Test Automation Support Tool for Automobile Software

Tomomi KATAOKA*, Ikuko SAKA, Ken FURUTO, Tatsuji MATSUMOTO

In recent years, automotive components have become more sophisticated and the electronic control unit (ECU) has employed more complex large-scale software. As the product scale becomes larger, an increasing number of tests are required to assure product quality. Even in the case that the auto-testing tools are used, test patterns need to be input manually. This process requires significant amounts of man-hours particularly when the product types vary and the test patterns need to be modified for input signal changes. In order to improve the efficiency of this product development process, we have developed a tool that converts the simulation patterns used in model-based design into those for product tests. This tool automatically adjusts the input signals, and thus, successfully reduces the man-hours by 50% and improves the test quality with common test patterns.

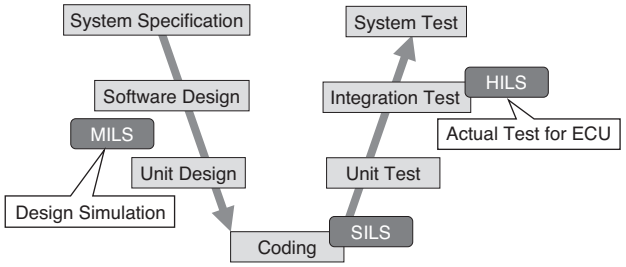
Keywords: software, test method, model based development

1. Introduction

We develop electrical control units (ECUs*1). ECU software (hereinafter referred as "vehicle-embedded software") is less likely to have a completely new design, and in many cases, the former model design is used in the development process of the software. When the former design is used, the former test pattern is also used in order to ensure software quality. However, there is the possibility that the entire design of the vehicle may be reviewed for functional decomposition to multiple ECUs and the design changes of connection interfaces. In this case, test patterns should be modified due to the changes of signal I/O interfaces for each ECU, even though the basic functions of the vehicle are not changed.

On the other hand, the vehicle-embedded software tends to be more complex and larger[1]. To solve the issues of the increase in the number of test items and risks for validation errors, the model-based development method has been promoted in the automobile industry[2]. In this method, design verification is conducted by simulation using a design drawing (model) (MILS*2) (**Fig. 1**). This method enables us to improve the design quality, minimize rework, and detect bugs during the test process[3]. Moreover, by using the common test patterns in the phases of design simulation and ECU testing, the efficiency of the entire development process can be improved. However, signal I/O timing may vary due to the variations of the signal I/O interfaces between the model and ECUs intended for verification (HILS*3). These issues may form a bottleneck in achieving the standardization.

This paper describes how we can improve the efficiency in embedded software development by reusing the former design. In our experiment, we first worked on the technology of using the common test patterns for I/O interfaces of ECUs before and after the design change. We then used the common test patterns at the phases of ECU testing and design simulation, with the technology specified above, to achieve high efficiency in the model-based development.



	MILS	SILS	HILS
ECUs	Model	Model	Actual Unit
Control Unit	Model	Program	Program
Vehicle	Model	Model	Model

Fig. 1. Model-based Development Process

2. Quality of ECU Common Development Process

2-1 Issues on design modification of test patterns

For software development with the former design, test patterns need to be modified in some cases even if the functions of the model have not been changed. This section shows an example of these cases.

As shown in **Fig. 2**, the functions of one ECU are allocated to multiple ECUs. In Structure **(a)**, one ECU controls the functions of receiving an input signal from "SW_X" and transmitting the output signal to load "RLY_Y." In Structure **(b)**, ECU-1 receives the input signal from "SW_X" and then the ECU-2 transmits the output signals to "RLY_Y."

By regulating the multiple ECUs via network communications, it is possible to reduce costs by shortening the entire length of vehicle wiring systems. If the structure of the ECU is changed, ECU test pattern in Structure 1 can be converted to the ECU-1 and ECU2 in Structure 2 for testing the same functions to achieve the effective test design.

As shown in **Figs. 2 (c), (d)**, many changes should be

made to the test pattern due to the variations of interfaces such as presence or absence of the signals and respective I/O signal types (a direct connection[*4] or network communications).

In **Figs. 2 (e), (f)**, we compared the switch timing, based on the test patterns between "ECU in Structure 1" and "ECU2 in Structure 2." This is an example of changing the input interfaces from a "direct connection" to "CAN[*5]."

