Automated Testing for Automotive Embedded Systems

Dae-Hyun Kum[1], Joonwoo Son[2], Seon-bong Lee[3] and Ivan Wilson[4]

[1] Department of Mechatronics, Daegu Gyeongbuk Institute of Science & Technology, Daegu, Korea
(Tel : +82-53-430-8455; E-mail: kumdh@dgist.ac.kr)

[2] Department of Mechatronics, Daegu Gyeongbuk Institute of Science & Technology, Daegu, Korea
(Tel : +82-53-430-8453; E-mail: json@dgist.ac.kr)

[3] Department of Mechatronics, Daegu Gyeongbuk Institute of Science & Technology, Daegu, Korea
(Tel : +82-53-430-8450; E-mail: mecha@dgist.ac.kr)

[4] DsysD Ltd, Nottingham, England
(Tel : +44-115-9881211; E-mail: ivan@DsysD.com)

Abstract: The automotive embedded system becomes even more complex and occupies more portion of vehicle price lately. Therefore, automotive industries apply a model-based approach for the embedded system development. The model-based approach improves quality and reduces cycle time by using modeling and its ability to simulate to introduce early validation of requirements. This paper describes automatic test generation and execution techniques for automotive embedded systems to maximize the early validation capabilities of model-based development process. The proposed techniques can cover from the functional high level model validation to the auto-generated software validation. For the functional model validation, a virtual prototype is used on the PC environment. A rapid prototype and target execution and monitoring system, Target Suite, is developed for the generated software validation. We also develop an automatic testing and analysis environment to manage the test executions and the corresponding results. The proposed model-based automated validation techniques will go far forward to improving the reliability and cost-effectiveness of the products

Keywords: Automated testing, Model-based development process, Automotive Embedded system

1. INTRODUCTION

The Automotive embedded system becomes even more complex and occupies more portion of vehicle price lately. 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 process such as model-based approach [1-2]. But to meet demand for high quality and lower cost, it is becoming increasingly necessary to rely on a new test systems and methodologies in all phases of the product life cycle.

Erroneous automotive embedded software may cause serious car accident related to human safety. Therefore, it is essential to detect any errors in the embedded software through various test processes. The embedded software is often subject to change to meet customer's needs and its functionalities, so that it is desirable to automate testing to improve efficiency of process.

Test automation consists of the following three steps: test case generation, test execution and result analysis. Many research efforts have been made at these automation techniques. During the requirement capturing phase, automated testing can be realized by introducing formal language [3-5]. 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 [6-8], and automation concerning test system and process has been presented [9]. However, research on seamless test automation based on model-based approach has not been applied.

In this paper, several techniques are presented that involves seamless test automation for automotive embedded systems. The proposed approach can cover from high level model validation to the auto-generated code validation in a real ECU(Electronic Control Unit)

Functional model is developed by using the I-Logix Statemate. Test cases are automatically generated by ATG(Automatic Test Generation) which allows automatic test generation out of Statemate design. A rapid prototype and target execution system, Target-Suite is developed for validation of auto-generated code from the behavior model. In addition, Test-Generator and Test-Analyzer is developed to automate and improve test process.

2. TEST PROCESS

2.1 Model-based development process

Model-based development process(MBDP) is introduced to improve quality and reduce cycle time. Fig. 1 describes the Model-based development process that follows typical V-process.

Development is started with requirement capturing and analysis. The success of any product development is depending on the creation of clear, complete and unambiguous requirements. Functional model should be created according to the requirement capturing document. The vehicle electronic system is too large to be addressed at a time. So it is often broken down according to the functional groups. Once the main sub-system has been identified, it is ensured that signals for input and output of the system are clearly defined. Virtual prototype of functional model enables us to test and validate the software functionality without hardware in the early stage. When behavior model is released,

ECU software development and network system development are started at the same time.

Once the system design and ECU design have been validated on virtual environment, the designs should be moved into the implementation phase. To generate code for embedded applications, all the software designs, implementation details and constraints have to be imposed into the model such that the generated code can satisfy all the requirements. Finally all source codes including generated application code, I/O drivers, communication kernel and operating system code are integrated.

