Virtualization for Testing in Model-driven Distributed System

Youngheum Kim
Control System Research Team
Hyundai Carnes Co. Ltd.
Uiwang-si, Gyeonggi-do, Korea
yhkim@carnes.co.kr

Seungyong Lee
Control System Research Team
Hyundai Carnes Co. Ltd.
Uiwang-si, Gyeonggi-do, Korea
seungyong.lee@carnes.co.kr

Seungbeom Kim
Control System Research Team
Hyundai Carnes Co. Ltd.
Uiwang-si, Gyeonggi-do, Korea
seungbeom.kim@carnes.co.kr

Abstract—**Most automotive companies are facing key challenges of improving quality and reducing time to market with limited resources. One of the methods for achieving key challenges is virtual simulation test in early development phase. The methodology of creating a model-driven distributed system with communication database as well as automatically generating models from specifications is proposed. In order to detect faults of functions and inconsistency of communication signals between specifications and vehicle network, the manual and automated test methods which can verify a model-driven distributed system is represented. This was applied successfully in the production project. The usefulness of the developed methodology and the tool is well ensured through the discovery of defects.**

Keywords-Model-based; Model-driven; Virtual; Distributed; Automated Test

I. INTRODUCTION

The number of Electronic Control Units (ECUs) in recent premium vehicle has increased to more than 70 ECUs [1]. This was accompanied by an increasing degree of distribution with lots of complex functions. Furthermore, the usage of various vehicle networks has risen significantly in recent years. Due to the increasing distribution and interaction of functions, interoperability of functions on the vehicle network as well as maturity of each single function should be ensured. The latest distributed design adopts security check and handshakes between ECUs which adds complexity to the vehicle systems [2].

Such a trend in vehicle system design presents new technical challenges for the design verification and the product validation. Especially, implementation of ECUs without identifying design flaws not only makes the development cost much higher but also affects product quality and timely delivery. One study indicates that 60% of defects are introduced in the specification while 55% of these defects are not detected until the testing phase of real ECUs, which results in inefficient specifications debugging by OEM test engineers or supplier engineers [3]. Thus, many embedded systems development organizations are adopting model-based design for early design analysis through model simulation.

Model-based design methodology mostly has been used for not only developing advanced new features in preceding division which isn't directly linked to development for mass production but also verifying early phase control algorithm with a plant in power-train division which mainly develops a centralized system. Relatively, the distributed system implemented by various suppliers with a lot of interoperation has not actively adopted model-based design. The distributed system configuration gets a specific feature depending on projects and sometimes changes a lot during the development lifecycle. Moreover, the complexity of interaction among the ECUs on vehicle networks has significantly increased in recent years. Therefore, adopting model-based design for the distributed system verification provides much opportunity to demonstrate savings in terms of correction cost due to the system architecture or specification errors [4].

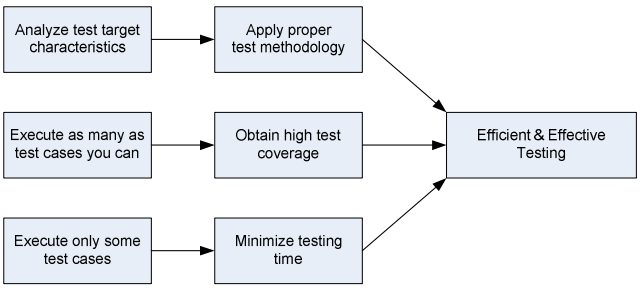

Figure 1. Efficient and effective test design

The issues and challenges described above will be presented in this paper. Methods of specification, modeling and

978-1-4673-0990-5/12/$31.00 ©2012 IEEE

configuration of model-driven distributed system are introduced with some automatic generation procedure. Manual and automated testing methods in terms of an efficient and effective testing as shown in Figure 1 are proposed in order to eliminate design uncertainty in early development phase [5].

