IJCSMS International Journal of Computer Science and Management Studies, Vol. 12, Issue 01, January 2012
ISSN (Online): 2231-5268
www.ijcsms.com

Specification Representation and Test Case Reduction by Analyzing the Interaction Patterns in System Model

Ashish Kumari[1], Dr. Rahul Rishi[2]

[1]M.Tech (Scholar), Computer Science & Engineering,
Shekhawati Engineering College, Jhunjhunu, Rajasthan
simplyaashi@gmail.com

[2]Professor, University Institute of Engineering & Technology,
Maharshi Dayanand University, Rohtak, Haryana
rahulrishi@rediffmail.com

Abstract

Extended Finite State Machine uses the formal description language to model the requirement specification of the system. The system models are frequently changed because of the specification changes. We can show the changes in specification by changing the model represented using finite state machine. To test the modified parts of the model the selective test generation techniques are used. However, the regression test suits still may be very large according to the size. In this paper, we have discussed the method which define the test suits reduction and the requirement specification that used for testing the main system after the modifications in the requirements and implementation. Extended finite state machine uses the state transition diagram for representing the requirement specification. It shows how system changes states and action and variable used during each transition. After that data dependency and control dependency are find out among the transitions of state transition diagram. After these dependencies we can find out the affecting and affected portion in the system introduced by the modification. The main condition is: "If two test cases generate same affecting and affected pattern, it means it is enough to implement only one test case rather than two." So using this approach we can substantially reduce the size of original test suite.

Keywords: *Interaction Patterns, EFSM dependencies, Data dependencies.*

1. Introduction

Regression testing is a necessary though expensive maintenance activity that attempts to validate modified software and ensure that modifications are not only correct but also have not inadvertently affected the software so that portions that used to work no longer work correctly. The simplest regression testing strategy, *retest all*, tends to rerun all of the test cases in the original test suite on a modified version, and is therefore very time-consuming and expensive. An alternative, *selective retest* chooses only those tests that are associated with the modified portions. In either case, it is necessary to generate some new tests to cover untested modified portions of the system [3]. In this paper, we present a novel approach of specification representation and EFSM based regression test reduction that uses l dependence analysis to reduce a given regression test suite. The approach automatically identifies the difference between the original and modified EFSM systems by identifying a set of elementary modifications [9]: elementary addition of a transition and elementary deletion of a transition. For each elementary modification, regression test reduction strategies that use EFSM dependence analysis are used to reduce the regression test suite by eliminating repetitive tests. Our initial experience shows that this approach may significantly reduce the size of regression test suites.

The paper is organized as following: Section 2 provides an overview of the Requirement specification representation using EFSM, Section 3 presents an approach of EFSM based regression test generation and implementation of the approach is also given, Section 4 introduces EFSM dependencies, and Section 5 presents an approach of EFSM based regression test suite reduction based on dependence analysis. In Conclusions future research is discussed.

2. Requirement-specification representation

In this section, we provide an overview of requirement-specification representation using EFSM models. EFSM is a very popular technique for modeling state-based systems like computer communications, industrial control systems, *etc*. EFSM consists of states (including an initial state and an exit state) and transitions between states. A transition is triggered by an event provided that the enabling condition is satisfied. When a transition is traversed, certain action(s) may be performed [1,10]. An action may manipulate variables, read input or produce output. An enabling condition is a Boolean

IJCSMS International Journal of Computer Science and Management Studies, Vol. 12, Issue 01, January 2012
ISSN (Online): 2231-5268
www.ijcsms.com

predicate that may use EFSM variables and must evaluate to TRUE in order for the transition to be taken. EFSM models are graphically represented as graphs where states are represented as nodes and transitions as directed arcs. The following elements are associated with each transition: an event, a condition, and a sequence of actions. Figure1 shows a graphical representation of an EFSM transition [1].