The filter processing is performed using the software to retrieve information of the switch input from a "direct connection." The filter processing is to monitor the signal state at regular time intervals. If the condition remains constant beyond a predetermined time, then it is judged as input information inside of the software. This function is also incorporated into the vehicle-embedded software to eliminate chattering of the contact switch. For Structure 1, the switch input operation by a user is judged as the internal information of the software, after a certain period of time of filter processing (Ta). On the other hand, for the ECU2 pattern in Structure 2, filter processing may not be triggered to receive information of the switch input via network communications on the ECU-2 side, because filter processing is already complete on the ECU1 side. Therefore, the time from the signal input to output may differ between those test patterns. Then, it is required to adjust the time duration from the signal input to output, in addition to the changes of I/O interfaces, I/O signal types, and presence or absence of signals.

In order to respond to the variations of I/O interfaces and timings for ECUs, design of each test pattern should be modified manually. Although the commercially available auto-testing tool is used to enhance the efficiency of testing, it is required to change the test patterns. For ECU software used for the body control module[*6], which is our development target, the affected area tends to be extensive, due to the increased number of changes for I/O interfaces for multiple I/O signals involved in this software.

Because of the significant amount of workload, there is the possibility that errors may occur due to a human intervention in creating the test patterns, and we were concerned about the product quality.

2-2 I/O signal allocation and conversion tool

To resolve the problems specified in 2-1, we developed a tool called "signal allocation and conversion tool" to automatically convert the changes in I/O signal allocations, such as signal names, input types, and input timing, based on the original test pattern.

To be concrete, the "basic test pattern" is created in advance, according to the timing of switch input and the recognition of input information, as shown in **Fig. 3 (a)**. As for the input type of a "direct connection," time duration (Ta) from the timing of switch input to that of input confirmation is moved to an earlier point, by filter processing. So, the test pattern that changes at the earlier point can be created automatically, as shown in **Fig. 3 (b)**.

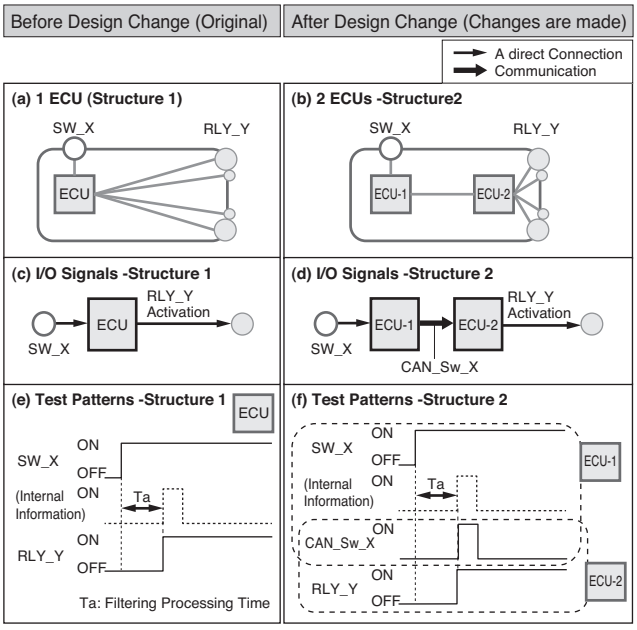


Fig. 2. ECU Structure and Test Patterns

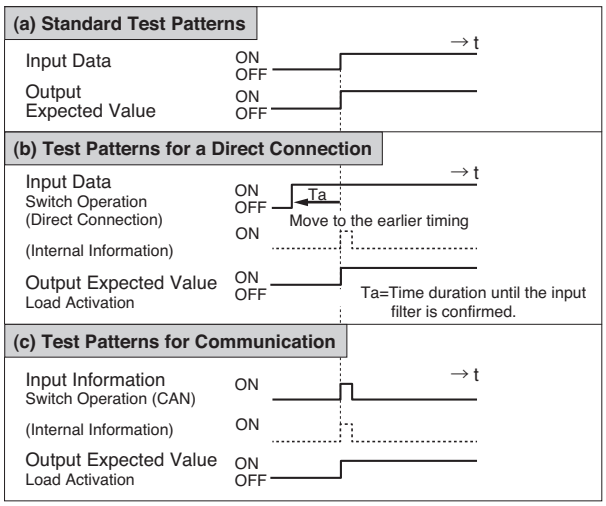


Fig. 3. Examples of Signal Input Timing

These allocation and conversion tools have several parameters that can be set separately for each signal included in the test patterns (**Table 1**). Necessary items for various test patterns related to signal names, input types, input timing, and deleting the signals, can be set to the parameter, although the functions are the same. Data can be consolidated using one setting file.

