RESEARCH ARTICLE OPEN ACCESS

A Novel Model Based Testing (MBT) Approach for Software Validation

Ghanshyam Gagged, Vengatesan Natarajan, Kauser Ahmed P
School Of Computer Science And Engg VIT University Vellore, India
Enterprise-NMS Alcatel-Lucent India Limited Chennai, India
School of Computer Science and Engg VIT University Vellore, India

ABSTRACT
Software validation is an important activity in order test whether the correct software has been developed. Several testing techniques have been developed, and one of these is model based testing (MBT). The main purpose of Model based testing is to test a software product from a user's point of view. Hence, usage models are designed and then test cases are developed from the models. The development of test cases from the usage model can be made automatically by using a tool. In order to increase the efficiency of the test phase, software tools are important. This paper presents a tool for model based testing called MaTeLo. The tool is developed in a joint European project with six industrial partners and two university partners.
As the major outcome of a European project, MaTeLo is a test tool based on the Markov Chain model. This project is used to investigate specification languages for various industries, specifications simulation methods and associated tools. MaTeLo provides an automatic generator which generates test cases based on a Markov Chain usage model and statistical testing methods. MaTeLo supports test execution in heterogeneous environments. In addition, it also provides users with test results analysis and quality reports generation. The tool use model-based metrics to accurately evaluate software reliability and performance throughout the development process. This project integrates the testing tools in real development environments for industrial validation.
- MaTeLo offers a graphical way to formalize the requirements test specification based on a usage model that makes the links between Doors, TestStand.
- MaTeLo generates automatically different kind of TestStand test sequences and systematically improves the SUT reliability; reduce the testing cost and duration
Keywords *- Model Based Testing, MaTeLo, Automation testing, Manual testing, Markov Chains*.

I. INTRODUCTION

One main purpose of the testing phase is to evaluate the quality of a software product. An important quality attribute is reliability, which is defined as failure-free operation during a specified time period. The goal of testing is failure detection, observable differences between the behaviours of implementation and what is expected on the basis of the specification. Testing has a predominant role when developing software.
Generally it is classified in two ways

Manual testing

This type includes the testing of the Software manually i.e. without using any automated tool or any script. In this type the tester takes over the role of an end user and test the Software to identify any un-expected behaviour or bug. There are different stages for manual testing like unit testing, Integration testing, System testing and User Acceptance testing [1]. Testers use test plan, test cases or test scenarios to test the Software to ensure the completeness of testing. Manual testing also includes exploratory testing as testers explore the software to identify errors in it [1].

Automation testing

Automation testing which is also known as test automation is when the tester writes scripts and uses different software to test the product [2]. This process involves automation of a manual process. Automation testing is used to re-run the test scenarios that were performed manually, quickly and repeatedly.

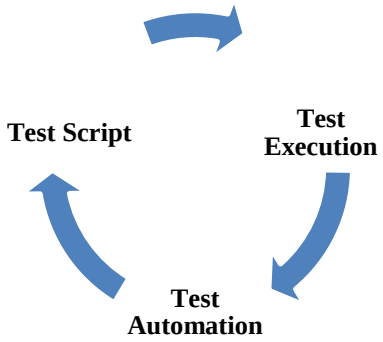

Fig1: Overview of Automation Testing

II. LITERATURE SURVEY

As per Sandeep et al. [5] manual testing is a very costly and time consuming process. This type includes the testing of the Software manually i.e. without using any automated tool or any script. In this type the tester takes over the role of an end user and test the Software to identify any un-expected behaviour or bug. In order to reduce this cost and increase reliability, model based testing approach is used. Model Based Testing (MBT) is a process of generating test cases and evaluating test results based on the design and analysis models. MBT lies in between specification and code based testing [5]. Hence it is also known as Gray-Box testing approach. A wide range of models like UML, State charts, Data flow diagrams, Control flow diagrams etc. have been used in MBT [5].

Nirpal et al have mentioned that Genetic Algorithm starts with a set of first generation individuals, which are then grouped at random from the problem area. The algorithms are generated to execute a series of executions that transforms the present generation into a new and filter generation [6]. Each individual in each generation is evaluated with a fitness function. Based on this evaluation the individual may approach the optimal solution to generate the test cases for the product [6]

Martin Glinz et al. [7] have mentioned that scenarios (Use cases) are used to describe the functionality and behaviour of a (software) system in a user-cantered perspective. A Sequence diagram represents interaction between different process and the order in which they interact. These interactions are arranged in time sequence, and represent the objects and classes that are involved in the scenario. The messages that are required to carry out the functionality between object are also depicted with the help of sequence diagram, as scenarios form a kind of abstract level test cases for the system under development, the idea to use them to derive test cases for the system test is quite intriguing. Yet in practice scenarios from the analysis phase are seldom used to create concrete system test cases [7].

