Volume 8, No. 5, May-June 2017
ISSN No. 0976-5697
International Journal of Advanced Research in Computer Science
RESEARCH PAPER
Available Online at www.ijarcs.info

Functional Test Case Generation based on Model Driven Testing using FSM and UML Activity Diagram

Supriya S. Patil
Research Scholar
Bharati Vidyapeeth Deemed University
College of Engineering,
Pune, India

Prof. Pramod A Jadhav
Information Technology
Bharati Vidyapeeth Deemed University
College of Engineering,
Pune, India

Prof. Dr. S D Joshi
Computer Dept.
Bharati Vidyapeeth Deemed University College of Engineering,
Pune, India

Abstract: Model Based Testing is one of the most critical area to be addressed efficiently to ensure effective testing of the given project. The system implemented combines UML with FSM to cover all scenarios with all possible paths. As Finite Machine also works on the trigger where conditions are framed and if these conditions are satisfied then next action is executed; this phenomenon motivates to build a framework for generating the test cases automatically covering all paths (activity diagram in UML helps to cover all paths) and conditions (Finite State Machine helps to frame set of conditions). Model Based Testing designed considering all paths and conditions to check all scenarios to generate detailed test cases for given project or application.

Keywords: Model Based Testing, Extended File System, Finite State Machine,Activity Diagram, Coverage of paths and conditions

1. INTRODUCTION

Software Testing is indivisible phase of life cycle used for the development of a software. Software Testing field is advancing day by day. We can see recent frameworks like TestNG, works for data driven testing where application processes comparatively large amount of data. Selenium tool is one of the popular tool now a day; Selenium Web Driver works for test cases and functionalities of all browsers, Selenium Grid works on distributed Computing, Selenium RC works on remote control. QTP is windows based testing tool used to check applications on Microsoft OS (windows). Industry standards demand customized agile approach. All this motivates for the approach where every unit can be tested for the functionality as per the specifications. As automation in the field of electronics, mechanical and civil and bio informatics has increased enormously. This demands quality check of software in all these fields mentioned above.

Unit and functionality testing with all scenario is the crux of software testing phase. The quality of the software depends upon the rigorous testing and the way adopted for performing the software testing. This type of quality testing can be performed in coordination with UML diagram which may explore different dimensions and functionality of a software, wise selection and tuning of the test cases with reference to the inputs collected from these diagrams will lead to the quality product.

2. LITERATURE SURVEY

Number of research papers analyzed to select the papers focusing on Model Based Testing. Few important approaches are discussed here to decide the problem statement and guideline for the approach to solve the existing problems.

Andr´e Takeshi Endo et. al. [1] The approach combine Model Based Testing and structural testing for a web services. Technique used is based on the events, known as ESG4WS. Structural testing [10] [11] [12] [14] is used to meet the quality of software intended in software requirement document. This helps to stop the process of testing after getting the satisfactory results. Event Sequence Graph of the application to be tested is plotted to understand the functioning of the software in detail, especially coverage and scope of application becomes clear. Authors have focused on the data flow and control flow. Control flow [13] is used analyze coverage of all nodes and edges. Data flow [7]deals with the use and potential use. Limitation of the system could be enhancement for the detection of faults in the system to be tested.

Decision of the condition could be made by using or analyzing Finite State Machine (FSM) [2]. In this case at every stage condition is checked and the further path to be traversed is calculated with this decision. Rules are designed for the effective functioning of the utility and it is forced on the traversal to make sure that system will check [6][9] all possible condition in the form of Boolean values. Limitation of this system is there are only two options for the condition because of Boolean values. This may not work, if we want to process the data where there are more than two conditions.

Finite State Machine (FSM) works on multiple conditions at every node and based on the desired values of the results of every parameter the further path to be traversed is decided.

The process of FSM could be categorized in three parts as

- E-block- To evaluate trigger for all conditions.
- FSM-block – To compute state next to current state &a signals, which controls A-block.
- A-block - To perform the required data operations and movements of a data.[5]

There is scope for the improvement as system is based on the web services and it supports only the utilities designed using java[3].
Metamodel Transformation [4] proposed there are five transformations are mentioned about five different metamodels. Metamodel gives information about the functionality, especially first two metamodels describe functional requirement and the third one specifies the test scenarios to be tested and fourth deals with the values associated. Fifth metamodel combines,[7][8] all inputs into segregated test cases format.
After analyzing all these approaches a system with coverage of all paths and capable to check all conditions at every node rigorously can be a challenge to be addressed.

