IJCSE International Journal of Computer Sciences and Engineering Open Access
Survey Paper Volume-4, Special Issue-5, Jul 2016 E-ISSN: 2347-2693

MBT for Functional Testing of Embedded Systems

Mohammed Naim Khan, Namita Arya and Amit Prakash Singh*

University School of ICT
Guru Gobind Singh Indraprastha University, Dwarka, Delhi

Available online at: www.ijcseonline.org

***Abstract*—** Embedded Systems require combination of Software Electrical and Mechanical Engineering. This leads to increased complexity of Embedded Systems with shorter production cycle. These evolutions pose new challenges to traditional embedded system testing approaches, because besides time constraints, embedded system products are expected to meet quality constraints prior to their deployment in field. The Cost and Time of Testing Embedded Systems consumes a considerable portion of entire development cycle. With Increased Complexity of Embedded Software it becomes imperative that our Functional Testing is effective at detecting issues in the system. With the above requirements we propose to use Model Based Testing (MBT) for the functional testing of Embedded Systems during the manufacturing process in the production line. The comprehensive and systematic approach of MBT facilitates automated testing with reduced test time and cost along with improved quality of the product.

***Keywords*—** Embedded system, Model Based Testing, Functional Testing, System Testing

I. INTRODUCTION

Embedded Systems design and development is governed by time to market and productivity constraints. This forces researchers to improve the overall embedded system development cycle including design, development and Testing. Embedded systems have integrated hardware and software components used in time critical and life critical applications where failure results in high financial loss or even in injury or death. As a consequence, it is important to ensure the correctness of these systems with systematic and comprehensive testing techniques.

Model-based testing (MBT) is defined as an approach whereby test sequences are generated automatically from models of the system under test, using different kinds of computing algorithms to optimize that process. Hence model-based Automated Test Generation (ATG) is the key feature of MBT. MBT is used in different flavours by several tools and projects. A more detailed list of applications of that approach using various notations is presented by Utting et al in their Taxonomy of Model-Based Testing [2].

UML and UTP introduced by the Object Management Group (OMG) [1] are the foundations for the Model Based Development (MBD) and Model Based Testing (MBT) of embedded systems respectively. MBD and MBT methods provide automation in the overall Embedded System development cycle. Adapting automation is expected to improve time to market, reduce cost and improve quality constrains of Embedded System.

The challenge in Functional Testing of Embedded Systems is the effort involved in developing and evaluating a test suite that systematically tests the system and reveals faults effectively. To deal with increased complexity and meet the high quality expectations of embedded systems it becomes necessary to test the functionality of embedded system in production line. In this paper, we present a functional test methodology for embedded systems in production line that helps reduce test time and improve quality of the system being delivered. The paper is structured as follows: In section 2 provides Background of Model Based Testing. In section 3 related works are described. In section 4 our approach is presented with the help of example, followed by conclusion in section 5.

II. BACKGROUND:MODEL BASED TESTING

Software Testing consumes 30 to 60 percent of the total development time [2]. Model Based Testing enhances the level of automation by automatic execution of test cases along with automatic generation of test cases. This provides repeatable testing process confirming coverage of all behaviours of the system and allowing tests to be linked to system requirements.

The general process of MBT begins with modelling the System under Test (SUT) and then the MBT tool generates the abstract test cases from that model based on the test strategy selected. These tests can be executed manually by the tester as the abstract test cases are similar to high level sequences that would be designed manually in action word testing or keyword testing [5]. These tests cases are easily understood by humans are complete to be used by manual tester for execution on SUT. On the other hand automatic executable tests [6]

can be generated from the abstract test cases. Finally the generated test suit is executed to generate the test results. The SUT is then evaluated based on these test results. The MBT process is divided in following steps:

A. Model the SUT

The Test model of SUT represents its expected behaviour. Standard modelling languages like UML are used to formalize the control points and observations points of the system. Embedded Systems inherently exhibit state driven behaviour so UML state charts which are extensions of Finite State Machines (FSM) can be used. Various tools like Spec Explorer, AGEDIS, Qtronic, Test Optimal, etc. [3] are available for state chart modelling of UML systems.

B. Select Test Generation Criteria

There are infinite number of test cases that could be generated for the test model of SUT, so we need to choose some test generation criteria that would help generation test cases for increased coverage of SUT. Various testing strategies like equivalence partitioning, cause-effect testing, pair-wise testing, Boundary Value Analysis, etc. [4] can be used for test selection criteria from the model. For embedded systems we select covering all the transitions in state machine as our test generation criteria.