III. MODEL BASED TESTING

Model-Based Testing is the application of Model-Based Design for software testing; this concept has been adopted by many companies and integrated into both hardware and software systems testing. The process typically consumes functional requirements as inputs, and produces test cases. The test cases generation is done according to a model describing the required behavior of the system. The testing effort is made by test engineers during the design of the usage model [4]. The effectiveness of Model-Based Testing is highly increased with automation, i.e. the capacity to automatically translate test cases into executable test material.

The model based testing process begins with requirements [4]. A model for user behavior is built from requirements for the system. Those building the model need to develop an understanding of the system under test and of the characteristics of the users, the inputs and output of each user, the conditions under which an input can be applied; etc

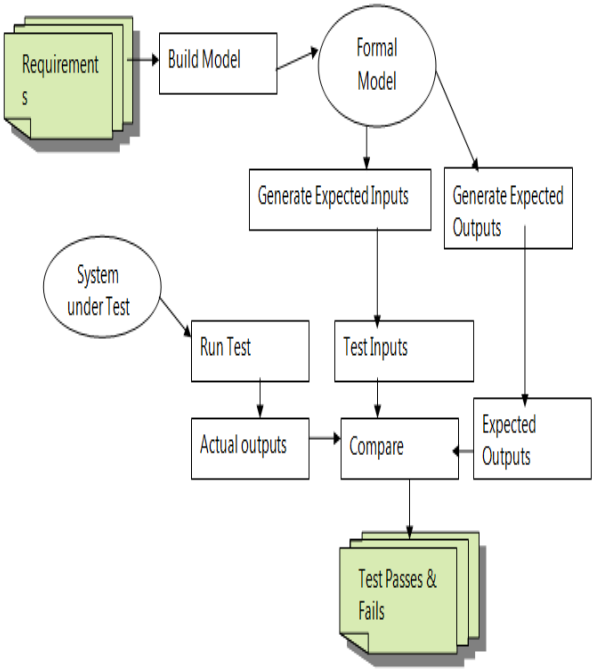

Fig2: Overall Architecture of Model Based Framework

The model is used to generate test cases, typically in an automated fashion. The specification of test cases should include expected outputs. The model can generate some information on outputs, such as the expected state of the system. Other information on expected outputs may come from somewhere else, such as a test oracle [4]. The system is run against the generated tests and the outputs are

compared with the expected outputs. Here, too, automation is extremely useful. The failures are used to identify bugs in the system. The test data is also used to make decisions, for example, on whether testing should be terminated and the system released.

IV. MaTeLo MODEL

Model can be used to represent the desired behavior of the System under Test (SUT). MaTelo is more independent of development technologies and it can be integrated ideally with the most frequently applied methodologies and processes [8]. MaTelo uses Markov Chains to describe the test model of the system. The test model is basically a collection of state diagrams and transitions and it represents the possible usage of the system and also helps to improve the business value for customer as well as the vendor [8]. MaTeLo tool accepts two kinds of input information: scenarios and usage patterns. These inputs are presented as MSCs or UML sequence diagrams, which specify the user interactions with the system. Users can use the tool to translate them into a state chart. Or users can directly specify the usage of system as state charts, and feed into MaTeLo [9]. All of these models describe the usage of the software