3. MATHEMATICAL MODELLING

Input: FSM, Activity diagram in the form of XML
Process FSM⊕ ACTIVITY
Output : test cases with all path, prioritization and removal of redundancy

Data Structure:

Serial Number	Variable	Meaning
1.	F	Functionalities
2.	S	States
3.	C	Conditions
4.	E	Edges
5.	N	Nodes
6.	T	test cases
7.	S	serial number
8.	Se	Sender
9.	Re	Receiver

```
SN- 0
For (i=0; i<=#F;i++)
{
        For (j=0; j<=#C;j++)
        {
                For (k=0; k<=#E;k<=#N;k++)
                {
                        T – (SN, condition, Cs, Se, Re)
                                SN++; T++;
                        Display T;
                }

        }

}
```

4 . SYSTEM ARCHITECTURE

4.1 Modules

1.Input Activity Diagram:

The proposed work accepts activity diagram as input. Every activity diagram are familiar for creating its ADT technically, that means to have all required details which will modify the model to look at capabilities and functionalities of all activity diagrams. The ADT can then create the ADG technically. ADG are access by using DFS for all necessary available test methods. Thus, the entire main points are added in every checked path using the ADT to have the ultimate test cases. Every activity diagram should be go through each of the four modules for making top collection of extremely economical test cases.

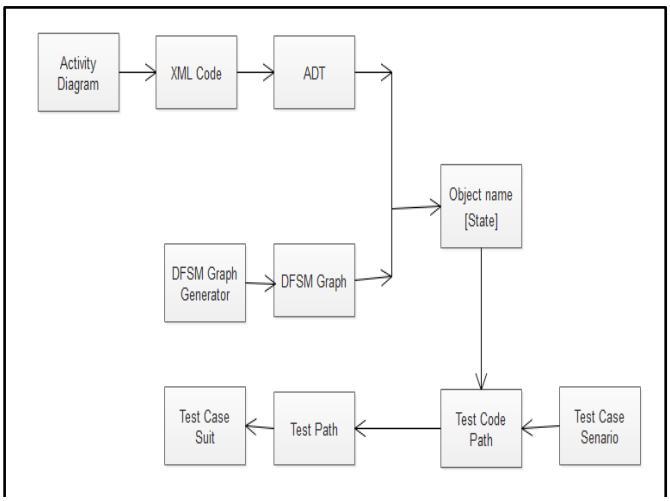

Figure 1: System Architecture of MBT using ADT and FSM

2. ADT Generation

ADT (Activity Dependency Table) and loops, synchronization and methods showing the activities of the task are created technically utilizing each activity diagram. This plan to indicating the activities will move to different entities which may be supportive for system, integration and regression testing. In addition, it contains the input with the desired value of output for every activity of the system. Activity Dependency Table shows every activity dependency on each other very clearly. Every activity has its special symbol for easily referencing it within determinant dependencies also using it within various concerned units of the system. To reduce the searches of the created ADG (that are going to be explain afterward during this section), activity which are permanent are distributed within one image exclusively rather than having many symbols for a related activity

3. DFSM Graph Generator

DFSM generator deals with the phenomenon where one single output is produced, this module elaborates the details of criteria used for controlling the criterion that are used to control generation of test cases

4. Test Suit Generation

Test suit generation covers various path to be evaluated under testing, it comprises following types of testing the coverage

- Round Trip – Total round trips in the path are covered and reported
- Sequence – Total number of sequences of inputs are covered in this sequence.
- Action – All actions are to be visited at least once in the path.

- Event – All events are to be visited in the path of the test cases.
- State – All states should be dealt at least once in a life cycle.
- Transition- This deals with the total number of transitions present in the coverage.

5. IMPLEMENTATION AND RESULT

Experiment is performed by considering the example of PIN change functionality of ATM. Experiment is performed by using individual approach where path is calculated using Activity diagram and FSM separately (Figure 2: Individual Approach); in combined approach (Figure 3: Combined Approach) results are combined then redundancy is removed before generating the test case. The generated test cases are again checked for duplication. Final outcome of the MBT using combined approach is to generate test cases with more test cases as we can see that number is 83 and removal of unnecessary, redundant test cases.