Manual test can be employed with small efforts, whereas automated test requires a significant up-front investment for building the automated test suite. Adopting a test method between manual test and automated test should be considered depending on characteristics of the project because automated test takes a long time for initial setup but gets a high turnover eventually. There is the number of about 63% of failed test automation projects [6]. On one hand, there is a high potential gain; On the other hand, there is a high probability of loss of investment. Organizations should prepare to identify automation opportunities, select the right test automation framework and thereby increase the return on investment [7]. This paper shows an automated test approach with test tool, test method and test procedure for verifying model-driven distributed system. This can be available seamlessly to test the functions of real ECUs on the network with minor test configuration changes throughout the entire product development cycle.

II. GENERATION OF MODEL-DRIVEN DISTRIBUTED SYSTEM

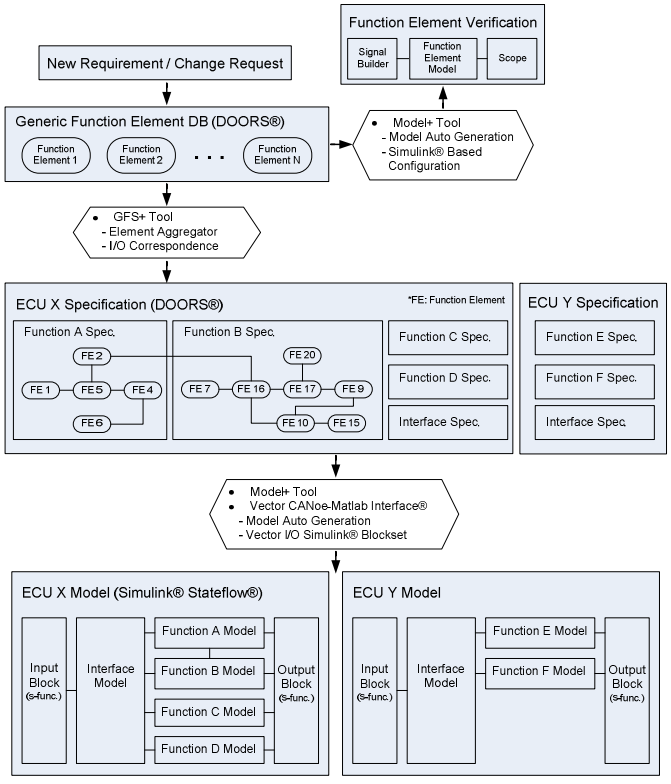

Figure 2. Generation procedure of ECU specification

GFS+ tool provides the users with a simple and easy way to define specific project's ECU specifications aggregating a set of generic function elements which can be partitioned to any ECU, improving reusability among projects. Each generic function element is developed by new requirement as well as maintained by change request. A newly developed or changed generic function element can be verified by a model and test harness with which are generated by Model+ tool. Verification of each generic function element is mainly carried out by function elements design engineer using the capability of signal builder and scope block in Simulink®. Model+ tool can automatically generate ECU models from ECU specifications as well. Overall procedure for generating ECU models is illustrated in Figure 2.

This procedure presents that the ECU specification for supplier implementation can be synchronized with the ECU model for design verification during project development lifecycle. What an engineer creates and updates the ECU model manually from the ECU specification provokes additional design faults inside models and induces the lack of engineering resources because of creation, correction and configuration management in terms of models. Model+ tool enables engineers to generate an ECU model from any baseline of ECU specification whenever they want.