Time base, which is used to adjust timings, can be set in a manner relative to the original test pattern. If the test pattern exists for a specific car model, it enables us to convert test patterns directly to the other input types without creating the "basic test pattern" again.

Table 1. Settings of Signal Allocation Parameters

(a) Descriptions of Signal Allocation Setting Parameter

No	Items	Descriptions
(1)	I/O types	I: Input to ECU /O: Output from ECU
(2)	Packaging Types	O: Packaging /N: No Packaging
(3)	Basic Signals	Signal name defined by the standard test patterns
(4)	Intended Unit Signals	Destination signal name
(5)	Intended Unit Frame	Destination frame name (as for CAN signals)
(6)	Signal Types	P: Port Signal /C: CAN Signal
(7)	Adjusting Time for "ON"(ms)	Adjusting time when switching OFF to ON
(8)	Adjusting Time for "OFF"(ms)	Adjusting time when switching ON to OFF

(b) Settings of Signal Allocation Parameters

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
I	O	SW_00	SW_X		P	30	30
I	O	CAN_Sw_00	CAN_Sw_X	F_CAN_Z	C	0	0

2-3 Efficiency of signal allocation conversion tools

In this experimental study, we assumed that the type of an input signal was changed and then test patterns were created for the specific car model. We experimentally compared the man-hours required for the development work in the following two conditions; (1) test patterns are manually changed and (2) the allocation-conversion tool is used.

In this experiment, nine types of input signals are converted from those of the original car model. We assumed that 20% of the functions could be affected in the later model after changes are made. The result is that when the tool specified in (2) is used, actual man-hours required for the test pattern changes are reduced by approximately 20% compare to manual work specified in (1).

The automation of test pattern changes prevents errors in converting signals due to oversight. This tool makes it possible to improve the design quality at an earlier phase of testing.

3. High-Efficiency of Model-based Development

3-1 Problems on software testing for model-based development

For the model-based development, the accuracy of the control function is verified by simulation at the design phase. Simulation tests are conducted for three targets: the designed model (MILS), the source code (program) retrieved from the model (SILS[*7]), and the ECU embedded with the program (HILS) (**Fig. 1**).

It is necessary to verify that the model, program, and ECU respond to the input signals in the same way. Therefore, common test patterns should be used for MILS, SILS, and HILS to enhance the work efficiency in the verification process and secure product quality. However, the test scopes differ between the model and ECUs, posing a problem in the use of common test patterns. The following describes the reasons why the scopes differ between ECU testing and design simulation.

In general, vehicle-embedded software is composed of three types of modules for input processing, applications, and output processing (**Fig. 4 (a)**). In this structure, when the I/O interfaces are changed, only the relevant modules need to be replaced. As the entire configuration of the application module is not affected, it can be effectively reused for new software.

In this case, the design simulation is performed by each function module of MILS application as shown in **Fig. 4 (b)**. Hence, internal signals of the software are used as input signals. For HILS, external signals of the hardware for ECUs are used as input signals. As a result, signal confirmation timing may be different between interfaces and the external part.

The following is an example of the input signals that are generated at fixed intervals by the communication interfaces.

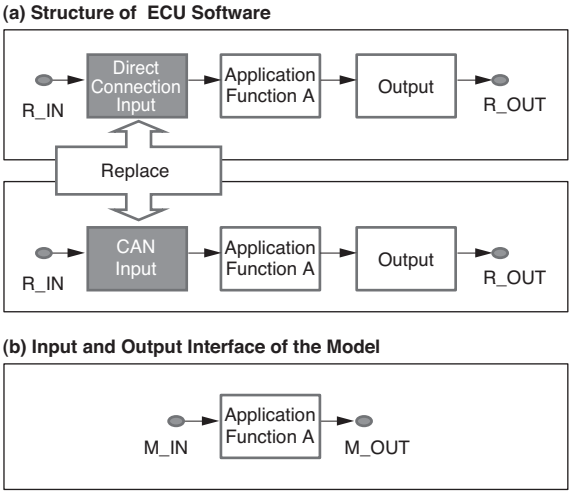


Fig. 4. Software Structure and Model Interfaces

To check for ECU communication errors, we interrupt the signals from the communication line deliberately (shift from **(a)** to **(b)** in **Fig. 5**). If an input signal is not retrieved from the communication line after a specified time (**Fig. 5 (c)**), the function is provided to set the value for communication errors.

