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*:** Testing of software package is a vital innovate the SDLC. This part facilitates to envision the standard the software package. This method of testing is split in 2 classes, one is Model based mostly technique and therefore the different is code driven testing. Code driven testing relies on the testing complete code (each and each line). Code driven testing relies on a knowledge flow and management flow that's flow is ordered. In model, based mostly testing approach, focus is on workable module not on each line of code; reason for this approach is to support third party elements, APIs, modules. One needn't to be associate skilled altogether domains, the tester can think about the practicality of individual element not on the small print of a code employed in that element. During this paper the mixture of extended finite state machine and sequence diagram is mentioned.

***Index Terms*: Model Based Testing, Extended Finite State Machine, UML sequence, Coverage and event based testing.**

I. Introduction

Quality of Software is checked with software testing which is an important phase of the software development life cycle. This phase is helps to check the whether software is implemented according to the requirement specified by the user or not. This can be done by following number of testing mechanisms. The ultimate form of these methods is to generate test cases. These test cases will check all components or functionalities. All parameters associated with these components or functionalities should meet standards. These standards are expected to reach the benchmark. Testing techniques used depends upon type of software development life cycle adopted that is waterfall model, spiral model, etc. The type of testing technique also depends upon the stage at which it is emphasized; here stage indicates stage of the software development life cycle. For example, in some case testing is applicable for all phases; in some cases it is restricted for some selected stages.

II. LITERATURE REVIEW

This literature survey is made to make a clear understanding of testing methodologies discussed in the paper. Common example of Registration system is considered and presented in a way so that readers can understand it easily. This is a simple system, where the user enters credentials for login, if a user is registered user, then user will login into the system and proceed for authentication. If the criteria at client side and server side are satisfied, then access is allotted. Event sequence.

TABLE I
DECISION TREE FOR LOGIN MODULE OF THE REGISTRATION SYSTEM

Login	**Rule 1**	**Rule 2**	**Rule 3**	**Rule 4**
<UN PW>	T	F	T	T
PW	F	T	T	F
Login Successful	T	T	F	T

TABLE II
VERIFICATION OF RESULTS LOGIN MODULE OF REGISTRATION SYSTEM.

Parameters	**All Nodes**	**All Edges**	**All Users**	**All POT**
Positive	70	80	78	90
Negative	30	20	22	10

Graph [1] Andre Takeshi Endo mentioned that set of events could be mentioned with this graph, in case of registration system mentioned above; four modules, login, validation, authentication, accession could be considered. Control flow standards: All Nodes: Every node in the registration should be reachable at least once. All Edges: Every edge of the registration system should be reached at least once. Data flow standards: All Uses: Every use should be reachable All Pot Uses: Every potential use should be reached. According to the author of this paper, this work could have been extended to a system, where fault detection could have been much easier. Decisions made with the help decision tree, on vertical side parameter values are used and on horizontal side rules referred to test the successful functioning of functionality is used. According to the decision tree plotted the successful test case is considered on the basis of rules followed by the parameters mention. For example, to use login system, Login should be successful (Rule 1, 2, and 4 must be satisfied and third need not to be satisfied in table I). The results are verified with the help of positive and negative values. In case of login model of registration system, the percentage of visiting all nodes by user is 70 percent which is acceptable (tableII). Combination of extended finite state machine and UML sequence diagram [2] Ching-Seh Wu mentioned that extended finite machine diagram is accompanied by trigger condition and necessary data. The process of EFSM could be categorized in three parts 1. E-block-To evaluate the trigger for all conditions. 2. FSM-block-To compute state next to current state and a signal, which controls A-block. 3. A-block – To perform the required data operations and movements of a data, along with the state diagram, sequence diagram is appended because, when the emphasis is on functionality based technique then dependency needs to be checked. This part is extremely important because as coding is not explored in MBT, dependency check is needed to ensure the appropriate call of the embedded entity. The table III shows path coverage of Extended Finite State Machine, Sequential Test Model, for every step the associated message, a number of states and transitions are mentioned. All paths, all states and all transitions are nothing but the product of paths covered. For example, the total number of tests performed for state = 2*1*2*3*2*1=24. All these parameter give information about the total number of tests in respective field.