C. Test Case Generation

This is a fully automated process which generates abstract test cases that are typically sequence of high level SUT actions with input and its expected output for each action. These tests can be executed manually by the tester as the abstract test cases are similar to high level sequences that would be designed manually in action word testing [5]. These tests cases are easily understood by humans are complete to be used by manual tester for execution on SUT. On the other hand automatic executable tests [6, 7] can be generated from the abstract test cases. Various tools such as IBM rational, HP quick Test, Test Optimal etc. [4] are available in market for generation of test cases or test suits from abstract models.

D. Execute the Tests

The generated test cases and test suits are executed either manually or automatically on SUT. Constraints defined during the model creation on the state machines like guard, pre/post conditions of triggers, etc. [8, 9] should be evaluated during the execution of the test cases. Many studies show that this is an effective way of detecting failures [10, 11].

E. Result Analysis

The final step in MBT is to analyse the results and take corrective actions. The test case execution results in some tests pass and some fail. Analysis is similar to traditional test analysis process where the failing results indicate some deviation from the expected behaviour of the SUT. The test may fail due to fault in SUT or it may fail due to error in model or requirements. Thus after result analysis we may have to refine our model based on the fail results due to the model error. MBT is good for finding SUT errors but it also exposes requirement errors [2].

III. RELATED WORK

As Embedded Systems encompass every aspect of our lives the complexity of the embedded systems increases and with it increases the number of embedded systems produced on daily basis. The increased complexity and production quantity of Embedded Systems call for new software development and test methodology. The MDD technique is being widely adopted for developing embedded systems as it offers several advantages over the traditional ways of development and Testing. Studies conducted in [12, 13] use general-purpose UML for modeling embedded software. From Various studies we see that UML finds widespread applicability for modeling embedded software systems [14]. The same is also evident from the various commercial tools available for model based design of Embedded Systems.

For systems being designed using MDD and MDA, a model based testing approach is suggested as it is associated with the MDD paradigm favouring models over code [15]. Model based testing is typically used for generating test cases from behavioural models of system. There has been a drastic increase in the number of model based testing techniques and tools available both commercial and academic in recent years. Although these techniques have a common goal, they differ in terms of modeling languages (e.g. UML), automatic test generation, testing targets and tool support. A survey of 15 different studies on model based testing techniques is provided in [15].

A majority of the model based testing techniques in [15] focus on tools and techniques for automatically generating test cases from models in the context of UML diagrams and the UML Testing Profile (UTP) [6, 7, and 15]. Studies conducted in [15] show the advantages of using model based testing approaches.

From the above discussions, it may be seen that there is a numerous work carried out for utilizing MDD for

embedded systems development and testing. In our previous work we had proposed a comprehensive testing technique for embedded system PCBA wherein we had proposed the various testing techniques for functional testing using USB, UART or Ethernet port. The above methods work well but require additional code for implementing the functional test. This may not be feasible with memory constraint embedded system. The work presented here is an extension of the functional testing of Embedded Systems in production line which is applicable to the systems developed using MDD and MDA architecture.

IV. PROPOSED METHODOLOGY

The architecture of functional test method for embedded system in production line is shown in figure 1.

The system under test (SUT) is connected adaptor or test execution tool that manages the interactions with the SUT. The adaptor records the input sequence to the SUT and the results generated by SUT. The Adaptor is connected to the host PC that runs the test case or test suits and stores the result of the testing performed. The Host PC send the test to be executed to the adaptor and gets the result of execution from the adaptor. If online MBT process is used the tests will be executed by the modelling tool that would be running on host PC. So the MBT tool itself will manage the test execution and store the results. With offline MBT we have generated the test suits in the form of test scripts in some existing language that are already in use for testing purpose. These test scripts are then running on the host PC and the results of execution of the The results stored in the host PC are analysed for correctness by comparing it with the predefined expected output from the SUT. As the Model is already stabilized during the development process of embedded system the failures occurring are completely of the SUT. Thus the models developed during the development process can be effectively utilized for functional testing of embedded system in the production line. We next demonstrate the functional testing of embedded system using the Test Optimal tool.