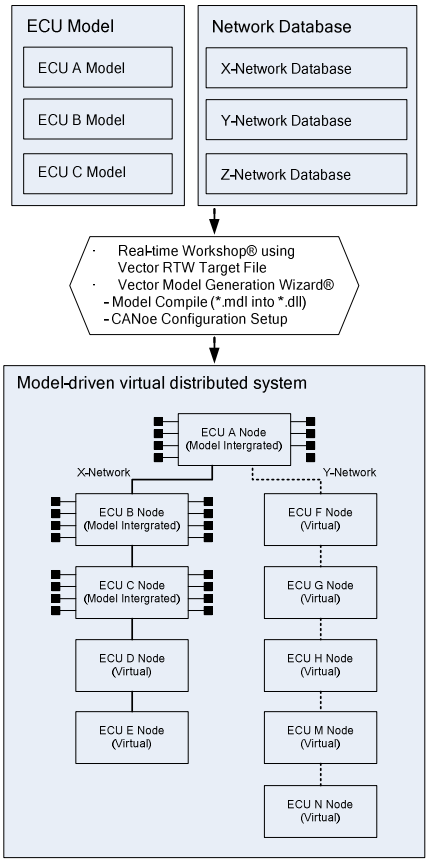

Figure 3. Generation procedure of model-driven distributed system

Each ECU model is a system element which is distributed on the virtual vehicle network as depicted in Figure 3. The virtual vehicle network has to be configured prior to integrating ECU models. When the vehicle architecture is setup, several kinds of communication and their nodes on the project specific

network are defined as a shape of communication database. The virtual distributed network can be easily configured based on the related communication databases utilizing Vector Model Generation Wizard®.

Usually an ECU has the amount of physical input and output ports as well as communication input and output signals. When ECU models are automatically generated using Simulink® block set supported by Vector CANoe-Matlab Interface®, individual physical input or output of ECU specification is mapped into a environment input or output block with the correspondent port name. Individual communication input or output signal is mapped into a signal input or output block with the correspondent signal name in addition. Afterwards, a desired ECU model should be compiled as a file type of dynamic link library through Real-time Workshop® and then can be integrated into the pre-configured virtual distributed network.

Design engineers or test engineers in OEM and implementation engineers in suppliers are generally charge of developing one or several ECUs. Thus, interesting ECU models can be integrated into the virtual distributed network. This specific project's model-driven distributed system is operating based on CANoe®. ECU's functional behavior as well as communication signal transmission can be checked as a real time rate like testing the real distributed system in vehicle.

Anyone can create a model-driven distributed system whatever methodology is used. The key point is how effective and efficient methodology is used for the dedicated engineers. In this paper, in-house developed tools as well as 3rd party tools for minimizing manual activities are employed and single source engineering database for synchronizing between specifications and models is applied.

III. TESTING OF MODEL-DRIVEN DISTRIBUTED SYSTEM

The main purpose of functional test of model-driven distributed system is verifying both function specifications and communication databases are complete, well-defined and ambiguity-free so that implementation engineers in suppliers fully can start the detailed design and coding with confidence. In general, a test harness configuration with signal builder and scope block in Simulink® is widely used for verifying one model itself. However, this isn't applicable to a model-driven distributed system which several complied models are operating in parallel with active network. There are several test methods with a specific test harness to verify a model-driven distributed system. A proper test method can be selected considering to characteristics and scope of the desired project.

A. GUI Based Manual Test

Using graphic user interface (GUI) for virtual test has been utilized a lot for the context of hardware in the loop simulation. Although a considerable amount of setup resources for GUI configuration, GUI enables engineers to test intuitionally the target under test on similar environment of vehicle by support of controlling and monitoring components.

Whenever a new project launches or new features need to be added in existing project, GUI has to be newly configured or changed. In order to test model-driven distributed system proposed in this paper, environment variables or communication signals have to be linked to GUI component using a panel editor in CANoe® as shown in Figure 4. GUI functionality of other tools which is possible to interface with CANoe® can be used likewise.

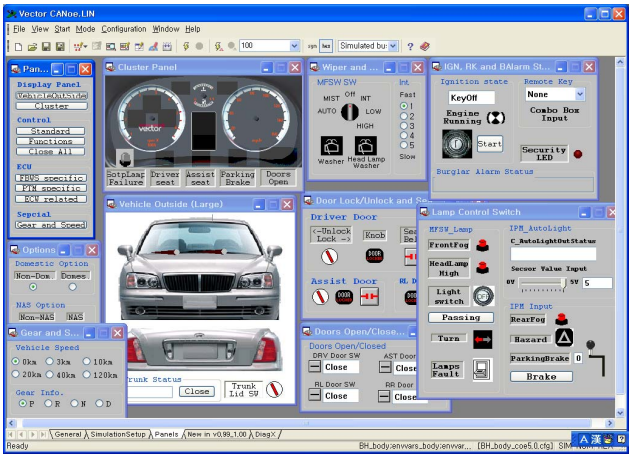

Figure 4. Manual test with GUI

B. Non-GUI Based Manual Test

Although a manual test with GUI is convenient to engineers, it takes much time to not only generate GUI components but also link variables of model-driven distributed system into the property of GUI components. Non-GUI test harness can be used to overcome the drawback mentioned above. Non-GUI based test harness is easily configured compared to GUI based test harness.

