalene	1.Compute triangle 2.Enter sides(a,b,c) 3.Enter sides(a,b,c) 4.Enter sides(a,b,c) 5.Enter sides(a,b,c)

Appendix H: Wiper controller

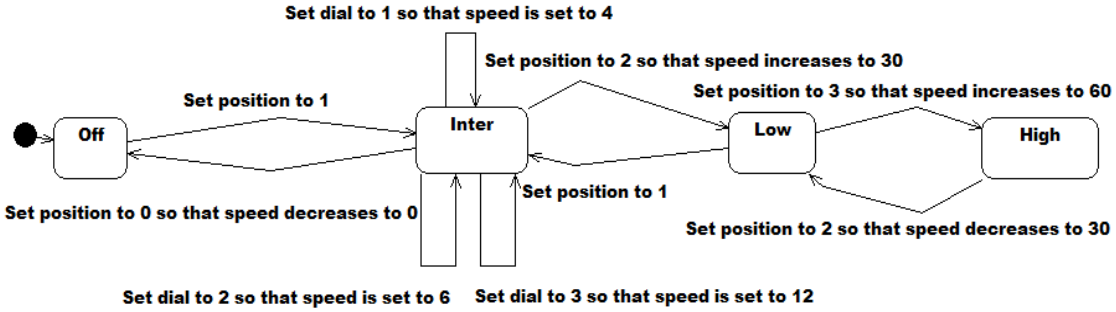

Figure H-1: State diagram for Wiper controller

Table H-1. LCSAJ test cases from Wiper Controller State diagram

TestCase ID	*Pre Condition*	*Description*	*Expected result*
TC1	Off	Set position to 1,Set position to 2 so that speed increases to 30	Low
TC2	Off	Set position to 1,Set position to 0 so that speed decreases to 0	Off
TC3	Off	Set position to 1,Set dial to 3 so that speed is set to 12	Inter
TC4	Off	Set position to 1,Set dial to 2 so that speed is set to 6	Inter
TC5	Off	Set position to 1,Set dial to 1 so that speed is set to 4	Inter
TC6	Inter	Set position to 2 so that speed increases to 30,Set position to 3 so that speed increases to 60	High
TC7	Inter	Set position to 2 so that speed increases to 30,Set position to 1	Inter
TC8	Low	Set position to 3 so that speed increases to 60,Set position to 2 so that speed decreases to 30	Low

TC9	High	Set position to 2 so that speed decreases to 30,Set position to 3 so that speed increases to 60	High
TC10	High	Set position to 2 so that speed decreases to 30,Set position to 1	Inter
TC11	Low	Set position to 1,Set position to 2 so that speed increases to 30	Low
TC12	Low	Set position to 1,Set position to 0 so that speed decreases to 0	Off
TC13	Low	Set position to 1,Set dial to 3 so that speed is set to 12	Inter
TC14	Low	Set position to 1,Set dial to 2 so that speed is set to 6	Inter
TC15	Low	Set position to 1,Set dial to 1 so that speed is set to 4	Inter
TC16	Inter	Set position to 0 so that speed decreases to 0	Off

Table H-2. MC/DC test cases from Wiper Controller State diagram

TC ID	*Off*	*Inter*	*Low*	*High*	*Expected Result*
1	Set position to 1	Set position to 2 so that speed increases to 30	Set position to 3 so that speed increases to 60	Set position to 2 so that speed decreases to 30	1.Inter 1.Inter 2.Low 3.High 4.Low
2	Set position to 1	Set position to 0 so that speed decreases to 0	Set position to 1	Set position to 2 so that speed decreases to 30	1.Inter 1.Inter 2.Off 3.Inter 3.Inter 4.Low
3	Set position to 1	Set dial to 3 so that speed is set to 12	Set position to 3 so that speed increases to 60	Set position to 2 so that speed decreases to 30	1.Inter 1.Inter 2.Inter 3.High 4.Low
4	Set position to 1	Set dial to 2 so that speed is set to 6	Set position to 1	Set position to 2 so that speed decreases to 30	1.Inter 1.Inter 2.Inter 3.Inter 3.Inter 4.Low
5	Set position to 1	Set dial to 1 so that speed is set to 4	Set position to 3 so that speed increases to 60	Set position to 2 so that speed decreases to 30	1.Inter 1.Inter 2.Inter 3.High 4.Low
6	~Set position to 1	Set position to 2 so that speed increases to 30	Set position to 1	~Set position to 2 so that speed decreases to 30	1.Inter 1.Inter 2.Low 3.Inter 3.Inter 4.Low
7	~Set position to 1	Set position to 0 so that speed decreases to 0	Set position to 3 so that speed increases to 60	~Set position to 2 so that speed decreases to 30	1.Inter 1.Inter 2.Off 3.High 4.Low
8	~Set position to 1	Set dial to 3 so that speed is set to 12	Set position to 1	~Set position to 2 so that speed decreases to 30	1.Inter 1.Inter 2.Inter 3.Inter 3.Inter 4.Low
9	~Set position to 1	Set dial to 2 so that speed is set to 6	Set position to 3 so that speed increases to 60	~Set position to 2 so that speed decreases to 30	1.Inter 1.Inter 2.Inter 3.High 4.Low
10	~Set position to 1	Set dial to 1 so that speed is set to 4	Set position to 1	~Set position to 2 so that speed decreases to 30	1.Inter 1.Inter 2.Inter 3.Inter 3.Inter 4.Low