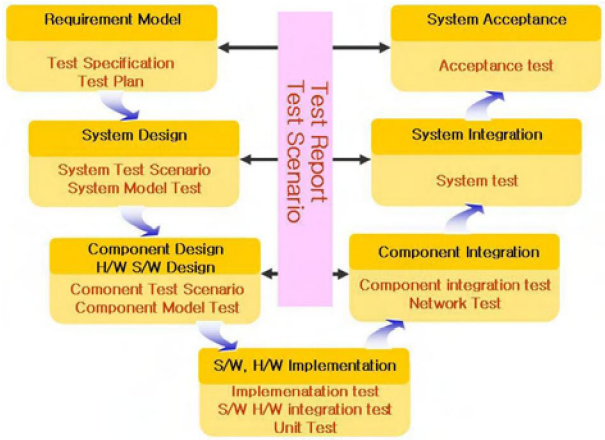

Fig. 1 Model-based test and development process

2.2. Model-based test process

Test process also follows model-based development process, but testing should be performed separately to the design. Fig. 1 describes model-based test process linked to development process.

Clear understanding of the system requirements should be preceded before getting down to testing. Exact I/O signals to be tested and test specification are defined by analysis of overall system structure, functional behavior so that test result is reliable and automated testing is performed efficiently.

Test cases which are simple, atomic and formal should be prepared based on the test requirements. The reason is that, complex test cases can make errors when executing the test case.

A test system capable of automatic testing is needed. Depending on the development process and the system under test, different test systems are chosen. The following table 1 shows a selection of test techniques.

Table 1 Test phases and techniques

Test phase	Test technique
Requirement testing	Model in the loop
Functional model testing	Model in the loop
Implementation testing	Software in the loop
Integration testing	Hardware in the loop

After test execution, the test results should be analyzed to unveil the error that caused a test to fail and discovered errors are needed to be fixed. Test results and related information should be documented throughout all test processes.

3. TEST AUTOMATION TECHNIQUES

3.1 Overview of test automation

Since there are huge numbers of test cases to execute in automotive embedded software, it is obvious that Automation is required to maximize coverage and reliability.

In this paper, several techniques are presented regarding automated testing that can cover requirements testing, functional model testing and auto-generated code testing. Fig. 2 describes the test phases and corresponding tool chains for the proposed techniques. Usually automated testing is consists of test case generation, test execution, and result analysis shown as following figure. It is important that these steps are connected seamlessly at each test phase

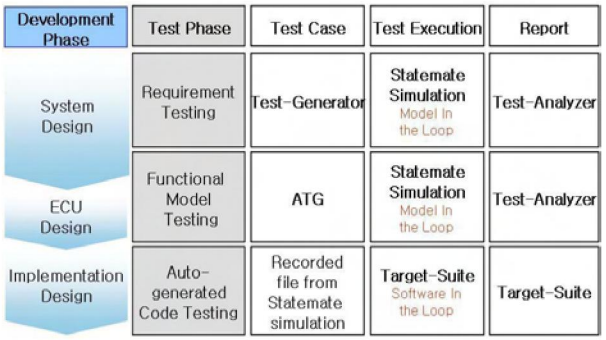

Fig. 2 Tool chain for the test automation

Test-Generator, Test-Analyzer and Target-Suite are developed for seamless test automation in this paper. Test-Generator allows test engineer to generate test cases that can be executed automatically in the Statemate simulation. Fig. 3 shows Test-Generator interface that is developed using Excel and Visual Basic language. If time, steps, input signals and values are written in the Excel cells and push GENERATION button, a file containing test cases is generated automatically.

