2007-01-0503

Model-Based Automated Validation Techniques for Automotive Embedded Systems

Daehyun Kum, Joonwoo Son, Seonbong Lee
Daegu Gyeongbuk Institute of Science & Technology – Department of Mechatronics

Ivan Wilson
DsysD Ltd.

ABSTRACT

Model-based approaches can improve quality and reduce cycle time by simulating the models to perform early validation of requirements. These approaches can provide the automated validation techniques by generating test cases from the model. This paper describes model-based automated test techniques in all phases of the product life cycle to maximize the early validation capabilities of model-based development processes. The paper proposes the model-based test process framework for all modeling phases including system model, architectural model and auto-generated software, and test automation technique which consists of automatic test generation, execution and analysis. A Test Management System, which enables the automatic generation of requirement-based test cases, analysis of the test results and test database management through entire test process, is developed through this study. In addition, an automatic target execution system, which comprises target hardware, simulation command transmission and target monitoring software, was developed to test the auto-generated ECU code on the real target system in the early stage.

INTRODUCTION

The Automotive embedded system becomes even more complex and occupies a greater portion of vehicle price. In order to design products which improve reliability and quality as well as reduce development time and cost, automotive industries have been focusing on well structured development processes such as model-based approaches [1][2]. But to meet demand for high quality and lower cost it is becoming increasingly necessary to rely on new test systems and methodologies in all phases of the product life cycle. Erroneous automotive embedded software may cause serious car accidents which relate to human safety. Therefore, it is essential to detect any errors in the embedded software through all test processes. The embedded software is often subject to change to meet new customer needs and revised legal requirements, so it is desirable to automate testing to improve the efficiency of the process.

Test automation consists of the following three parts: test case generation, test execution and analysis reporting. Many research efforts have been made for these automation techniques. During the requirement capturing phase, automated testing can be realized by introducing formal language [3][4][5]. Automatic generation of test cases from statecharts, control flow, information flow and MSCs(Message Sequence Charts) are widely used for specifying the dynamic behavior of the model [6][7][8], and automation concerning test systems and processes has been presented [9]. However, research on seamless test automation process based on the model-based approach has not been applied.

In this paper, an automated validation process framework for all phases of the product life cycle and automated testing techniques are proposed to help the seamless validation process. The proposed automatic testing techniques consist of a Test Management System (TMS) which assists requirement-based test cases generation, automatic test execution, automatic analysis report generation and test database management and an Automatic Test Execution System, namely Target Practice, which comprises target hardware, simulation command transmission software and target monitoring software for auto-generated software testing. In order to show the effectiveness of the proposed process and techniques, a case study on automotive network system for window lift feature was demonstrated. Where Target-Practice is used to test automatically on a real target.

MODEL-BASED AUTOMATED VALIDATION PROCESS FRAMEWORK

During automotive embedded system development, testing is repeated in each development phase, validating the design at every step. The substantial

benefit of the model-based method is to the ability to automate the entire test process by generating the test cases from the model. Figure 1 shows the model-based design and validation process. In this section, we will discuss about model-based automated validation process framework based on this process.

MODEL-BASED TEST PROCESS

The model-based test process like the model-based development process follows the typical V-process, however the testing should be performed separately from the design. Development starts with requirements capture and analysis. The success of any product development depends on the creation of clear and unambiguous requirements. A clear understanding of the system requirements must precede testing. Functional models should be created according to the requirements. The vehicle electronic system is too large to be addressed singularly so it is often decomposed according to functional groups. Once the main sub-systems have been identified their input and output signals should be clearly defined. Also the exact test signals should be defined so that the test result is reliable and automated testing is performed efficiently. At the early stages of the system development virtual prototypes of the functional models enable us to test and validate the software functionality without hardware. Once the system and component design have been validated in the virtual environment, the designs should be moved to the implementation phase. To generate code for embedded applications the model must be enhanced to include the software design and implementation details. These newly added items impose execution constraints on the model, however the system must still satisfy all the requirements. This implementation model should be validated. Finally, all the source code including the generated application code, I/O drivers, the communication kernel and operating system code are integrated.