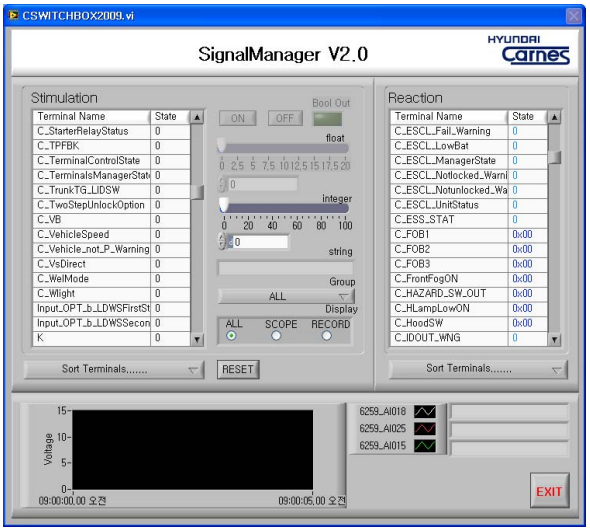

Figure 5. Manual test without GUI

SignalManager tool as shown in Figure 5 built in connectivity with CANoe® is developed based on Labview®. The environment variables and communication signals can be

interacted with model-driven distributed system should be defined at the main window in SignalManager tool, and then model-driven distributed system can be tested by triggering inputs or monitoring outputs at the running mode window in SignalManager tool.

C. *Automated Test*

A well coordinated test automation framework can help find critical defects faster. One of the goals of automated test is to find regression defects sooner than later. Regression defects either are the by-product of correction or may be associated with any new feature implementation. If automated test is utilized just for verifying a model-driven distributed system in the short term throughout the entire product development cycle, test productivity will be decreased. The benefits of automated test can be increased while regression testing is running repeatedly. The proposed automated test method can be applied to not only verify a virtual system but also validate a real system with minor changes in test environment, so high return of investment can be obtained. Moreover, test harness and test scripts for automated test can be firmly developed in advance before the prototyping phase.