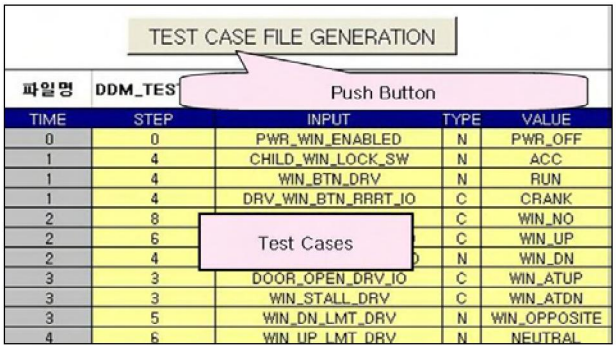

Fig. 3 Test-Generator

Since usually test cases are from hundreds to millions, it is time consuming and not easy to verify numerous test cases one by one. In this research, Test-Analyzer is developed in order to analyze test result automatically. It compares test results from the simulation of the behavior model with expected results. Fig. 4 shows analysis result by Test-Analyzer. If an error is detected, the cell is filled with orange color. Target-Suite will be introduced for the auto-generated code testing in chapter 3.4 in detail.

		OUTPUT-VARIABLES (PASS FAIL)					
Time	Step	DOOR_KEY_ CYL_DRV	CMD_WIN_ MTR_DRV	DRV_WIN_ BTN_PASS	DRV_WIN_ BTN_RRLT	DRV_WIN_ BTN_RRRT	DOOR_OPEI DRV
0	4	0	0	0	0	0	0
1	8	2	0	2	2	2	1
2	11	2	0	4	4	4	1
3	15	1	0	4	4	4	1
4	20	1	1	4	4	4	1
5	23	1	1	4	4	4	1
6	27	1	1	4	4	4	1
7	30	1	1	4	4	4	1
8	33	1	1	4	4	4	1
9	37	1	1	4	4	4	1
10	40	1	1	4	4	4	1
11	43	1	1	4	4	4	1

Detected Error

Fig. 4 Test-Analyzer

3.2 Requirement testing

Requirement testing is performed to ensure that created behavior model is consistent with the requirements. Requirement based test cases might be automatically generated by translating requirements into formal language, however it is known that the most reasonable method is to express requirements in narrative language. In this study, test cases are generated manually by test engineer with Test-Generator, generated test cases are executed in the Statemate simulation and the results are analyzed with Test-Analyzer.

Fig. 5 shows test cases generated by Test-Generator based on the functional requirements. Generated test cases are executed automatically in the Satemate simulation. And then test results are easily evaluated with Test-Analyzer.

```
Project Name: WIN_TEST_01
Work Area Directory: d:/10_Working/MODERATO_KUMDH/wa_win_test_01
Profile Name: test_ddm_102
Date/Time Produced: Wed Apr 19 09:19:10 2006
Recorded time mode: Relative
Recording starts at simulation time: 0.000000

#Data Section:
0.000000 0 DDM_TARGET_SUITE:DOOR_KEY_CYL_DRV_IO N 0
1.000000 4 DDM_TARGET_SUITE:DRV_WIN_BTN_RRLT_IO N 2
1.000000 4 DDM_TARGET_SUITE:DRV_WIN_BTN_PASS_IO N 2
1.000000 4 DDM_TARGET_SUITE:DOOR_OPEN_DRV_IO C 1
2.000000 8 DDM_TARGET_SUITE:DRV_WIN_BTN_RRLT_IO N 4
2.000000 8 DDM_TARGET_SUITE:DRV_WIN_BTN_PASS_IO N 4
2.000000 8 DDM_TARGET_SUITE:PWR_WIN_ENABLED C 0
3.000000 11 DDM_TARGET_SUITE:DOOR_KEY_CYL_DRV_IO N 2
3.000000 11 DDM_TARGET_SUITE:DRV_WIN_BTN_RRRT_IO N 0
4.000000 15 DDM_TARGET_SUITE:DOOR_KEY_CYL_DRV_IO N 2
4.000000 15 DDM_TARGET_SUITE:WIN_BTN_DRV N 3
4.000000 15 DDM_TARGET_SUITE:DRV_WIN_BTN_RRRT_IO N 0
4.000000 15 DDM_TARGET_SUITE:PWR_WIN_ENABLED C 1
```

Fig. 5 Test cases generated by Test-Generator

Static error checking should be preceded prior to test execution. If there is any static error, test execution may not be operated and exact test result can not be expected. Check model from the Statemate menu helps to unveil static error in the model.