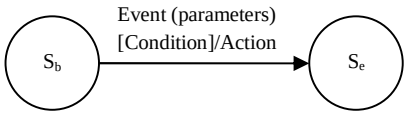

Figure 1 EFSM Transition

A simplified EFSM model of an ATM system is shown in Figure 2 [11]. This ATM system supports three types of transactions (withdrawal, deposit, and balance inquiry) represented by transitions. Before ATM transactions can be performed, user must enter a valid PIN that is matched against the PIN stored in the ATM card. A user is allowed a maximum of three attempts to enter the valid PIN. For example, the transition labeled T_4 is triggered when the system is in state S_1, event PIN is received, the value of parameter *p* of the event equals to variable *pin*. When the transition is triggered, the menu is displayed [10, 12].

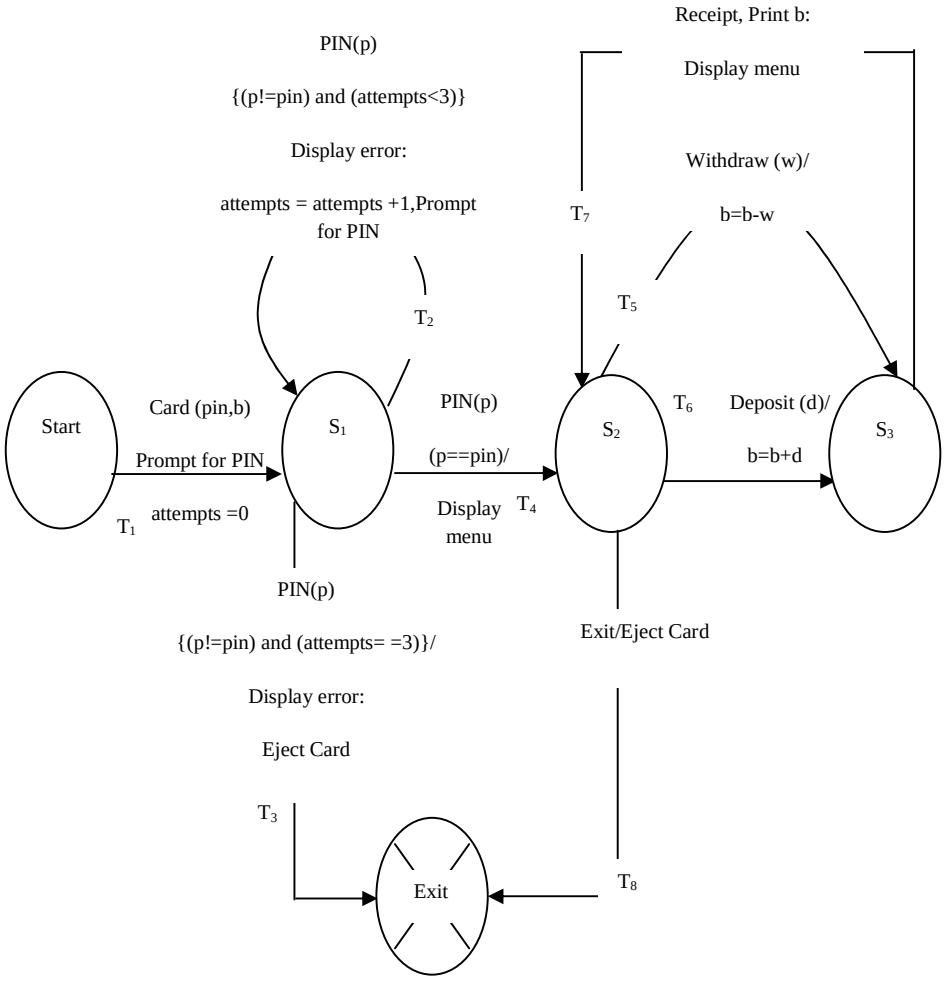

Figure 2 EFSM Model of the ATM System

We assume that this ATM system model was created from a set of individual system requirements. In this EFSM model, the requirement labeling is provided (either through an automated model generation or manually). Since individual requirements are, in most cases, represented as individual transitions in the

IJCSMS International Journal of Computer Science and Management Studies, Vol. 12, Issue 01, January 2012
ISSN (Online): 2231-5268
www.ijcsms.com

EFSM model, in this paper, we use transition and individual requirement interchangeably.