TABLE III
TABLE NAME (PATH COVERAGE BY EFSM SETM)

Service Name	**Message**	**No. of States**	**No. of transitions**
Authentication	verify()	2	2
Validation	validate()	1	3
Registration	Signup()	2	2
All Path	P1*P2*Pn	3	1
All State	S1*S2*Sn	2	1
All transitions	T1*T2*Tn	1	2

The scope for improvement in EFSM SeTM is that the system may work only for web services designed using Java. Metamodels and transformations [3] J.J. Gutirrez mentioned important concepts in MBT, that is Meta-model and Transformations. In the system proposed there are five transformations are mentioned about five different meta-models. Metamodel gives information about the functionality, especially first two meta-models describe the functional requirement and the third one specifies the test scenarios to be tested and the fourth deals with the values associated. Fifth meta-model combines, all inputs into the segregated test cases format. In case of registration example of login module mentioned in Fig. 1, functional requirement model will demand a username and password in valid form, test scenarios are whether the user is registered or not. Test value is the actual value of username and password entered and the test case is a combination of all login, validation, and authentication and accessions module. This system demand input in the form of interaction between external user and system. There could be least applicability where users are not actually involved; system likes a compiler. As it is based on a functional requirement completely, quality of the system depends upon the quality of functional requirement collected.

III. SYSTEM ARCHITECTURE OVERVIEW

The architecture of the predictive range framework represents the distinguished approach to test the functionality. The system shows the input of functionality from the client of the software. These functionalities are converted into the. Desired format by the administrator in the form of software requirement specification. The administrator of a system predicts the range of values for every test case to be generated; this is named as an ideal meta-model. The values stored before actually preceding for test cases these values are compared with actual values. This model helps to analyses the performance of every test case.

IV. ALGORITHM

```
Input Er, Fr, Ts, Tv, Sfr
Output Tc
Process
For
  ➨ For                                                        (1)
```

$$T = \sum_{i=0}^{m} i$$

- For (2)

$$Mt = \sum_{i=0}^{m} i = 0^{b} = n$$

- $Er = Ts \oplus Tv \oplus Fr \oplus SFr$ (3)
- IF $Er = Fr$ (4)
- Generate Test Cases (5)
- End For
- End For
- Set
- $Tc = Ts \oplus Tv \oplus Fr \oplus SFr \oplus Pr$ (6)

V. SYSTEM ANALYSIS

Model Based Testing using predictive Range Framework isa reference Meta model that could be referred for any desktopor web applicationPRF based MBT is convincing approach for test cases to checkthe quality of software thoroughly.

VI. RESULTS

Results of Model Based Testing using Predictive Range Framework are compared with the results of the conventional model based system. It has been stated in the base paper that their system is not capable to deal with the system where there is no inference of users. Looking at the result mentioned in the Fig.2 above implemented model outperforms the conventional model. The results can be better interpreted with the help of the error gaps; this will help to analyze the improvement in the performance.

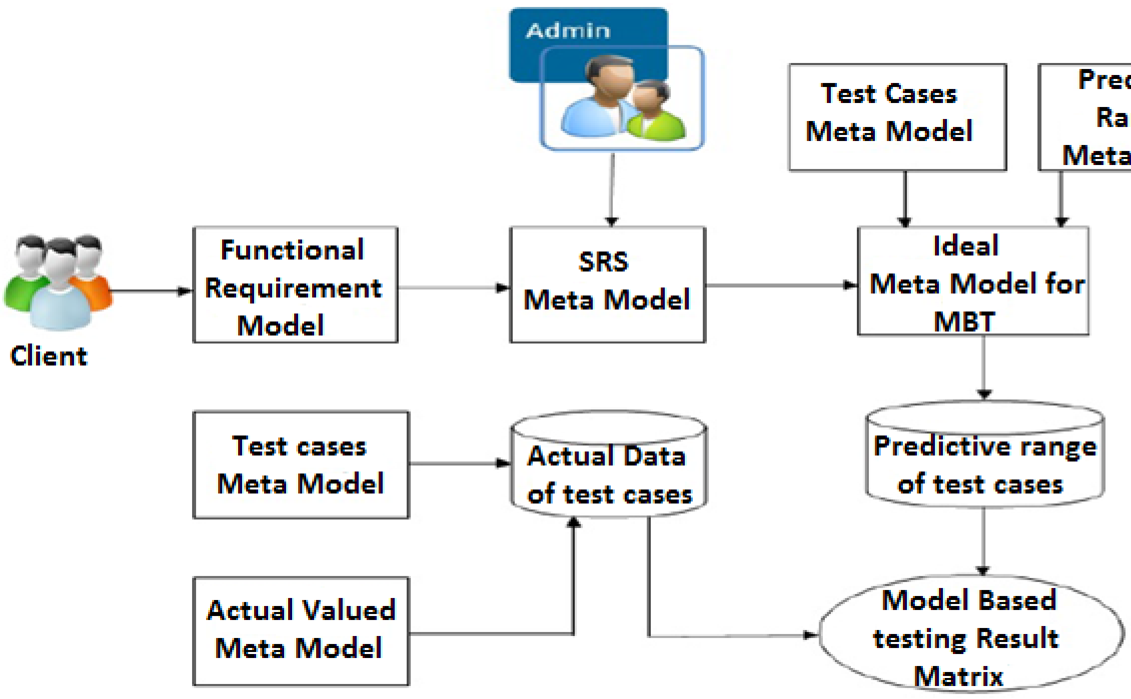
Admin
Test Cases Meta Model
Predicted Range Meta Model
Client
Functional Requirement Model
SRS Meta Model
Ideal Meta Model for MBT
Test cases Meta Model
Actual Data of test cases
Predictive range of test cases
Actual Valued Meta Model
Model Based testing Result Matrix