3.3 Functional model testing

In order to generate test cases automatically uses ATG which is Statemate plug-in program. Test stimuli and the expected responses are computed by based on selected activity charts and test goal is selected to cover outputs, states or transition. Two kinds of files are obtained after ATG execution. One is test cases which can be executed in the Statemate simulation. The other is report file including test coverage information.

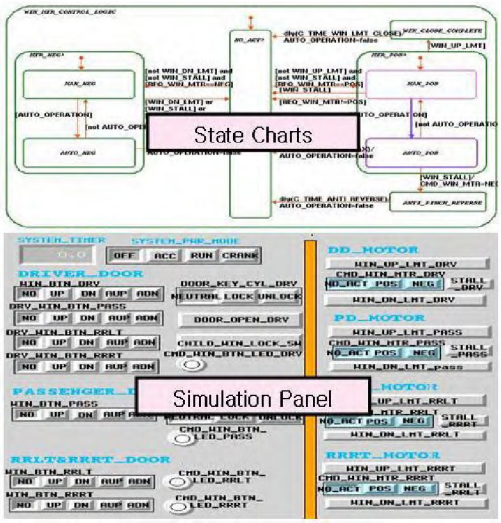

Fig. 6 Model in the loop simulation in Statemate

Statemate simulation executes generated test cases automatically. Stimuli and simulated output values are recorded into files during the simulation. The simulated responses are compared with the expected result with Test-Analyzer. If an error is detected, testing should be performed step by step to find cause of an error. Fig. 6 illustrates error fixing process using virtual panel and state charts.

Since early functional model may contain errors, in order to correct these errors, generating test cases, test execution and analysis result are repeated several times. This process is inevitable but labor-intensive and time-consuming. Automating this process leads to increasing correctness and reliability and reducing time and efforts on testing.

3.4 Auto-generated code testing

After completing validation of behavior model, C code is generated automatically by Statemate. There might be differences in task execution time and values between simulation of behavior model running in PC and auto-generated code running in a real ECU environment. Target-Suite provides automated testing environment for the auto-generated code.

Target-Suite consists of hardware equipment and control software. Hardware comprises of two main boards: one is target board and the other is data acquisition board. And there are two control software programs. One is Target-Practice which can map I/O signals of the functional model with a real ECU model. The other is Target-Monitor which analyzes and shows the execution result on PC monitor. Fig. 7 depicts the picture of Target-Suite.

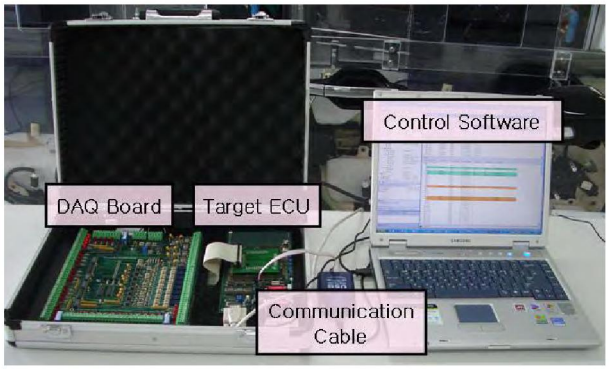

Fig. 7 Target-Suite

To test generated application code running in ECU environment, Statemate I/O signals should be mapped to ECU I/O ports. When generating code, these mapped I/O signals are considered automatically. Fig. 8 shows signal mapping from functional model to target ECU by drag and drop.