3. Requirement specification based test case generation

An EFSM system model becomes an input to EFSM test generator. The generator may support a variety of the existing EFSM model-based test generation strategies [8, 20]: state coverage, path coverage, constrained path coverage, *etc.* Depending on the selected testing strategy, the test generator automatically generates a set of EFSM paths (from an initial state to the final state) that satisfies the selected strategy. For each path, appropriate test values (inputs) that lead to the traversal of the selected path are identified. Clearly, a test case consists of a sequence of events (transitions) with appropriate input values. The following is an example of a test case for the ATM system shown in Figure 2:

> **Card (1234, 100.00); PIN (1234); Withdrawal (60); Continue; Deposit (70); Continue; Exit.**

In this paper, we concentrate mainly on generating tests as sequences of transitions (events) and we do not consider input values. Therefore, the test case shown above is represented as the following sequence of transitions: **T_1, T_4, T_6, T_8, T_5, T_8, T_9.**

Most of the existing EFSM model-based test generation strategies are mainly used to test the whole system, referred to as system testing. This type of testing is expensive because of a large number of generated test cases. In the earlier stages of a testing process, the frequently used type of testing is selective testing [1,11].

In selective testing, testers want to partially test the system with respect to a set of selected requirements (referred to as requirement-based selective testing). This type of testing is used to test selected system functionality represented by a requirement(s). In requirement-based selective testing, requirement information is first mapped to corresponding transitions of the EFSM model [10]. A tester selects a requirement(s) that should be tested. Based on this information, the test generator determines which transition(s) of the EFSM model corresponds to the selected individual requirement(s). Several selective testing strategies may be used: state coverage, transition coverage, path coverage, and constrained path coverage.

For example, consider the EFSM system model shown in Figure 2. A tester chooses to generate a system test suite using a system constrained path coverage testing strategy. This strategy requires that every path in the model be traversed at least once where each path can contain at most *n* "occurrences" of the same transition (any transition can be traversed at most *n* times in a path). The resulting system test suite contains 64 tests for *n* = 3; and 160 tests for *n* = 4.

3.1 Algorithm for test case generation:

The algorithm define below based on depth first search and gives details to find out the possible path or test cases from a finite state machine represented using state transition diagram. The algorithm path generate is invoked on the start state of the finite state machine.

Transition of the ATM state transition diagram contains the following details:

```
Struct transition

{

int  no;

int  source;

int  dest;

int  no_of_variable_used;

char  *action[no_of_action];

char  *events[no_of _events];

char  *var[no_of _var_used];

int accurance;

}
```

Path_generation(struct transition T1)

{

If (T1.occurance<n)

> **Insert this particular transition T1 into the stack.**
>
> **T1.occurance+=1;** /* Transition would be traversed up to n times in path when there is cycle to avoid infinite no. of possible test cases. */

If (T1.dest==exit_state)

> **Display the contents of the stack array from 0 to top of the stack.**

IJCSMS International Journal of Computer Science and Management Studies, Vol. 12, Issue 01, January 2012
ISSN (Online): 2231-5268
www.ijcsms.com

/* which consist of the sequence of transition traversed in executing this particular path or test case.*/

Else

Repeat the step for all adjacent transition T to transition T1.

/* adjacent transitions are those whose source state is same as destination state of T1.*/

Path generation (T)

/*call path generation for the next adjacent transition to T1 */

Pop (); /* pop out the last transition from the stack. This algorithm is based on depth first search so after finding out the one path the. We backtrack to second last node of the path and find out another node that is adjacent. */

}

4. EFSM dependencies

Before we present the approach of regression test suite reduction using EFSM dependence analysis, we introduce dependencies that may exist in the EFSM model. We define two types of dependencies between transitions ("active" elements of an EFSM model): data dependence and control dependence. Note that states are "passive" elements of the EFSM model. These dependencies capture the notion of potential "interactions" between transitions in the model. Let T be a transition. The following notation related to transition T is introduced [1, 11]:

Sh(T) is a state from which T is outgoing. So(T) is a state to which T is incoming. .U (T) is a set of variables used in transition T. i.e. .variables used in a condition or an action of T. D (T) is a set of variables defined by transition T .i.e.. Variables defined by an action or defined in an event of T and not redefined by the action of T. C(T) is an enabling condition associated with transition T. E(T) is an event associated with transition T.