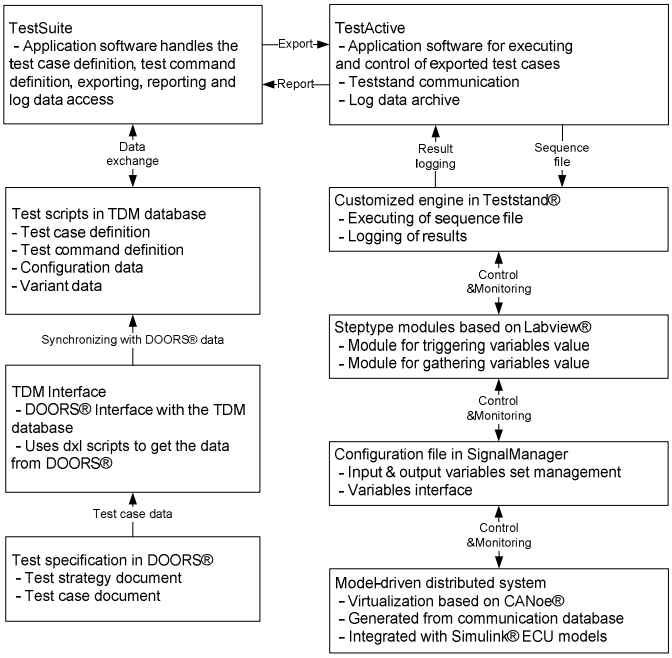

Figure 6. Test automation procedure with test environment

Automated test requires a well defined procedure with technically easy-to-use test environment. Figure 6 represents an overview of the test procedure as described in this section. Test environment has several layer separations which enable test configuration engineers to setup backbone of test harness as well as test engineers to concentrate to develop test specification and test scripts without the need for programming skills. Test specifications composed of test strategy documents and test case documents are designed in DOORS®. In order to develop the test strategy documents, all possible test combinations firstly are identified as a shape of list table using test techniques in view of the test coverage, and then optimized test combinations finally are selected through eliminating invalid or unnecessary test conditions. Test case documents are induced by each line of the list tables in test strategy documents. Individual test case in test case documents consists of precondition, procedure and expected result. Test strategy documents are linked with function specifications to track down the requirement changes. There is extra traceability between test strategy documents and test case documents.

Test scripts are implemented as a form of test cases and test commands based on test case documents. Test scripts are located in the TDM database which can be edited by a test editor tool, so called, TestSuite. Compatibility between test case documents and test scripts can be maintained utilizing TDM Interface tool. A portion of test scripts in TDM database can be exported into TestActive tool considering to the desired test scope. Test can be executed with result logging in TestActive tool which runs with TestStand® engine and Labview® modules as a background task. Labview® modules developed for interfacing CANoe® can interact with the model-driven distributed system. Input and output variables of the model-driven distributed system have to be assigned in SignalManager tool prior to the test run. Test environment and model-driven distributed system operate independently in real time and don't influence each other.

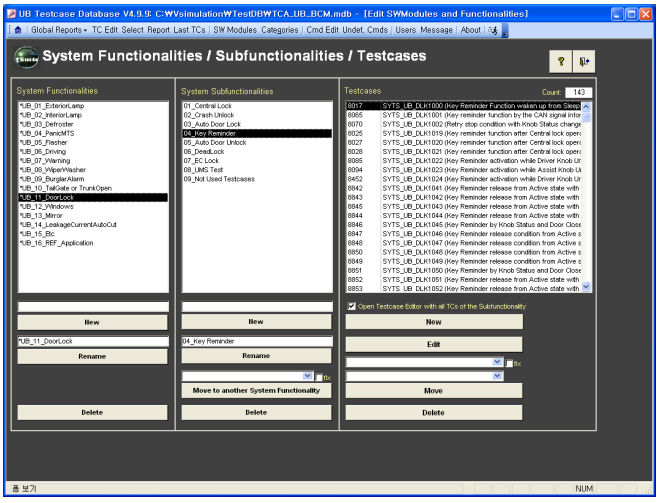

Figure 7. Functional category of test cases in TestSuite tool

TestSuite tool as shown in Figure 7 supports for test engineers to efficiently develop test scripts through concurrent connection into TDM database. TestSuite tool is the application software that handles the TDM database. TDM database contains a large number of test commands and test cases. The set of test cases is managed as a different category according to their functionality. Each test case is composed with combination of test commands as test steps. Each test command can not only trigger or measure specific environment variables and communication signals but also be designed to evaluate a pass or a fail. Test cases can be exported according to regional or optional variants using filter function of

TestSuite tool. Test engineers can easily check and analyze log data of test results in TestSuite tool after execution of exported test cases in TestActive tool.

TestActive tool as shown in Figure 8 is responsible for test execution and test log archive. TestActive tool operates based on a customized engine of TestStand® which is designed to configure conveniently a test automation system through open interfaces. Test cases are exported as a type of sequence file from TestSuite tool, and then sequence files in TestActive tool are executed by calling different kinds of Labview® modules which are defined as customized step types in TestStand®. Project specific configuration file of SignalManager tool contains assignment of input and output variables of the model-driven distributed system. The variables in the configuration file are initialized when a test run starts. After the configuration file is loaded, the background operation to trigger and gather variables is running during the automated test.