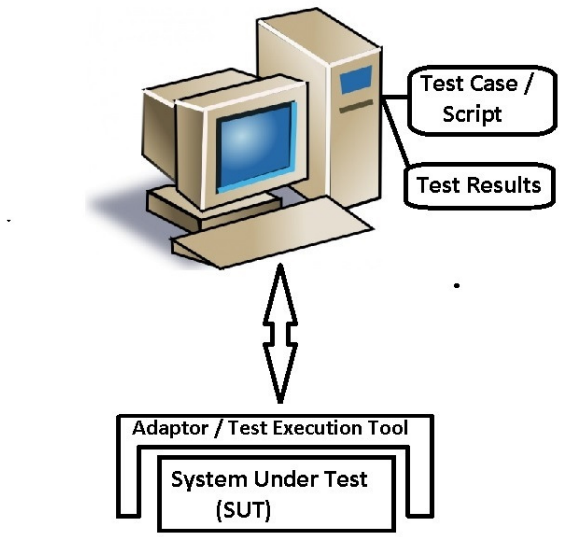

Figure 1: Test Architecture

For the SUT we take a simple example of porch light the behaviour of which can be stated as follows:

- There is a switch for porch light with three states: ON, OFF and Auto.
- When switch = ON the porch light is turned ON.
- When switch = OFF the porch light is turned OFF.
- When switch = Auto the porch light shall turn ON or OFF based on the ambient light which is read by light sensor in the range of 0% to 100%.
- At start up in auto mode the porch light shall turn ON/OFF if ambient light is below/above 40%.
- In Auto mode the porch light shall turn ON if ambient light falls below 40%.
- In Auto Mode the porch light will turn OFF if ambient light is more than 60%.

The top level model for the porch light is shown in figure 2 and the test optimal model is shown in figure 3.
The Test Optimal model requires a start state and an end state. The same is depicted in the model. From the Init state we have three outgoing transitions depicting the three states of switch: ON, OFF and Auto. The Auto state has single outgoing transition to the check brightness that measures the ambient light and accordingly switches to ON or OFF state. In Auto mode we continuously need to monitor the ambient light so both ON and OFF state return to Auto state for the brightness measurement. The state is only changed if required conditions are met. The

input switch can toggle at any time asynchronously so from all the three states ON, OFF and Auto we have a transition back to Init state and finally we have transitions to End state from ON and OFF state as required by Test Optimal tool.

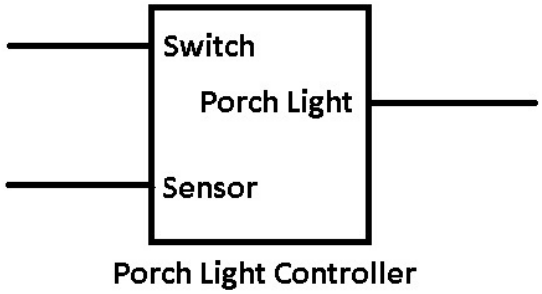

Figure 2: Top Level Model of Porch Light Controller

Test Optimal supports various test generation criteria we select random walk greedy algorithm with 100% coverage to test case generation. The other options are shown in figure 4.

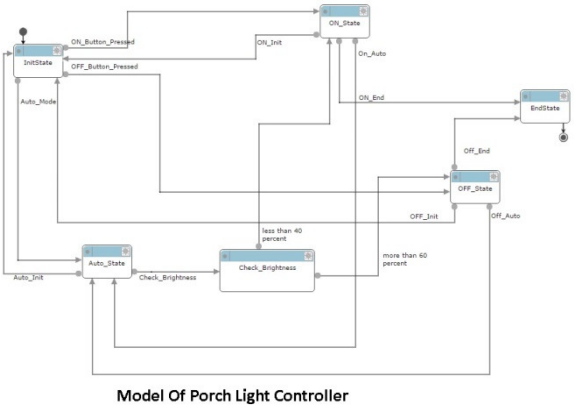

Figure 3: Test Optimal Model

The random walk greedy approach ensures coverage of all the transitions in the model. It prefers un-traversed transitions over already traversed transitions, as the result it can achieve better coverage of the model more quickly.

The execution coverage of the model with the selected test case generation and specified coverage criteria can be seen from the coverage graph. The Coverage graph of porch light controller is shown in Figure 5.

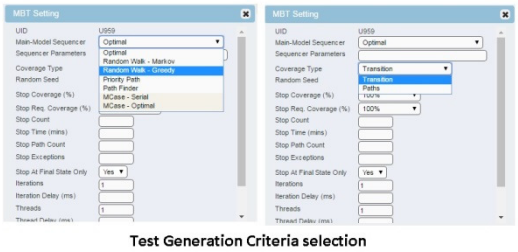

Figure 4: Test Generation Option

The coverage graph is very helpful when due to time limitations in the production line we would like to test only critical paths of our model that affect the SUT in such a manner that it can cause system failure. So coverage graph helps us visualize the transitions being tested when we set certain priority for state traversal test using MScript in test optimal. Using MScript we can set guard conditions, certain specific actions for data path or control path coverage.