Data dependence

Data dependence captures the notion that one transition defines a value to a variable and another transition may potentially use this value. More formally, there exists data dependence between transitions Ti and Tk if there exists a variable v such that: (I) v E D(Ti), (2) v E U(Tk), and (3) there exists a path (transition sequence) in the EFSM model from Ti to Tk along which v is not modified; such a path is referred to as definition-clear path. For example, there exists data dependence between transitions T I and T 5 because transition T I assigns a value to variable b (in the event Card (pin, b), transition Ts uses variable b (in action "b = b -w"), and there exists a path (sequence of transitions TI, T4, T5) from TI to T5 along which b is not modified.

Control dependence

Control dependence captures the notion that one node in the control graph may affect the execution of other node. Let Y and Z be two states (nodes) and T be an outgoing transition (edge) from Y. State Z post-dominates state Y iff Z is on every path from Y to the exit state. State Z post dominates transition T iff Z is on every path from Y to the exit state through transition T. Transition Ti has control dependence on transition T k. (transition T k is control dependent on transition Ti) iff (1) state Sh(Tk) does not post-dominate state Sh(T;), and (2) state Sh(Tk) post-dominates transition Ti. For example, transition T4 has control dependence on transition Ts in the EFSM model of Figure 2 because state S2 does not post dominate state SI and state S2 post-dominates transition T4.

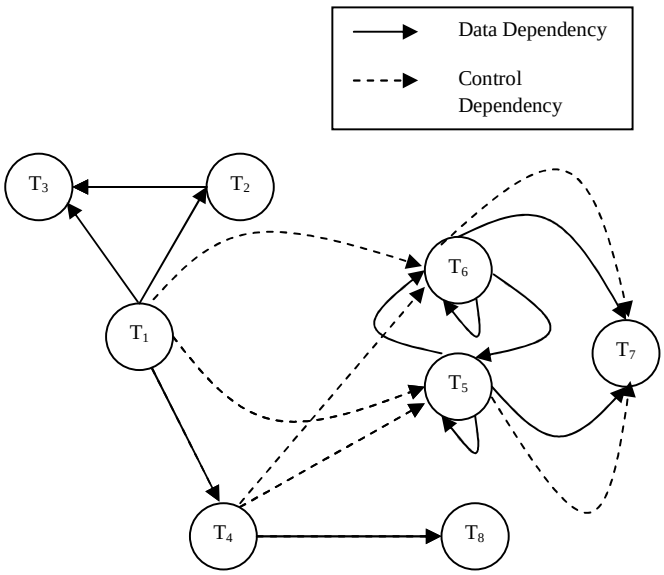

Figure 3. ATM EFSM Dependencies graph.

Graph shows the data and control dependencies among the transition of the ATM model. Node represents transitions and edges represent the dependencies among these transitions. For example

IJCSMS International Journal of Computer Science and Management Studies, Vol. 12, Issue 01, January 2012
ISSN (Online): 2231-5268
www.ijcsms.com

there are data dependencies between T1 and T6 for data variable balance because balance is initialized in T1 and it is redefined in T6 and in b/w these there is no modification in the value of the balance variable. In graph I have not shown the dependencies on the particular variable.

In this paper, we present a EFSM based regression test suite reduction approach that uses EFSM graph dependence analysis to reduce regression test suites. The approach accepts as inputs: the original model, the elementary modification, and a given regression test suite. Now based on this elementary modification, modified EFSM graph is created and the modifications are marked using some special symbol.