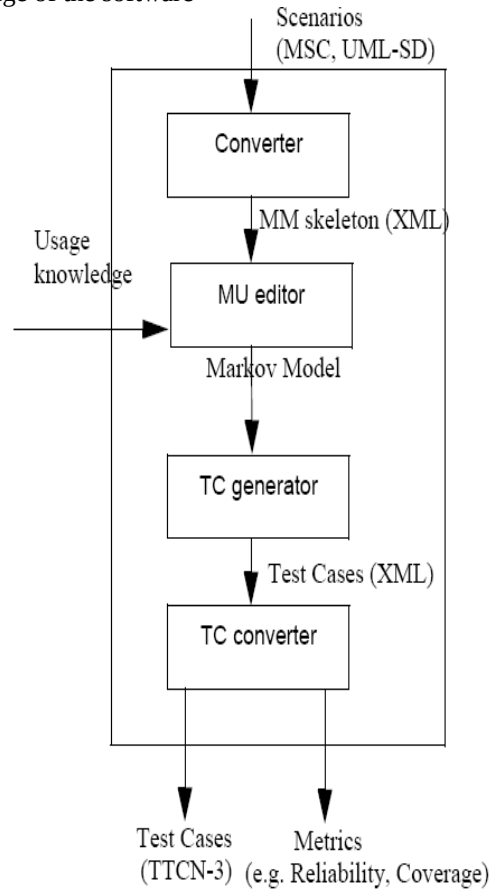

Fig 3: Overview of the logical component in the MaTeLo tool

When the state chart has been derived, probabilities in the transitions between states are either automatically calculated using a pre-defined distribution or they can be specified by the user [9]. This is carried out in the Markov Usage Editor (MU). In the MU, one or several usage profiles can be used. This is valuable since different intended users will utilize the software differently.

The state, condition, asynchronous and the transitions can be added in the graph by using the button available in tool bar of MaTelo- Usage Model Editor [10]. For state properties addition use the button in the tool bar to enter in selection mode. It is possible to create two different types of states

- A normal state
- A macro state which calls sub-chains

To add a properties on transitions there are four type of parameter available on transition: Input, Manual Expected Result (MER), Automatic Expected Result (AER) or Function.

The edition window of transition properties include 5 tabs

- The "main" tab allows defining the usage frequency, input, AER, MER and transferring function.
- The "Condition" tab is active only on conditionals transitions and permit to define the going by condition of this transition.
- The "context" tab allows defining one or several functions to update the context.
- The "TestStand Functions" tab allows defining the functions which are used with TestStand.
- The "Requirement" tab allows adding some requirement for the current transition.

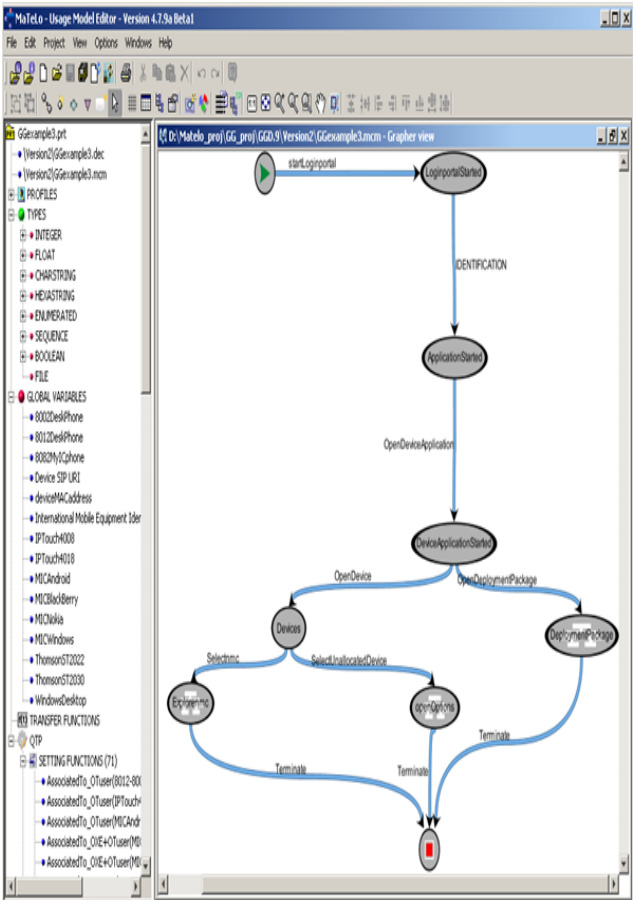

Fig 4: A screen shot of the usage Model editor

When the usage models have been designed, the tool checks whether the model has been correctly designed and then it converts the usage models into test cases [12]. The test language used is TTCN-3. In Figure 5 a screen shot of the test interface of MaTeLo is shown.