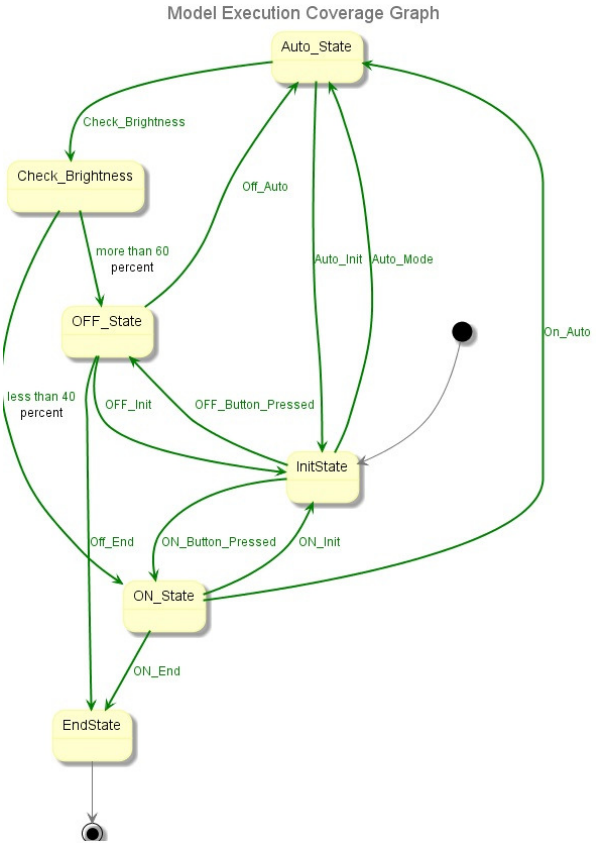

Figure 5: Coverage Graph

The test optimal tool also provides the Test sequence diagram that gives the test sequence in the form of transition diagram depicting all the states in the model

and how the tests proceed from one state to another. The same information can be obtained in tabular form.

The Test sequence diagram for the porch light controller is shown in Figure 6. As can be seen from the test sequence diagram the tool has automatically generated three test coverage paths from Init state to End state and each time the test sequence traverses through different path in order to cover all the states and paths in the model.

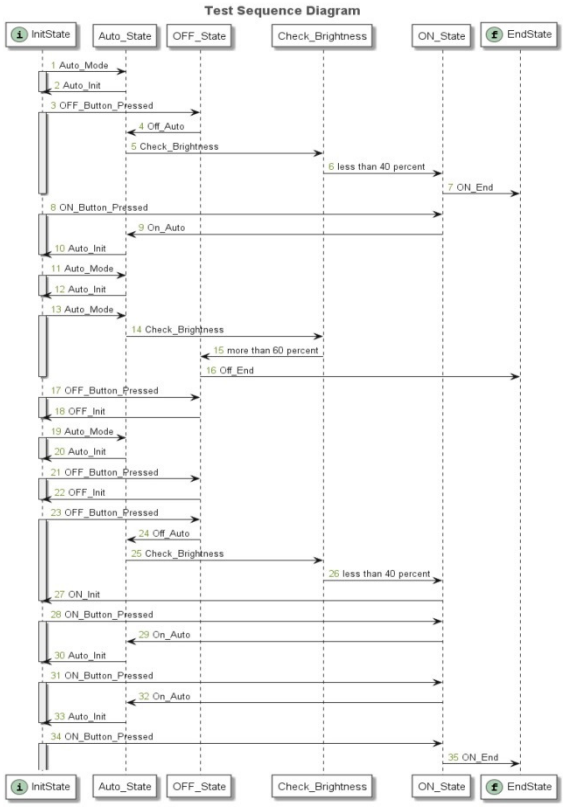

Figure 6: Test Sequence Diagram

These state transitions represent test cases for the model. These can be further fine-tuned as per the requirements by using the MScript. Test Optimal has inbuilt support for web based testing. We can also have the test cases generated for offline testing by using the traditional scripts that run on the host PC and collect the results from the SUT in response to each test case executed. These scripts further validate the results against the expected output from the SUT.

Thus MBT leads to automatic generation of test cases that can be optimised as per the requirement of critical path, cost and time constraint in the production line. This results in efficient and effective functional test strategy for Embedded System.

V. CONCLUSION

Embedded systems are covering more and more aspects of our daily life thus increasing in complexity and finding application in safety critical applications. This increases the number of Embedded Systems produced on daily basis and also it is necessary to functionally test the embedded systems for quality and safety. Along with this cost constraint of embedded system due to its inherent commercial applications restrict the test time allocation for the embedded systems.