5. Regression test suite reduction

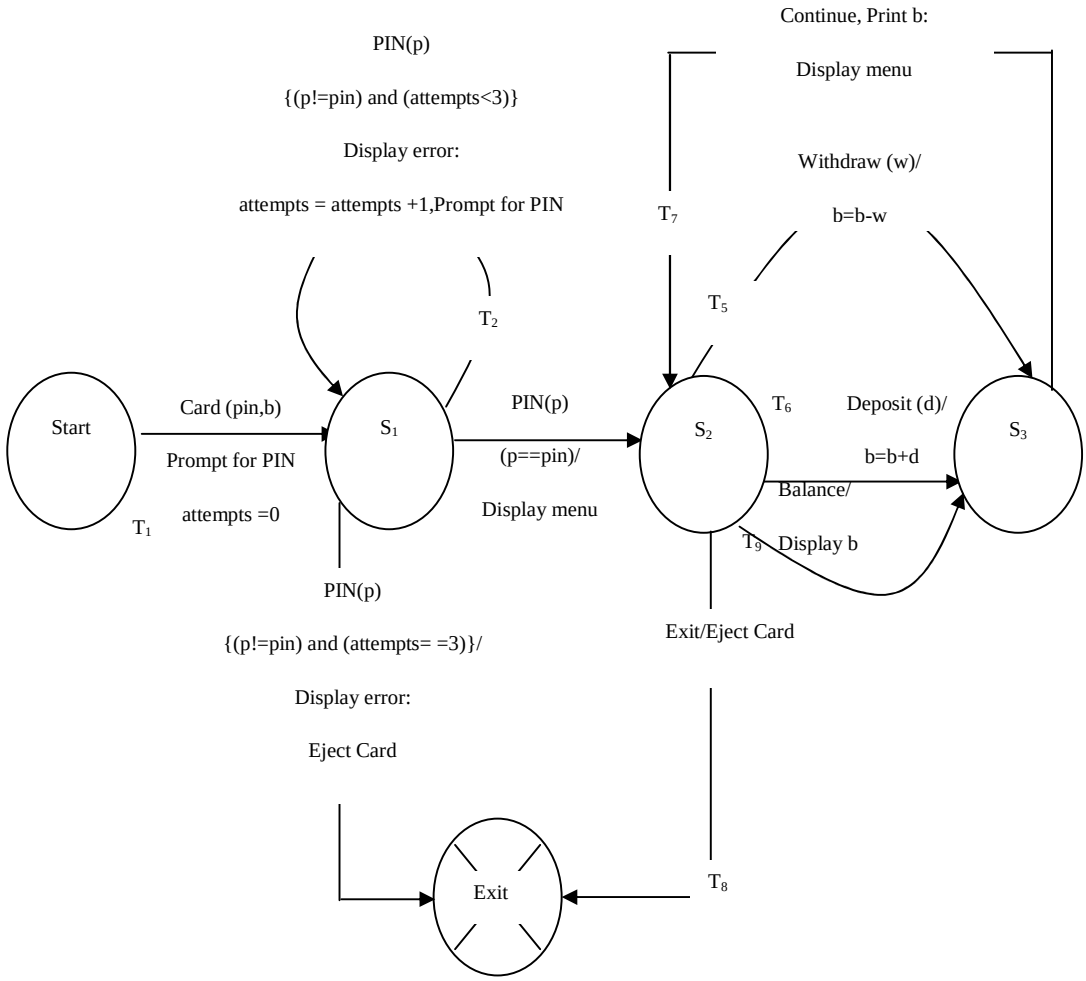

Figure 4 Modified ATM EFSM

Modified ATM EFSM contain on more transition T9 which shows the balance. So regression test suite reduction is achieved based on two things.

5.1 Selective test reduction: only those test cases are included in the selective test suite which traverses the modification during the traversing of the selected test case. Test cases which do not traverse the added transition means they are already tested and no need to be retested. For example Test case T1T2T2T2T3 does not include any new transition so there is no need to retest this case and this is not included in the selective test case suite.

5.2 Test case reduction using dependence analysis: In this selective test suite would be the input for this method. Here for every test case we find affecting and affected interaction pattern.

I Affecting interaction pattern: This graph contain those transitions which affect the newly added transition.

IJCSMS International Journal of Computer Science and Management Studies, Vol. 12, Issue 01, January 2012
ISSN (Online): 2231-5268
www.ijcsms.com

Following steps should be performed for finding out affecting interaction pattern based on the control and data dependencies.

1. We traverse each test case in the selective test suit and create a graph of data dependencies and control dependencies which occur during the traversal of that particular test case.
2. Now this graph is traversed in the backward direction from the added transition and creates a new graph which includes only the traversed transition. Now this graph is the interaction pattern for that particular test case.