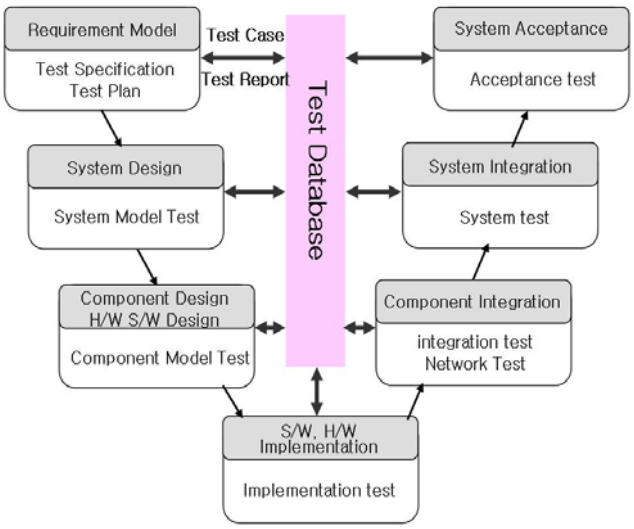

Figure 1.Model-based design and validation process

Test results from an ideal PC environment may differ from those where the model runs in a real target, so software performance and functionality must be tested in a target ECU. Then an integration test is performed to see if the complete integrated system satisfies the specifications.

TEST AUTOMATION IN MODEL-BASED PROCESS

Test automation is composed of three parts:

- Automatic test case generation
- A test execution system
- Test results analysis and documentation

The first test cases should be created manually based on the requirements. The simple, atomic and formal test cases are suitable for automation. Complex test cases can lead to errors when executing the tests automatically. A test system capable of automated testing is needed for each test phase. Depending on the development phase and the system under test, different test systems are chosen. The following, table 1, shows a selection of test techniques. After test execution, the test results should be analyzed to find and correct any errors. Test results and related information should be documented throughout all test processes. Therefore, test management system and automatic target execution system are developed to support test automation in model-based approach.

Table 1 Test phases and techniques

Test phase	**Test technique**
System model testing	Model in the loop simulation
Implementation testing	Software in the loop simulation
Integration testing	Hardware in the loop simulation

AUTOMATED TESTING TECHNIQUES

TEST MANAGEMENT SYSTEM

Usually there are millions of test cases to be performed on automotive embedded software, it is time consuming and difficult to execute and verify all these test cases one by one. It is obvious that automation can help to maximize coverage and reliability. In this chapter, several technologies are presented regarding automated testing, along with how they are all effectively coordinated and the data managed via a Test Management System.

Test case generation

In order to get the most benefit from the test cases and automate the testing process a Test Management

System was developed. The Test Management System was designed to support both manual and automatic test creation methods and to support the reuse of the test cases throughout the whole process. Figure 2 shows test cases written in the Test Management System. In the picture, the upper table is for input and the lower table is for expected responses.

Requirement based test case

To ensure that the functional model is designed according to the requirements, test cases based on the requirements are created manually by a test engineer. Requirement based test cases might be automatically generated by translating requirements into a formal language. However, the most common method is to express requirements in narrative language. The Test Management System can help describe test scenarios easily and generate test cases based on the test case templates.

Automatic test case generation from the model

Automatic test generation has the great benefit of high test coverage and accuracy in model-based development process. Intensive test cases can be generated automatically from a system model. Some modeling tools provide automatic test generation tool kits.

INPUT-VARIABLES

Test_ID	Time Step	Time	Step	PWR_WIN_ENABLED	WIN_BTN_DRV	DRV_WIN_BTN_PASS_IO	D TN
Initial		0	0	0	0	0	
2.1.1	1000	1000	0	0	0	0	
	1000	2000	0	2	0	0	
	1000	3000	0	2	0	0	
Initial	1000	4000	0	2	0	0	
3.2.1	1000	5000	0	2	0	0	
	1000	6000	0	2	1	1	
	1000	7000	0	2	0	0	
	1000	8000	0	2	2	2	
	1000	9000	0	2	0	0	
	1000	10000	0	2	3	3	
	9000	19000	0	2	0	0	
	1000	20000	0	2	0	0	
	1000	21000	0	2	0	0	

OUTPUT-VARIABLES

Test_ID	Time Step	Time	Step	CMD_WIN_BTN_LED_DRV	CMD_WIN_MTR_DRV	DRV_WIN_BTN_PASS
Initial		0	0	0	0	0
1.1.1	1000	1000	0	0	0	0
	1000	2000	0	1	0	0
	1000	3000	0	1	0	0
Initial	1000	4000	0	1	0	0
1.2.1	1000	5000	0	1	0	0
	1000	6000	0	1	1	1
	1000	7000	0	1	0	0
	1000	8000	0	1	2	2
	1000	9000	0	1	0	0
	1000	10000	0	1	1	1
	9000	19000	0	1	1	1
	1000	20000	0	1	0	0