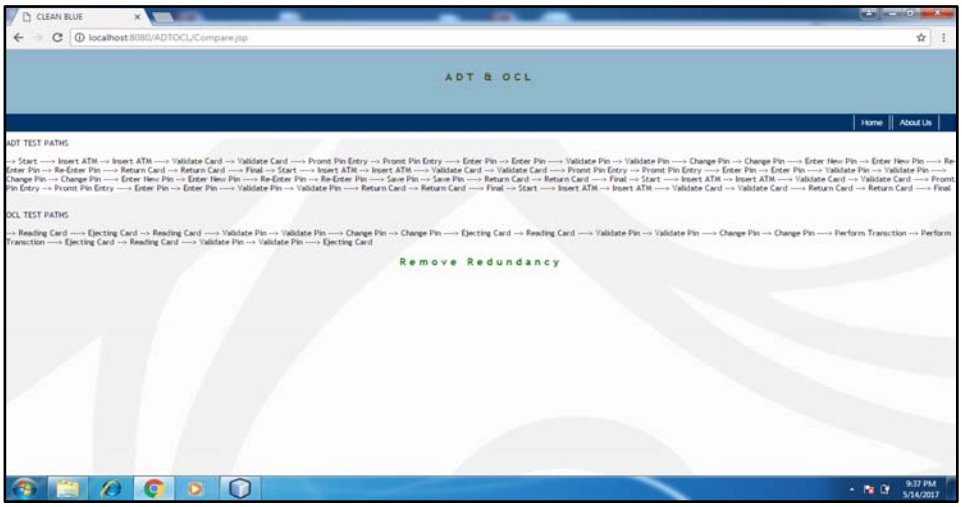

Figure 2: Individual Approach

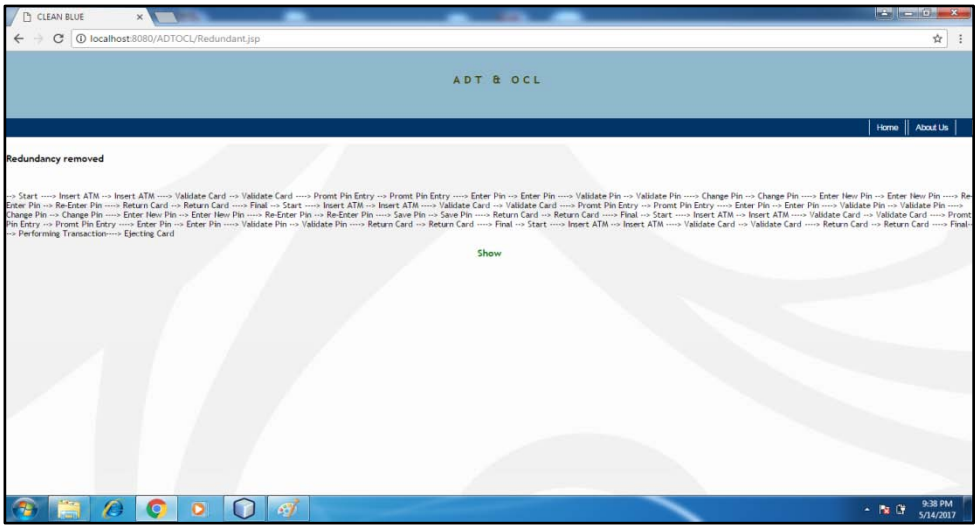

Figure 3: Combined Approach

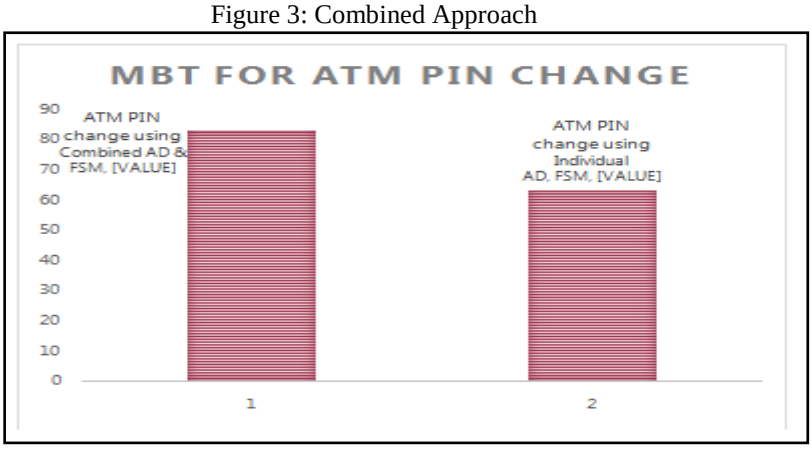

Figure 4: MBT for ATM PIN change

6. CONCLUSION

The system is capable to deliver the satisfactory result as we can see number of steps needed to check all conditions and cover all path are reduced considerably from 83 to 63, because of removing the redundancy; Because of it time complexity is improved significantly.
Another important contribution in the system is to improve the number of test cases to test each and every minute functionality; while doing so redundancy in test cases is removed to remove unnecessary functionality testing.

5. REFERENCES

[1] Aritra Bandyopadhyay, Sudipto Ghosh, "Test Input Generation using UML Sequence and State Machines Models"

[2] Vikas Panthi, Durga Prasad Mohapatra, "Automatic Test Case Generation using Sequence Diagram", International Journal of Applied Information Systems (IJAIS) – ISSN : 2249-0868 Foundation of Computer Science FCS, New York, USA Volume 2– No.4, May 2012 – www.ijais.org

[3] Md Azaharuddin Ali et.al. "Test Case Generation using UML State Diagram and OCL Expression", International Journal of Computer Applications (0975 – 8887) Volume 95– No. 12, June 2014

[4] S. ShanmugaPriya et.al, " Test Path Generation Using UML Sequence Diagram", Volume 3, Issue 4, April 2013 ISSN: 2277 128X International Journal of Advanced Research in Computer Science and Software Engineering

[5] Ching-Seh Wu , Chi-Hsin Huang," The Web Services Composition Testing Based onExtended Finite State Machine and UML Model", 2013 Fifth International Conference on service Science and Innovation