II Affected interaction pattern: This graph contains that transition which may be affected by the newly added transition.

Following steps should be performed for finding out the affected interaction pattern based on the control and data dependencies.

1. We traverse each test case in the selective test suit and create a graph of data dependencies and control dependencies which occur during the traversal of that particular test case.
2. Now this graph is traversed in the forward direction from the added transition and creates a new graph which includes only the traversed transition. Now this graph is the interaction pattern for that particular test case.

Now those test cases whose both affecting and affected interaction pattern is same, we can emit one of the test cases in retesting. So only one of the test cases would be included in the test reduction test suite.

For example:

Test case#1. T1 T2 T2 T4 T5 T7 T6 T7 T9 T7 T8.

Test case#2 T1 T2 T4 T5 T7 T6 T7 T9 T7 T8.

Now the interaction pattern for the both the test cases are same. So we include only one of them in reduction test suite.

6. Conclusion

This paper proposed a approach using which we can represent the requirement specification and algorithm which is used to generate the possible test cases and a test reduction approach which reduce the test case suite based on the dependence analysis among the transition. Data dependencies and control dependencies are described in a very simple manner in the paper which forms the basis for test case reduction and generating the interaction pattern. Implementation of the said approach is under development in C language. In future I try to find the approach which can also consider the side effect of the added transition and to investigate the present approach on some other state based system and analyze the efficiency of the said approach.

7. References

1. Korel, B., Tahat, L.H., and Vaysburg, B., “Model-based regression test reduction using dependence analysis”, In *Proc. of ICSM’02* (Montréal, Canada, October 3-6, 2002). IEEE Computer Society Press, Washington, DC, 2002, 214-223.
2. Tahat, L., Vaysburg, B., Korel, B., Bader, A.,"Requirement-Based Automated Black-Box Test Generation," Proceedings of the 25th Annual IEEE International Computer Software and Applications Conference (COMPSAC), Chicago, IL, pp. 489-495, 2001.
3. Lihua Xu and debra Richardson,” Generating regression tests using model checking” Computer Software and Applications Conference,pp. 336 - 341 vol.1 , 2004
4. Rothermel, G., Harrold, M., "A Safe, Efficient Regression Test Selection Technique," ACM Transactions on Software Engineering and Methodology, 6(2), pp. 173-210,1997.
5. Savage, P., Waiters, S., Stephenson, M., "Automated Test Methodology for Operational Flight Programs," Proceedings of IEEE Aerospace Conference, vol. 4, pp. 293-305, 1997.
6. Sherlund, B., Korel, B., Modification Oriented Software Testing," Proceedings of Quality Week, pp. 1-17, 1991.
7. Tahat, L., Vaysburg, B., Korel, B., Bader, A., "Requirement-Based Automated Black-Box Test Generation," Proceedings of the 25th Annual IEEE International Computer Software and Applications Conference (COMPSAC), Chicago, IL, pp. 489-495, 2001.
8. Tsai, W., Bai, X., Paul, R., Yu, L., "Scenario-Based Functional Regression Testing," Proceedings of the 25'th Annual IEEE International Computer Software and Applications Conference (COMPSAC), Chicago, IL, pp. 496-501, 2001.

IJCSMS International Journal of Computer Science and Management Studies, Vol. 12, Issue 01, January 2012
ISSN (Online): 2231-5268
www.ijcsms.com

9. Vaysburg, B., Tahat, L., Korel, B., Bader, A., " Automating Test Case Generation from SDL Specifications," Proceedings of the 18th International Conference on Testing Computer Software (TCS),
10. Vaysburg, B., Tahat, L., Korel, B., "Dependence Analysis in Reduction of Requirement Based Test Suites," to appear in Proceedings of IEEE international Symposium on Software Testing and Analysis (ISSTA), Rome, Italy,2002.
11. Ynaping Chen, Robert L. Probert and Hasan Ural , “Regression Test Reduction Using Extended Dependence Analysis” in SOQUA’07 , September 3-4 2007, ACM Transaction , Dubrovnik , Croatia , 2007.
12. Luay Tahat, Bogdan Korel, Mark Harman, Hasan Ural,” Regression test suite prioritization using system models”, 2011.