Figure 2. Test cases viewer in Test Management System

Automated testing flows and tool chains

Figure 3 describes the testing flows and tool chains for the proposed automated testing technique. As mentioned before, automated testing consists of test case generation, test execution, and result analysis shown as above. It is important that these steps are connected seamlessly at all test phases. Automatically and manually created test cases are imported into the Test Management System in a generalized format shown in Figure 2. This enables the test results to be automatically evaluated with the Test Management System.

The functional model can be designed in Telelogic's Statemate or Mathworks' MATLAB/Simulink/Stateflow. In order to generate test cases automatically, ATG which is a Statemate plug-in program or T-VEC Tester which is a 3rd party tools in Simulink can be used. The test cases are executed on three kinds of test platforms. The model simulation feature can be used for system model testing. A rapid prototyping system is used for auto-generated code testing and Vector's CANoe for network integration testing.

Static error checking and formal validation should precede test execution. If there are any errors test execution should not be operated as exact test results can not be expected. Statemate has Model Checker/Certifier and MATLAB has Safety-Checker Blockset to help to unveil errors in the model.

AUTOMATIC TEST EXECUTION

System model testing

System model testing is executed to verify if the model is designed according to the requirements. The requirement based test cases are generated by the Test Management System as a file to stimulate the functional model. The model outputs are recorded during the simulation. The Test Management System collects the test results and manages the data.

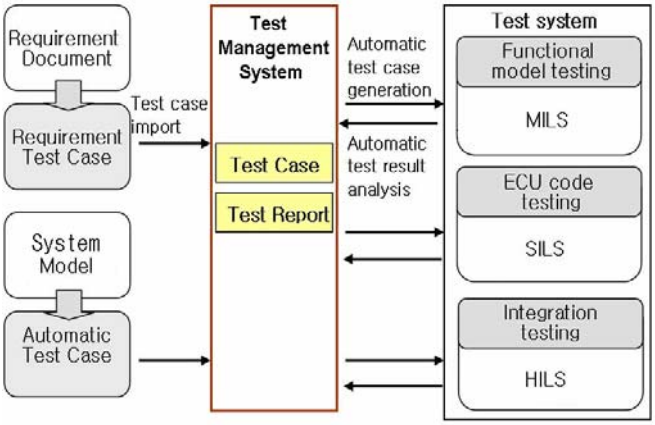

Figure 3. Automated testing flows and tool chains

The responses are compared with the expected results in the Test Management System. If an error is detected, the Test Management System highlights the problem in red. Figure 4 shows an analysis of simulated results. Early system models usually contain errors. Consequently correcting the errors, rerunning the tests and analyzing the results are repeated several times. This process is inevitable, labor-intensive and time-consuming. Automating this process leads to increasing accuracy and reliability as well as reducing testing time and effort.

Automatic Target Execution System

After validating the functionality C code is auto-generated from the models. Due to possible differences in execution time and response values between simulations of functional models running on the PC and the auto-generated code running in a real ECU, the auto-generated code must be validated. The automatic target execution system, namely Target-Practice, provides the rapid-prototyping environment, easy hardware input and output configuration and real-time software testing on a real target to do this validation.

Target-Practice consists of hardware equipment and control software. The hardware comprises of a target ECU and a data acquisition board. Control software is used to map I/O signals of the functional model to the real ECU I/O as well as control, monitor and test the model running on the target. Figure 5 shows a photo of Target-Practice.

