A Result of Model Based Testing using Predictive Range Framework

[1]Abhijit Kadam,[2]Mrs. Bhagyashree Dhakulkar
[1,2]Department of Computer Engineering,
[1,2]Dr. D. Y. PatilSchool of Engineering and Technology, Pune

***Abstract*:**Model based mostly Testing is one amongst the foremost crucial areas to be addressed with efficiency to make sure effective testing of the given project. The system enforced combines UML with FSM to hide all situations with all doable methods. As Finite Machine additionally works on the trigger wherever conditions are framed and if these conditions are glad then next action is executed; this development motivates to create a framework for generating the take a look at cases mechanically covering all methods (activity diagram in UML helps to hide all paths) and conditions (Finite State Machine helps to border set of conditions). Model based mostly Testing designed considering all methods and conditions to ascertain all situations to come up with elaborated take a look at cases for given project or application.

***IndexTerms*:** Model Based Testing, Extended File System, Finite State machine, Activity Diagram, Coverage of paths and conditions.

I. INTRODUCTION

Software Testing is undividable section of life cycle used for the event of code. Code testing field is advancing day by day. We will see recent frameworks like check weight unit, works for knowledge driven testing wherever application processes relatively great deal of information. chemical element tool is one in every of the popular tool currently a day; chemical element internet Driver works for check cases and functionalities of all browsers, chemical element Grid works on distributed Computing, chemical element RC works on device. QTP is windows based mostly testing tool wont to check applications on Microsoft OS (windows). Trade standards demand bespoke agile approach. All this motivates for the approach wherever each unit are often tested for the practicality as per the specifications. As automation within the field of physical science, mechanical and civil and bio information processing has hyperbolic hugely. This demands quality check of code altogether these fields mentioned on top of.

Unit and practicality testing with all eventualities is that the crux of code testing section. The standard of the code depends upon the rigorous testing and therefore the approach adopted for acting the code testing. this kind of quality testing are often performed in coordination with UML diagram which can explore completely different dimensions and practicality of code, wise choice and standardization of the check cases with respect to the inputs collected from these diagrams can result in the standard product.

II. LITERATURE REVIEW

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 combines 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 (EFSM) 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 & 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 metamodel. Metamodel gives information about the functionality, especially first two metamodel 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.

III. MATHEMATICAL MODELLING

Input: EFSM, Sequence diagram in the form of XML
Process EFSM⊕ SEQUENCE
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;
                }