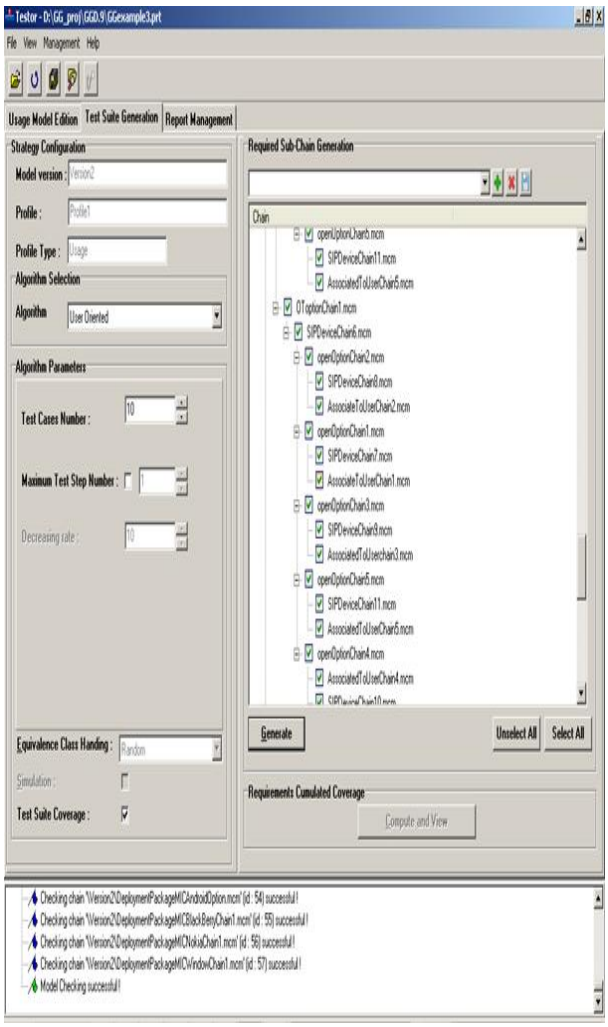

Fig 5: A screen shot of test suite generator

To generate the test suite you have to select the test suite generation tab, there are five algorithms available for test suite generation User-oriented generation, User-oriented generation (limit), Most probable generation, Minimum generation, Tagged path [11]. If the generation is ok then the generation report will be generated

After the test cases have been derived, they are run on the system under test and the outcome is analyzed. Several important metrics are automatically calculated in the tool, for example, the reliability of the system and the coverage of the usage models [12]. This provides useful information to the testers of whether the software is good-enough to be released. The output of the testing also shows what failures have been detected. The reliability calculation is based on theory of Markov models.

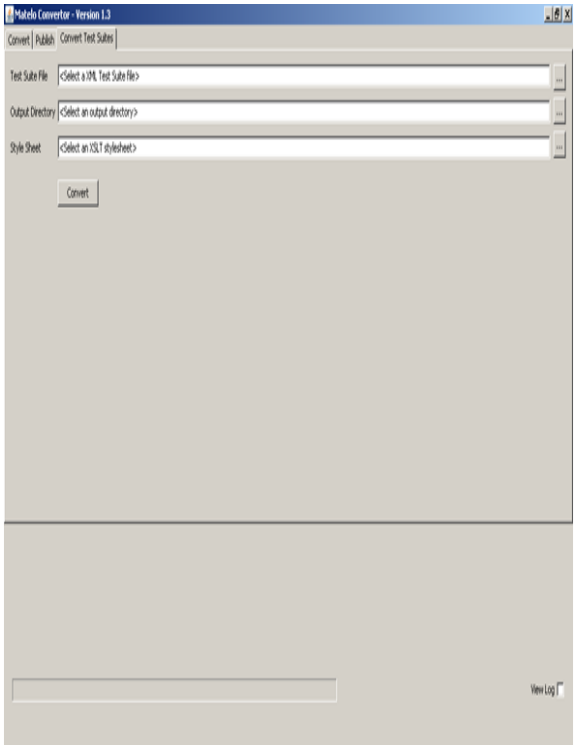

Fig 6: A screen shot of convertor

The objective of this functionality is to convert the test files generated by MaTeLo to any test format. The conversion is done with a style sheet XSLT. The XSLT processor used is XALAN 2.7.1 (this library is integrated to the convertor), this property allows for writing a test file directly from the style sheet. Apart from this convertor is also used to convert the version of model.

V. IMPORTANCE OF MaTeLo TOOL

The main objective of MaTeLo is to utilize SUT to derive test cases from usage models. Hence, this means that the testers will develop a model of the usage of the software [10]. Since different users will use the software in different ways, several usage profiles can be developed. For example, the usage profiles can be divided into novice users and experienced users. Since the usage of the software will differ

Another benefit of MaTeLo is that if someone is more familiar with Message Sequence Charts (MSC) or UML sequence diagrams, the user can specify the basic input to the tool in these notations [11]. After the communication between the user and the software under test has been specified, MaTeLo automatically converts the input to a state chart. Then the user can provide probabilities to the transitions in the chart, or even this can be made automatically by using a certain distribution.