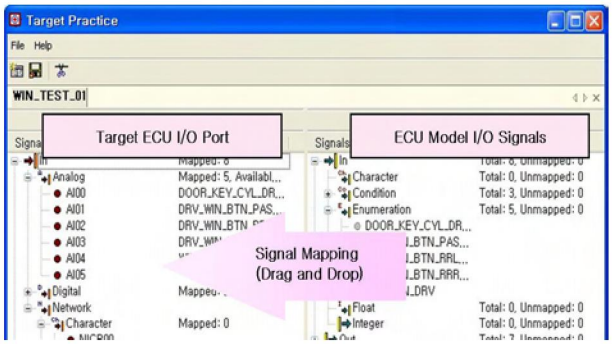

Fig. 8 Target-Practice

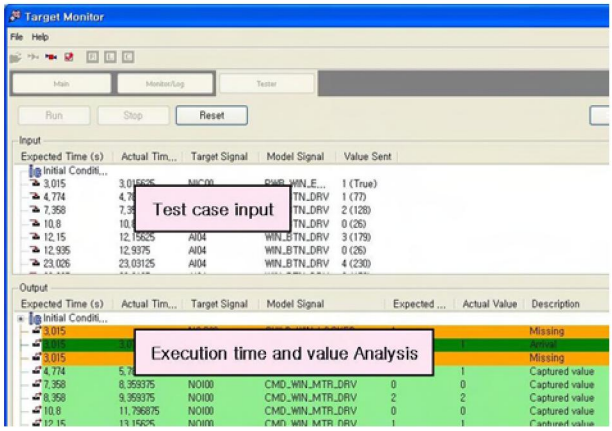

Fig. 9 Target-Monitor

Target-Monitor analyzes and compares execution time and values between different environments and shows the test results on PC. Fig. 9 shows validating results by Target-Monitor.

4. CASE STUDY

Proposed test automation techniques for automotive embedded systems are applied to window lift system.

4.1 window lift system

The functional model of window lift system is designed by Statemate. This system consists of 4 door modules and BCM(Body Control Module) and each ECU is connected by CAN/LIN network. Fig. 8 shows physical architecture of window lift system and activity charts of DDM.

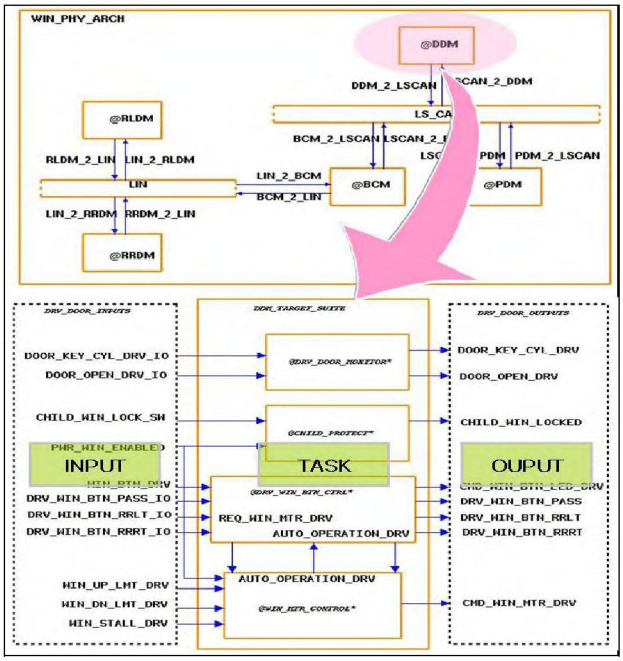

Fig. 10 Functional model of window lift system

4.2 Requirements testing

According to the functional requirements, about 50 test cases are generated for each ECU model with Test-Generator. Statemate simulation executes generated test cases automatically and Test-Analyzer analyzes the simulated result in order to verify the model.

4.3 Functional model testing

ATG was used to generate test cases automatically in order to validate each ECU model. Test goal is set to cover all states and transitions and about 300 test cases are generated for each ECU model.

Generated test cases are automatically executed in the Statemate simulation and the simulated responses are analyzed by Test-Analyzer which compares with the expected result form ATG.

4.4 Auto-generated code testing

After the model validation is complete, software of each ECU is auto-generated and downloaded into Target-Suite to test performance and functionality of the software before hardware is built. Recorded file during Statemate simulation is used for the stimuli to the Target-Suite. Execution time and response value is validated using Target-Monitor.

5. CONCLUSION

In this research test automation techniques are proposed for functional model and auto-generated code running in target ECU. Case study of window lift system is introduced to apply this automation process.

Erroneous automotive embedded system may result in serious accident or problem related to human life. Error detection process is simple and repeated process that is time consuming. Adaptation of proposed techniques can reduce time and efforts on testing dramatically. Furthermore test reliability and consistency could be increased.

In the future study, test automation process is expended to network and integration test to realize test automation through the entire process.

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.