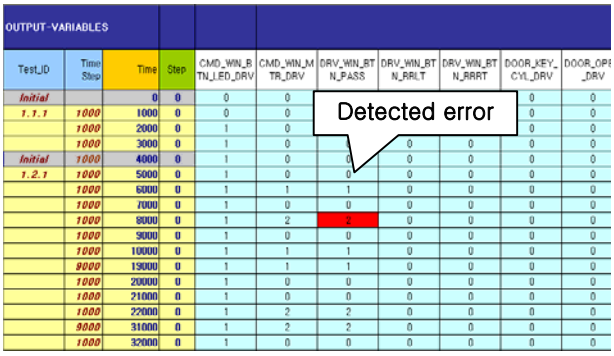

Figure 4. Analysis of Simulated results

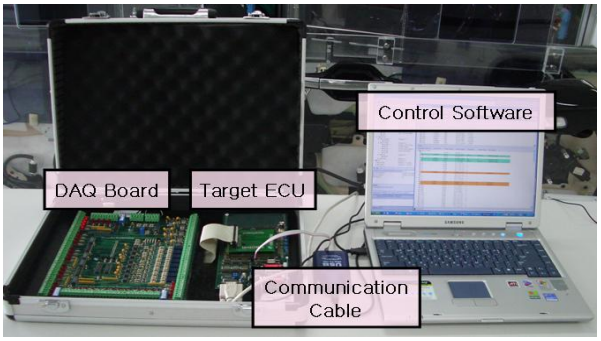

Figure 5. Target-Practice

The automated testing facility within Target-Practice is designed for model-based development processes. Test cases created while simulating the model in Statemate can be rerun on the Target-Practice rapid-prototyping hardware. This enables the model to be validated against the real hardware using the same tests as on the PC. The simulated results are deemed the expected results and are automatically compared to the rapid-prototype results so that faults can be instantly detected. To further extend the process to include all test cases the Test Management System was configured to create test cases in the correct format for Target-Practice. Figure 6 shows validating results by Target-Practice.

Network integration testing

Automotive network communication tests are performed to ensure that the ECUs are sending CAN messages and data as specified. Each ECU should send a certain number of messages on the bus with specified time intervals. Network test and analysis for the CAN bus system is performed with CANoe. CANoe supports sequential test procedures described in a proprietary language called CAPL (CANoe Application Programming Language). A test report is automatically written during execution of a test.

The application software for each ECU is generated from the model. The model I/Os are mapped to network signals so the application can communicate on the bus. The test cases in the Test Management System are automatically reconfigured into CAPL files for automated testing of network communication and functionality. The CAPL file for automated testing consists of `MainTest`, `Testcase` and suporting `Functions`. `MainTest` calls test cases, and writes test information into the report file. `TestCase` is composed of individual test steps which create test results. `Functions` are used as storage for input signals and expected responses. Figure 7 shows CAPL files automatically generated by the Test Management System. Test report contains the verdict of each test step. So if an error is detected, the time and signal of the error can be easily checked.