        }

}
```

IV. SYSTEM ARCHITECTURE

4.1 Modules

1. Input Activity Diagram:

The planned work accepts activity diagram as input. each activity diagram square measure acquainted for making its ADT technically, meaning to possess all needed details which is able to modify the model to seem at capabilities and functionalities of all activity diagrams. The ADT will then produce the ADG technically. ADG square measure access by victimization DFS for all necessary accessible take a look at ways. Thus, the complete small print square measure additional in each checked path victimization the ADT to possess the last word takes a look at cases. Each activity diagram ought to be probing every of the four modules for creating high assortment of very economical take a look at cases.

2. ADT Generation

ADT (Activity Dependency Table) and loops, synchronization and ways showing the activities of the task square measure created technically utilizing every activity diagram. This attempt to indicating the activities can move to totally different entities which can be ancillary for system, integration and regression testing. Additionally, it contains the input with the specified worth of output for each activity of the system. Activity Dependency Table shows each activity dependency on one another terribly clearly. Each activity has its special image for simply referencing it among determinant dependencies additionally victimization it among numerous involved units of the system. To scale back the searches of the created ADG (that square measure attending to be make a case for subsequently throughout this section), activity that square measure permanent square measure distributed among one image completely instead of having several symbols for a connected activity

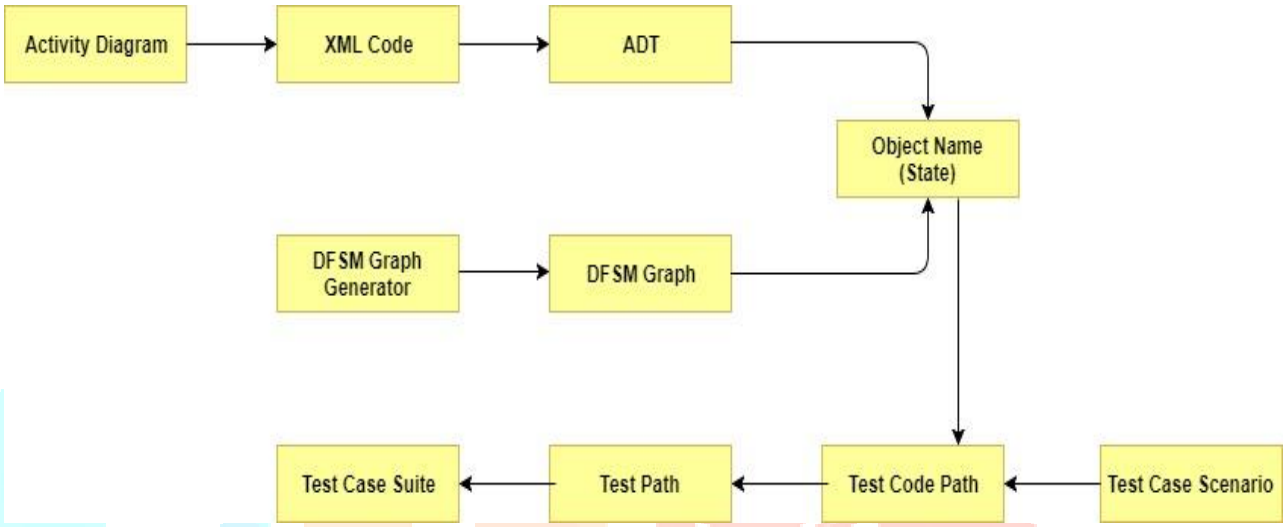


Figure 1: System Architecture of MBT using ADT and FSM

3. DFSM Graph Generator

DFSM generator deals with the development wherever one single output is created, this module elaborates the main points of criteria used for dominant the criterion that square measure wont to management generation of take a look at cases.

4. Action at law Generation

Test suit generation covers numerous paths to be evaluated below testing, it contains following sorts of testing the coverage:

- Round Trip – Total spherical journeys within the path square measure coated and reportable
- Sequence – Total variety of sequences of inputs square measure coated during this sequence.
- Action – All actions square measure to be visited a minimum of once within the path.