Fig.1 (Model Based Testing using Predictive Range Framework)

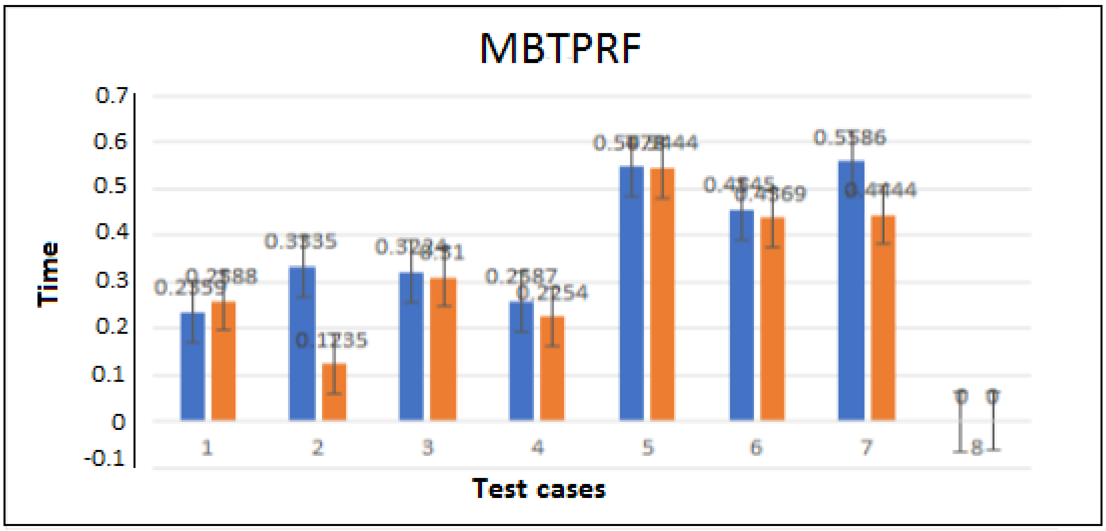

Fig.2 (Results Model Based Testing using Predictive Range Framework)

VII. CONCLUSION

The systems discussed in this paper are especially for Model based testing using the predictive range framework. All these systems heavily rely on functionality. If functionality input is not recorded correctly, then system may get affected with this. There is a necessity of a model based testing technique with detailed testing coverage, consideration of events, inter module communication. There is also need to clearly design template or a system to conduct test cases, these test cases should be strictly examined with the metrics to check quality of the system.

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] Andre Takeshi Endo," Event- and Coverage-Based Testing of Web Services", 2010 Fourth IEEE International Conference on Secure Software Integration and Reliability Improvement Companion.
[2] Ching-Seh Wu , Chi-Hsin Huang," The Web Services Composition Testing Based on Extended Finite State Machine and UML Model", 2013 Fifth International Conference on service Science and Innovation.
[3] J.J. Gutirrez, M.J. Escalona, M. Mejas ," A Model Driven Approach for Functional Test Cases", Elsevier 12 August 2015.
[4] AritraBandyopadhyay, SudiptoGhosh,"Test Input Generation using UML Sequence and State Machines Models" Software Testing Verification and Validation, 2009. ICST '09. International Conference_
[5] 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.
[6] 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.
[7] S. Shanmuga Priya et.al, Test Path Generation Using UML Sequence Diagram, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 3, Issue 4, April 2013 .
[8] 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.
[9] 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.
[10] 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.