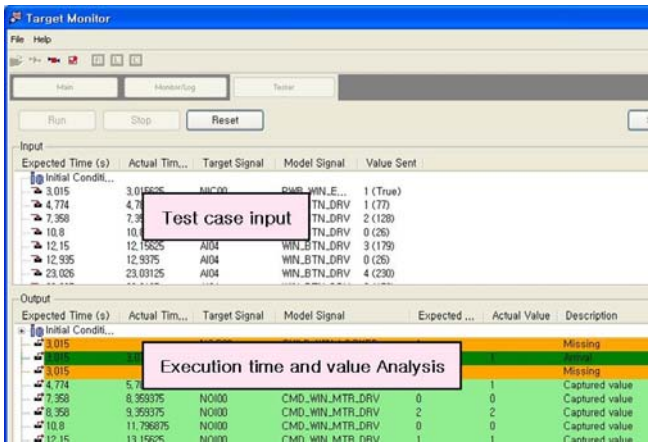

Figure 6. Analysis results of ECU code testing

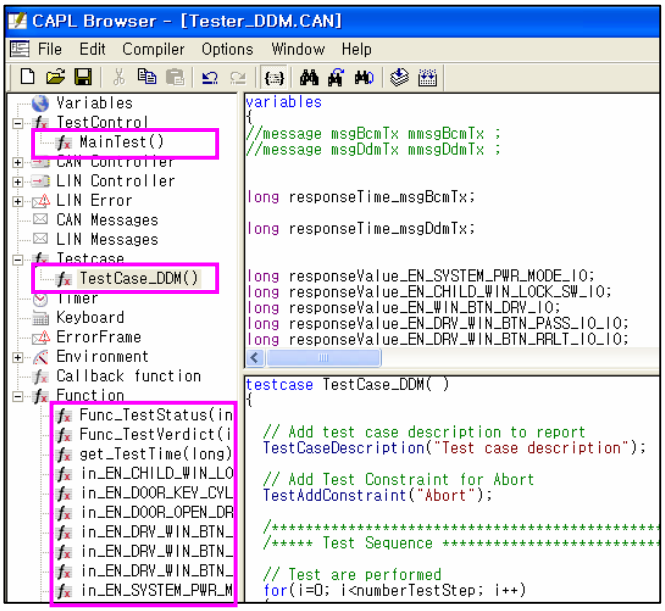

Figure 7. A CAPL file from Test Management System

CASE STUDY – WINDOW LIFT SYSTEM

The proposed test automation techniques for automotive embedded systems are applied to the window lift system. A functional model of the system-under-design was developed using Telelogic's Statemate. Test cases were automatically created by ATG (Automatic Test Generation, a Statemate add-on tool also available from Telelogic) which automatically generates test cases from a Statemate model. The generated test cases were imported into the Test Management System. The window lift system consists of 4 door modules and a BCM (Body Control Module) and each ECU is connected by CAN/LIN network. Figure 8 shows the physical architecture of a window lift system and the functional decomposition of the DDM (Driver Door Module).

TEST CASE GENERATION

Test cases are automatically created from the Statemate model with ATG and manually created based on the requirements. ATG creates two files; a test case file and an accompanying report file detailing test coverage information. About 2,000 test cases are created for the window lift system and these generated test cases and expected results are automatically imported into the Test Management System. Figure 9 shows the user interface of the Test Management System for created test case import and generation of each test phase.

FUNCTIONAL MODEL TESTING

The functional model of a whole window lift system and 5 ECU modules are tested. All functional requirements are satisfied. A file containing test cases is created with the Test Management System and the generated file executes Statemate simulations automatically and the simulated responses are saved to a file. The saved outputs are compared with the expected results from the Test Management System.

SOFTWARE TESTING

After the model validation is complete, ECU software of the window lift system is auto-generated and downloaded into the rapid-prototyping hardware, Target-Practice, to test performance and functionality of the software before hardware is built. This test is separately performed for the 5 ECUs.

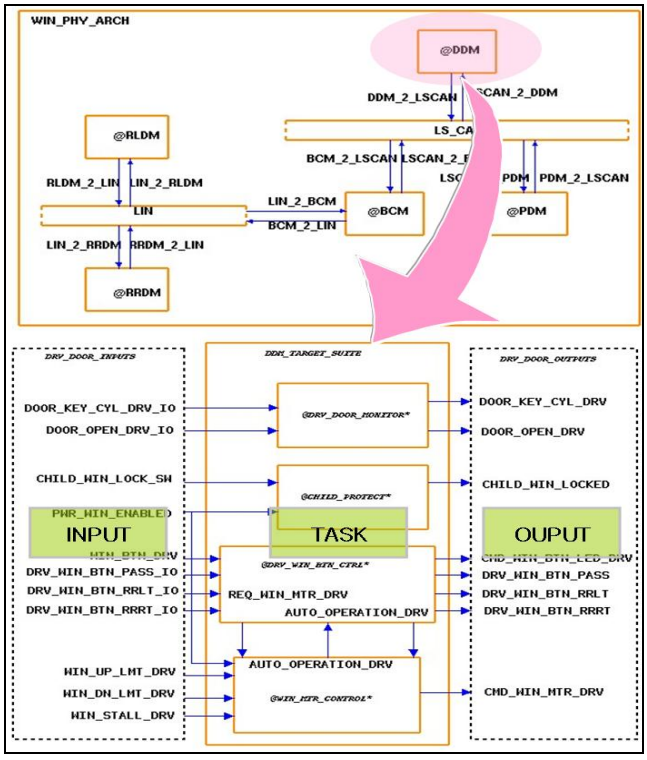

Figure 8. Functional model of window lift system

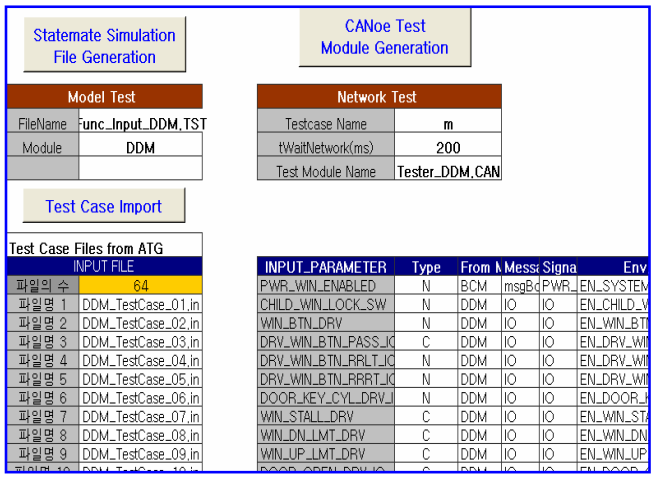

Figure 9. User interface of the Test Management System for test case import and test case generation