In this paper we have proposed a functional test methodology of embedded system using model based testing in the production line. This test strategy is best suited for systems developed using MBD as the model for the system would then be readily available. Modelling of the system requires to put in additional efforts towards the architecture of the system to understand its complete behaviour. This process is helpful in finding specification errors, omissions and conflicts. When designing the system model, the tester should not consider test cases – the tester should, only consider of the system behaviour.

The MBT tools provides various test execution environments through a customizable interface and gives end to end automation from test design to test execution and management. MBT introduces formal modelling and enhances the test development processes. Competency building within the conventional tester is necessary and the tester should be trained for modelling and programming skill demands of MBT tools.

The functional testing of Embedded Systems using MBT provides the benefit of High productivity maintaining Good Quality with lower costs. For the Embedded Systems already developed using the MBD process applying the MBT in the production line is easily feasible increasing the test coverage and testing the system in real time.

REFERENCES

[1] OMG: http://www.omg.org [28/04/16].
[2] Mark Utting and Bruno Legeard, “Practical Model-Based Testing: A Tools Approach". Elsevier, 2006.
[3] Muhammad Shafique, Yvan Labiche, “A Systematic Review of Model Based Testing Tool
[4] Support,” International Journal on Software Tools for Technology Transfer February 2015, Volume 17, Issue 1, pp 59-76.
[5] Yasir Dawood Salman, Nor Laily Hashim, “An Improved method of obtaining basic path testing

for test case based on UML State Chart," International Symposium on Research in Innovation and Sustainability 2014 (ISoRIS '14)

[6] B. Beizer, "Software Testing Techniques", Second Edition, 1990.

[7] C. Bourhfir, R. Dssouli, E. Aboulhamid, N. Rico, "Automatic executable test case generation for extended finite state machine protocols," Testing of Communicating Systems Part of the series IFIP — The International Federation for Information Processing pp 75-90.

[8] Fuqing Wang, Shuai Wang, Yindong Ji, "An Automatic Generation Method of Executable Test Case Using Model Driven Architecture ," The 4th International Conference on Innovative Computing, Information and Control (ICICIC), 2009.

[9] Fabrice Ambert, Fabrice Bouquet, Jonathan Lasalle, Bruno Legeard and Fabien Peureux, "Applying an MBT Toolchain to Automotive Embedded Systems: Case Study Reports," The Fourth International Conference on Advances in System Testing and Validation Lifecycle (VALID) 2012.

[10] Jonathan Lasalle, Fabien Peureux, Frédéric Fondement, "Development of an automated MBT toolchain from UML/SysML models," SI : FM & UML Innovations in Systems and Software Engineering December 2011, Volume 7, Issue 4, pp 247-256

[11] Lionel Briand, Yvan Labiche, "A UML-Based Approach to System Testing," UML 2001 — The Unified Modeling Language. Modeling Languages, Concepts, and Tools Volume 2185 of the series Lecture Notes in Computer Science pp 194-208.

[12] L. C. Briand; M. Di Penta; Y. Labiche, "Assessing and improving state-based class testing: a series of experiments," IEEE Transactions on Software Engineering 2004, Volume: 30, Issue: 11 Pages: 770 - 783.

[13] Sang-Uk Jeon, Jang-Eui Hong, Doo-Hwan Bae, "Interaction-based Behavior Modeling of Embedded Software using UML 2.0," Ninth IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC'06).

[14] Martin Grant, Lavagno Luciano, Jean Louis-Guerin, "Embedded UML: a merger of real-time UML and co-design," Springer 2003.

[15] Helmerich, A., Koch, N. and Mandel, L., Braun, P., Dornbusch, P., Gruler, A., Keil, P., Leisibach, R., Romberg, J., Schätz, B., Wild, T. Wimmel, G.: "Study of Worldwide Trends and R&D Programmes in Embedded Systems in View of Maximising the Impact of a Technology Platform in the Area," Final Report for the European Commission, Brussels Belgium, 2005.

[16] Mohamed Mussa, Samir Ouchani, Waseem Al Sammane, Abdelwahab Hamou-Lhadj, "A Survey of Model-Driven Testing Techniques," Ninth International Conference on Quality Software 2009, Pages: 167 - 172.

AUTHORS PROFILE

Amit Prakash Singh is an Associate Professor in University School of Information & Communication Technology, Guru Gobind Singh Indraprastha University, Delhi since 2012. He is working in the are of Embedded Testing and Artificial Neural network.