- Event – All events square measure to be visited within the path of the take a look at cases.
- State – All states ought to be dealt a minimum of once in a very life cycle.
- Transition- This deals with the whole variety of transitions gift within the coverage.

V. 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.

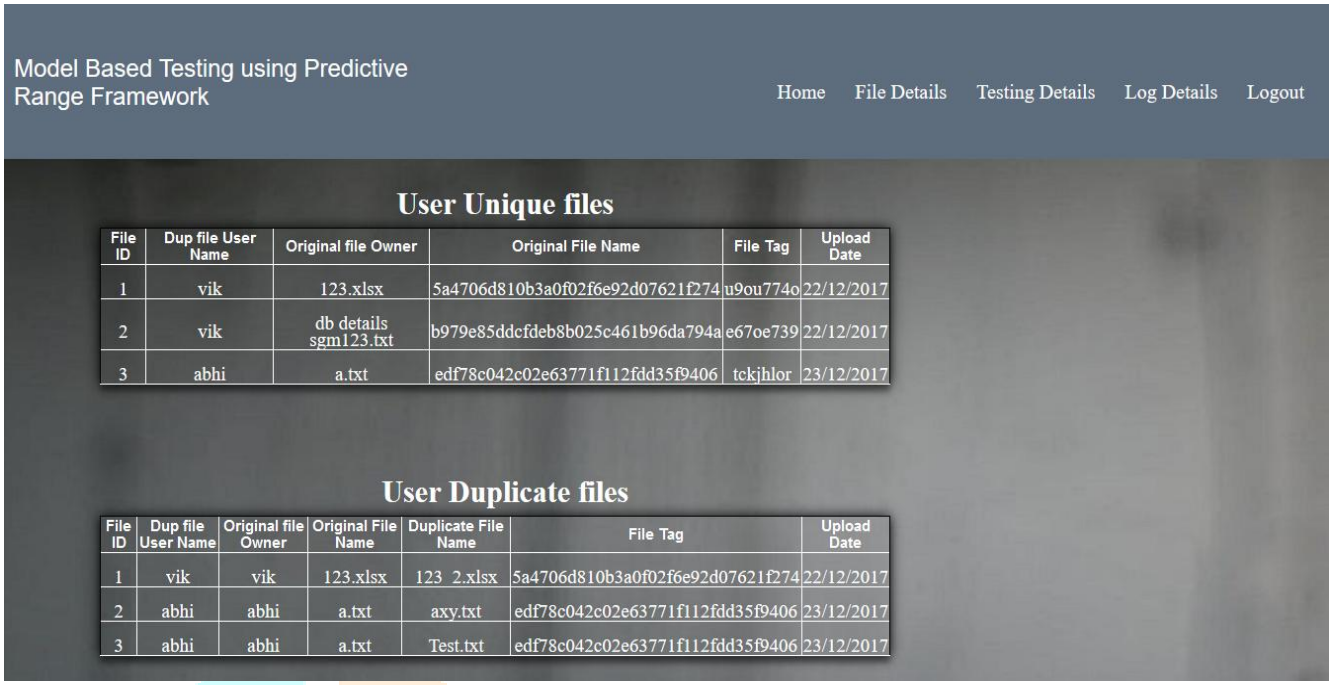


Figure 2: Individual Approach

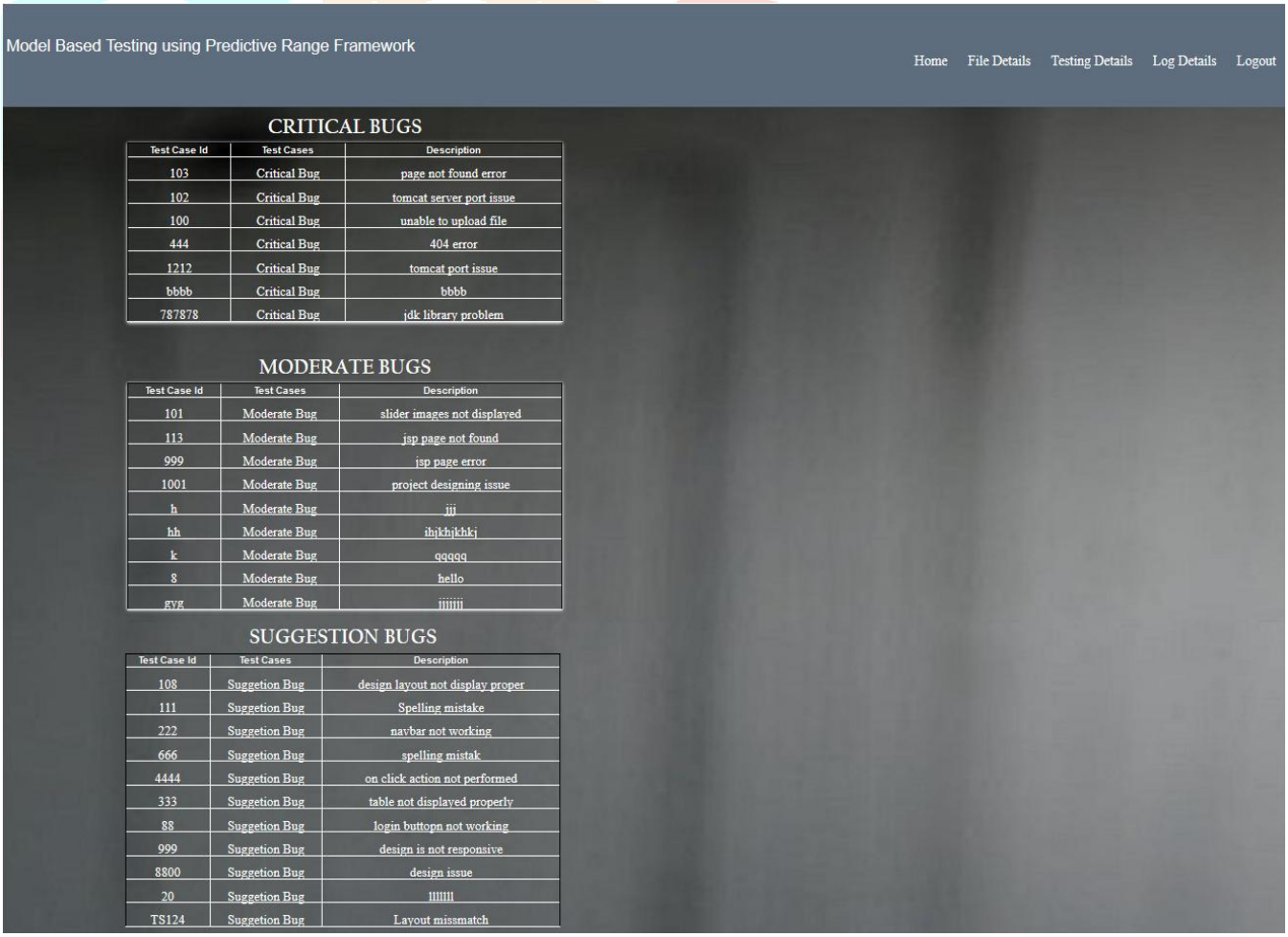
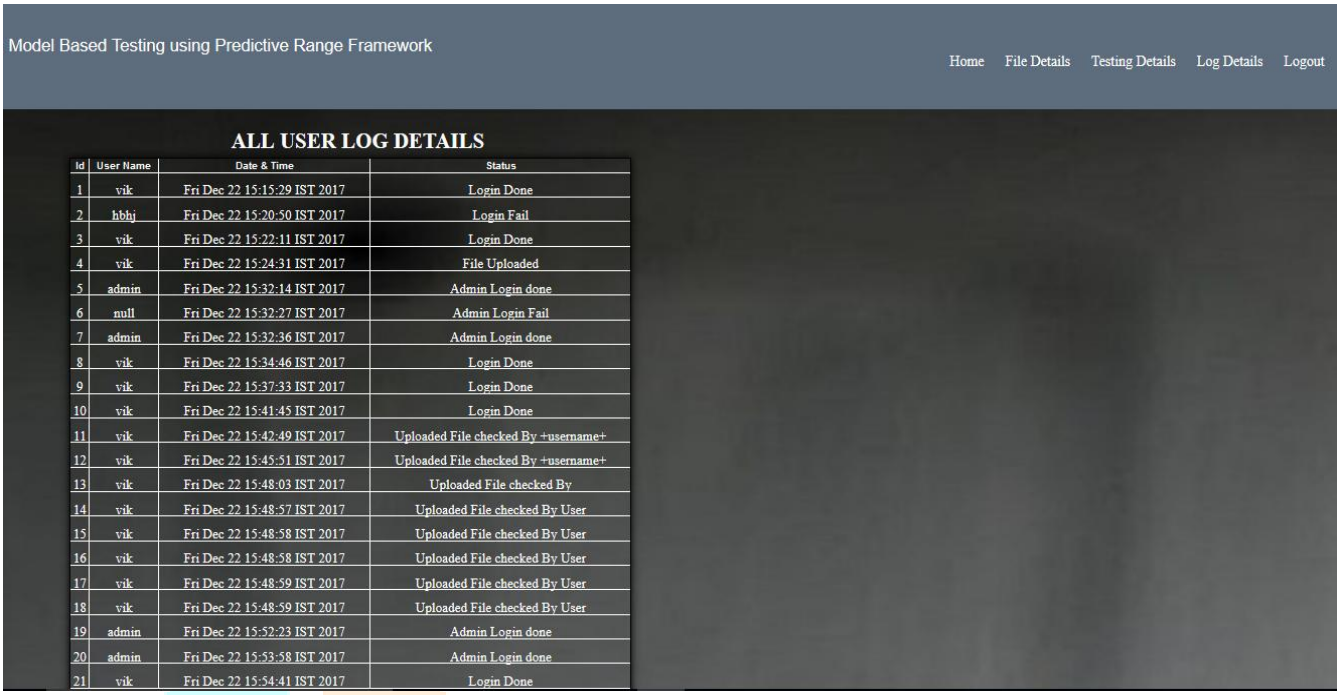


Figure 3: Combined Approach



Id	User Name	Date & Time	Status
1	vik	Fri Dec 22 15:15:29 IST 2017	Login Done
2	hbhj	Fri Dec 22 15:20:50 IST 2017	Login Fail
3	vik	Fri Dec 22 15:22:11 IST 2017	Login Done
4	vik	Fri Dec 22 15:24:31 IST 2017	File Uploaded
5	admin	Fri Dec 22 15:32:14 IST 2017	Admin Login done
6	null	Fri Dec 22 15:32:27 IST 2017	Admin Login Fail
7	admin	Fri Dec 22 15:32:36 IST 2017	Admin Login done
8	vik	Fri Dec 22 15:34:46 IST 2017	Login Done
9	vik	Fri Dec 22 15:37:33 IST 2017	Login Done
10	vik	Fri Dec 22 15:41:45 IST 2017	Login Done
11	vik	Fri Dec 22 15:42:49 IST 2017	Uploaded File checked By +username+
12	vik	Fri Dec 22 15:45:51 IST 2017	Uploaded File checked By +username+
13	vik	Fri Dec 22 15:48:03 IST 2017	Uploaded File checked By