NETWORK INTEGRATION TESTING

Integration tests are performed to see how different individual systems work together in an automotive network system. Interval time, busload, and transmission and receiving of signals satisfy the specifications for a window lift system. All five network nodes are tested separately. Figure 10 shows a network system of a window lift system and testing environment.

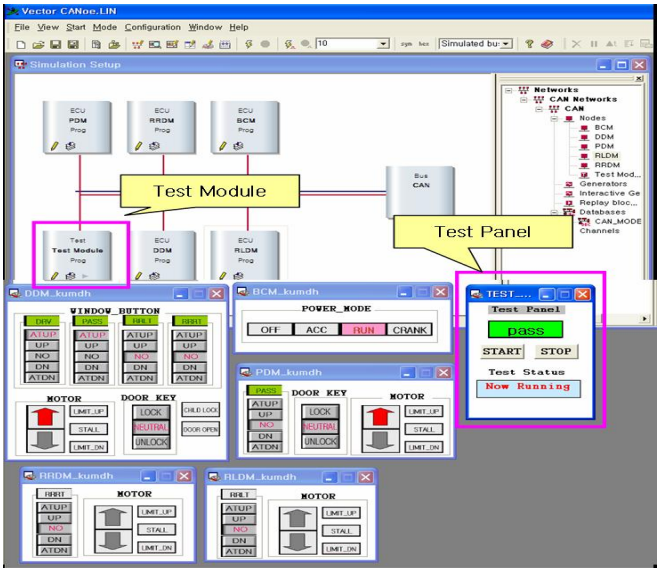

Figure 10. CANoe Simulation for network testing

CONCLUSION

The model based development process is becoming popular in the automotive industry because of its benefits such as low cost and short time to market. Automated testing technologies are needed to accelerate the benefits of the model based method. In this study, automated testing technologies are presented for all test processes. A case study of the window lift system is introduced to apply this automation process.

A Test Management System is developed through this study, which enables us to generate test cases and analyze test results automatically through entire test process. In addition, Target-Practice is employed for auto-generated ECU code testing. The error detection process is a simple and repeated process and it is time consuming. Adaptation of the proposed technologies can reduce the time and effort of testing dramatically. Furthermore, test reliability and consistency could be improved.

REFERENCES

1. J. Son, I. Wilson, W. Lee and S. Lee, "Model Based Embedded System Development for In-Vehicle Network Systems", *SAE*, 2006-01-0862, 2006.
2. Y. Dong, M. Li and R. Josey, "Model Based Software Development for Automotive Electronic Control Units", *SAE*, 2004-21-0038, 2004
3. R. Weber, K. Thelen, A. Srivastava and J. Krueger, "Automated Validation Test Generation", *Digital Avionics Systems Conference of IEEE*, Page 99-104 ,1994.
4. R. W. Butler, "An Introduction to Requirements Capture Using PVS: Specification of a Simple Autopilot", *NASA Technical Memorandum* 110255, 1996
5. C. L. Heitmeyer, R. D. Jeffords and B.G. Labaw, "Automated Consistency Checking of Requirements Specifications", *ACM Transactions on Software Engineering and Methodology*, vol. 5, No. 3 , Page 231-261, 1996.
6. Y. G. Kim, H. S. Hong, D. H. Bae and S.D.Cha, "Test Cases Generation from UML State Diagrams", *IEE proceedings,* online no. 199990602, 1999
7. N. H. Lee and S. D. Cha, "Generation Test Sequences from a Set of MSCs", *The International Journal of Computer and Telecommunications Networking*, Volume 42 , Issue 3, Page 405 - 417, 2003
8. R. L. Probert, H. Ural and A.W.Williams, "Rapid Generation of Functional Tests using MSCs, SDL and TTCN", *Computer Communications*, Volume 24, Issues 3-4, Page 374-393, 2001
9. D. L. kaleita and N. Hartmann, "Test Development Challenges for Evolving Automotive Electronic Technologies", *SAE*, 2004-21-0015, 2004.