Another main benefit of MaTeLo is the automatic calculation of quality metrics, such as reliability and coverage of the usage models [10]. This provides valuable input to the testers, so they know when to stop testing. The reliability value can be decided beforehand; then the tool will provide answers of whether it is necessary to continue testing or not.

The main benefits of MaTeLo are

- Tests software from the users' point of view by using different usage profiles.
- Automatically converts usage models into test cases.
- Calculates important metrics (for example, reliability) that can be used as stopping criterion of the testing.

VI. CONCLUSION

This paper discusses a review of model-based software testing and MaTeLo tools as a tutorial. It presents the basic concepts, workflow, objectives, importance and benefits in model-based testing. Model-based testing involves the following major activities: building the model, defining test selection criteria and transforming them into operational test case specifications, generating tests, conceiving, setting up the adaptor component and finally executing the tests on the SUT. MaTelo is based on statistical usage testing. At the front-end of the tool, testers provide the tool with models the intended usage of the software. This means the most critical failures will be detected, since the test cases cover the usage and not the technical specification. The output of the tool is test cases, which can be run on the system under test. Furthermore, the tool also provides testers and managers with important quality metrics such as reliability of the system and coverage of the user model.

ACKNOWLEDGEMENTS

I would like to show my immense gratitude to **Mr. Murali Krishnan Arunachalam**, **program director, Alcatel-Lucent India Pvt Ltd** for giving me the opportunity and support he has provided during my project.

I express my sincere thanks to **Mr. Vengatesan Natarajan**, **QA Manager, Alcatel-Lucent India Pvt Ltd** for giving me the opportunity and guidance to do this project. His constant support and discussion on relevant topics cannot be thanked by mere use of words

I take immense pleasure and very deep sense of gratitude in conveying my special, sincere and heartfelt thanks to my guide **Mr. Kauser Ahmed P, Professor School of Computing Science and Engineering, VIT University, Vellore**.

REFERENCES

[1] R. Mall, Fundamentals of software engineering, 2nd Ed. New Delhi: Prentice-Hall of India Ltd, 2008.

[2] S.-D. Gouraud, A. Denise, M.-C. Gaudel, B. Marre. A new way of automating statistical testing methods. Proceedings of the 16th IEEE international conference on automated software engineering, 2001

[3] Blackburn, M., Busser, R., and Nauman, A. "Why Model-Based Test Automation is Different and What You Should Know to Get Started." In International Conference on Practical Software Quality (2004).

[4] A. Pretschner, W. Prenninger, S. Wagner, C. Kühnel, M. Baumgartner, B. Sostawa, R. Zölch, and T. Stauner, "One Evaluation of Model-Based Testing and Its Automation," Proceedings of the 27th International Conference on Software Engineering, May 15–21, 2005, St. Louis, Missouri, ACM, New York (2005), pp. 392–401.

[5] Sandeep K. Singh, Sangeeta Sabharwal, J. P. Gupta, "A Novel Approach for deriving test scenarios and test cases from events", Journal of Information Processing Systems, Vol. 8, No. 2, June 2012.

[6] Premal B Nirpal, K. V. Kale, "Using Genetic Algorithm for Automated Efficient Test case generation for Path testing", Int. J. Advance Networking and Application, Volume 02 Issue 06, pp 911-915 (2011)

[7] Johannes Ryser, Martin Glinz, "A scenario-Based Approach to Validating and Testing Software Systems Using Statecharts", 12th International Conference on Software and Systems Engineering and their Applicaions ICSSE 99

[8] R. V. Binder, "Testing Object-Oriented System Models, Patterns, and Tools", NY: Addison-Wesley, 1999.

[9] Kelly D.P. and Oshana, R.S., "Improving software quality using statistical testing techniques", Information and Software Technology, 42(12):801-807, 2000.

[10] Whittaker J. A and Thomason, M. G., "A Markov Chain Model for Statistical Software Testing", IEEE Transactions on Software Engineering, 20(10):812-824, 1994

[11] Kirk Sayre. Improved Techniques for Software Testing Based on Markov Chain Usage Models. PhD thesis, University of Tennessee, Knoxville, December 99.

[12] J. Grabowski, A. Wiles, C. Willcock and D. Hogrefe. On the design of the new testing language TTCN-3. Proceedings 13'h IFIP International Workshop on Testing Communication Systems (TestCom 2000), Ottawa, August 2000.