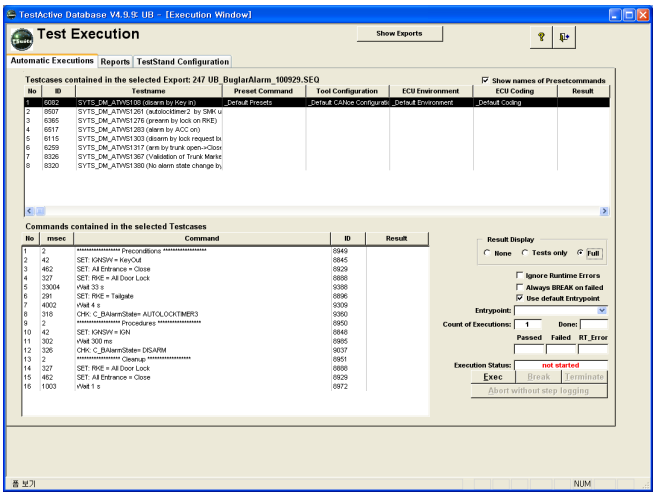

Figure 8. Test execution of exported test cases in TestActive tool

Test scripts in TDM database can be reused for testing the real distributed system in vehicle. While input and output variables of ECU models have to be defined as environment variables in communication database for testing the model-driven distributed system, those of real ECUs have to be replaced as physical ports for testing the real distributed system. The configuration file in SignalManager tool should be changed in order to control and monitor physical ports of real ECUs through the hardware of testing devices. Additionally, ECU nodes integrated with model have to be replaced by real ECU nodes in the configuration file of CANoe®. At last the automated test of real distributed system can be carried out by reusing the most of test cases and test commands in TDM database. This means the effects of test automation can be maximized with limited resources throughout the development lifecycle.

IV. CONCLUSIONS

The ever-increasing complexity of electronic systems in modern vehicles and the rising portion of distributed functions require new ways of developing and testing. The way of configuring a model-driven distributed system through automatically generating models from ECU specifications is described. Several test methods being able to verify the model-driven distributed system are presented. An additional benefit is assured that test specifications and test scripts for automated test can be established in early development phase. We have successfully applied to a production project and found some design defects as follows:

- Unintended interaction of functions on the vehicle network
- Inconsistency and omission of communication signals between ECU specifications and the vehicle network
- Undecided internal variables and interface signals inside ECUs
- Inadequate assignment of either optional variants or regional variants
- Unreasonable functions partitioning into ECUs from the architectural point of view
- Misuse of function elements during the generation procedure of ECU specifications

REFERENCES

[1] S. Furst, “Challenges in the Design of Automotive Software,” BMW Group, 80788 Munich, Germany, EDAA, 2010.

[2] J. Yang, J. Bauman and A. Beydoun, “A Systems Engineering Approach to Verification of Distributed Body Control Applications Development,” SAE Technical Paper 2010-01-2328, 2010.

[3] J. B. Dabney, “Return on Investment of Independent Verification and Validation Study Preliminary Phase 2B Report,” Fairmont, W.V.: NASA IV&V Facility, 2003.

[4] T. Baumann, “Simulation-Driven Design of Distributed Systems,” SAE Technical Paper 2011-01-0458, 2011.

[5] D. D. Grood, TestGoal – Result-Driven Testing, 1st ed.: Springer-Collis, 2008.

[6] B. Lesuer, “Supporting Steps for Successful Test Automation Projects,” SQGNE, 2004.

[7] B. Beersma, “The Future of Automated Testing,” Testing Experience Magazine 16_12_11, pp. 116–118, 2011, Available: http://www.testingexperience.com.

[8] J. Bach, Rapid Software Testing Appendices, Available: http://www.satisfice.com/rst-appendices.pdf.

[9] P. Bret, “Seven Steps to Test Automation Success,” STAR West, 1999.

[10] J. Fetzer, D. Lederer, M. Wernicke and K. -J. Amsler, “Virtual Design of Automotive Electronic Networks – From Function to ECU Software,” Automotive Electronics I/2004, special issue of ATZ, MTZ and Automotive Engineering Partners, 2004.