[6] M. Benjamin, D. Geist, A. Hartman, Y. Wolfsthal, G. Mas and R. Smeets, "A study in coverage-driven test generation", In Proc. of the 36 th Conference on Design Automation Conference, pp. 970-975, 1999.

[7] M. Born, I. Schieferdecker, H.-G. Gross, and P. Santos. "Model-Driven Development and Testing – A Case Study". In Proc. of the 1st European Workshop on Model Driven Architecture with Emphasis on Industrial Application, pp. 97-104, 2004

[8] F. Bouquet, C. Grandpierre, B. Legeard, and F. Peureux, "A Test Generation Solution to Automate Software Testing", In Proc. of the 3rd international workshop on Automation of software test, pp. 45-48, 2008.

[9] F. Bouquet, C. Grandpierre, B. Legeard, F. Peureux, N. Vacelet, and M. Utting, "A subset of precise UML for Model-based Testing", In Proc. of the 3rd International Workshop Advances in Model Based Testing (AMOST), pp. 95-104, 2007.

[10] Q. Farooq, M. Z. Z. Iqbal, Z. I. Malik and A. Nadeem, "An approach for selective state machine based regression testing", In Proc. of 3rd International Workshop Advances in Model Based Testing (AMOST), pp. 44-52, 2007.

[11] C. Crichton, A. Cavarra, and J. Davies, "Using UML for Automatic Test Generation", In Proc. of the Automation of Software Testing, 2007.

[12] S. R. Ganov, C. Killmar, S. Khurshid, and D. E. Perry. "Test Generation for Graphical User Interfaces Based on Symbolic Execution". In Proc. Proc. of the 3rd International Workshop on Automation of Software Test, pp. 33-40, 2008.

[13] H. Garavel, F. Lang, R. Mateescu, and W. Serwe, "CADP 2006: A Toolbox for the Construction and Analysis of Distributed Processes", In Proc. of the 19th International Conference on Computer Aided Verification, pp. 158-163, 2007.

[14] J. R. Calame, "Specification-based Test Generation with TGV", Technical Report SEN-R0508, Centrum voor WiskundeenInformatica, 2005.