14	vik	Fri Dec 22 15:48:57 IST 2017	Uploaded File checked By User
15	vik	Fri Dec 22 15:48:58 IST 2017	Uploaded File checked By User
16	vik	Fri Dec 22 15:48:58 IST 2017	Uploaded File checked By User
17	vik	Fri Dec 22 15:48:59 IST 2017	Uploaded File checked By User
18	vik	Fri Dec 22 15:48:59 IST 2017	Uploaded File checked By User
19	admin	Fri Dec 22 15:52:23 IST 2017	Admin Login done
20	admin	Fri Dec 22 15:53:58 IST 2017	Admin Login done
21	vik	Fri Dec 22 15:54:41 IST 2017	Login Done

Figure 4: MBT for log change

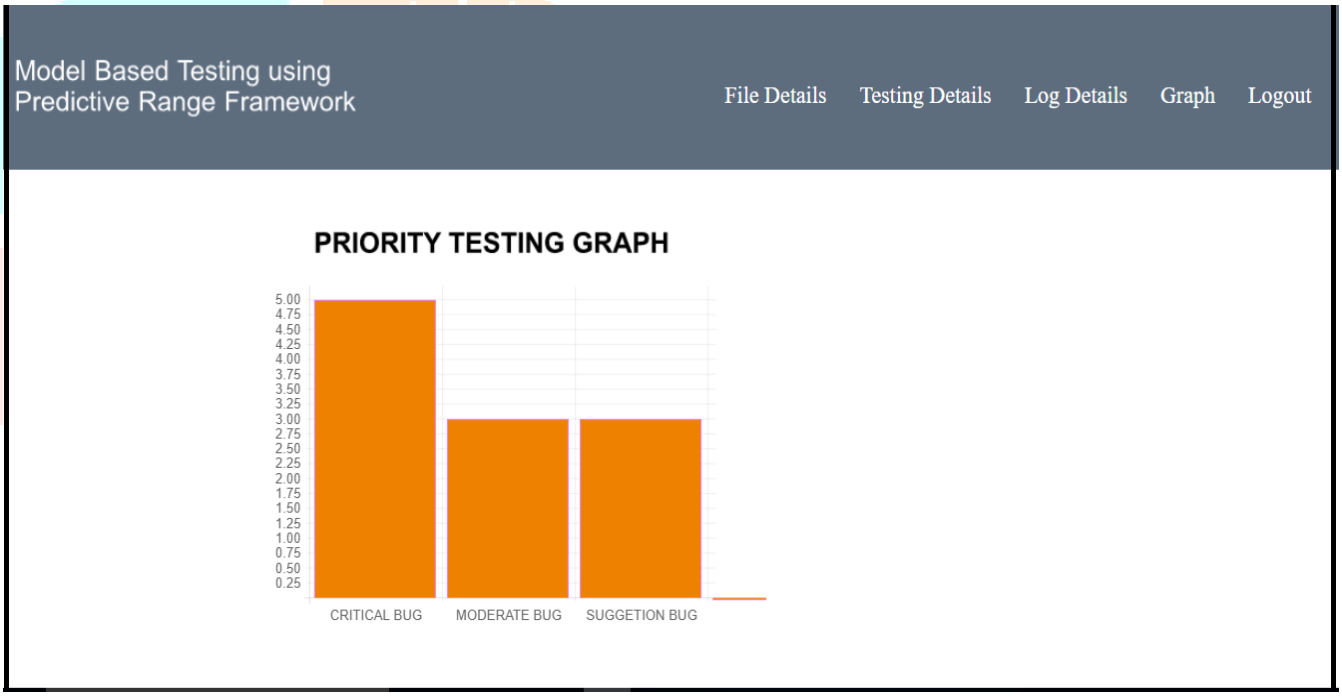


Figure 5: Graphical Representation of Results

VI. 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 paths is 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.

ACKNOWLEDEGMENT

I would prefer to give thanks the researchers likewise publishers for creating their resources available. I'm conjointly grateful to guide, reviewer for their valuable suggestions and also thank the college authorities for providing the required infrastructure and support.

REFERENCES

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

[2] VikasPanthi, 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] MdAzaharuddin 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 voorWiskundeenInformatica, 2005.