In test scenarios of design simulation, the model intended for error testing is regarded as a function unit that is independent from the communication interfaces. Therefore, the specified value should be set to generate a signal only for communication errors (**Fig. 5 (d)**).

To generate the output signals at the same timing between MILS and HILS, the test pattern must have the earlier input timing, in order to interrupt the flow of signals from the communication line before confirmation of communication errors inside the software, for test scenarios of HILS.

From the reasons above, test patterns for the models

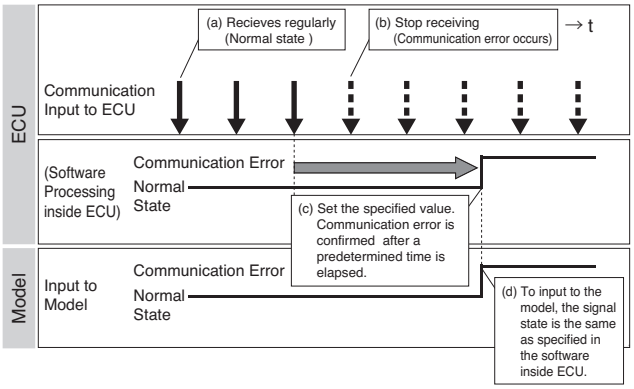


Fig. 5. Difference in Signals between Models and ECUs

cannot be used alternately for ECU testing. As we discussed in the previous chapter, the problem is the variation of input signal types in test patterns. But the main issue here is the variations of signal input timings. We concluded that there is the method for absorbing the variations in each test pattern according to input types used for the development of the software to be reused.

3-2 Creating input signal conversion programs for HILS

We created an input signal converting program for HILS, in order to absorb the variations of timing. This is an example of the signal converting process specified in **Fig. 6**.

In order to generate the output signals at the same timing between MILS and HILS, communication errors need to be generated outside the ECU before reaching the point where the communication error state is detected inside the ECU.

Figure 6 shows that a waveform of the signal is generated to trigger interrupted periodic transmission of a communication signal C, at the point where the input timing is made earlier based on the reference point t1, where the

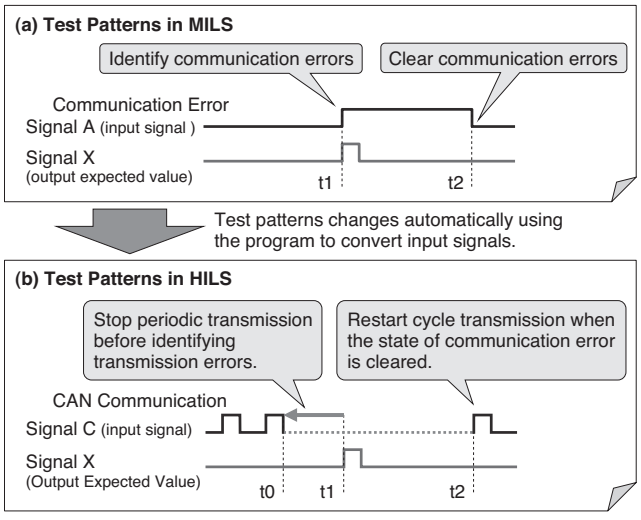


Fig. 6. Changing Test Patterns for Communication Errors

communication error state is confirmed, for test patterns in MILS.

To clear the communication error state for test patterns in MILS, a waveform of the signal is generated to restart transmitting the periodic signal at the same timing as t2, as the test pattern in HILS.

For this signal conversion program, the automatic conversion function is incorporated to convert variations of signal input timings between "network communications" and a "direct connection," as mentioned in 2-2. It enables us to convert the signal variations within test patterns for MILS to the signal input timing of the ECU in HILS.

The configuration of the newly developed test supporting tool is shown in **Fig. 7**. This tool is composed of the input signal converter program and the evaluation program, which absorbs the variations of test patterns automatically for verification in MILS and HILS. The former is to convert the variations of signal input timing automatically, and the later is to absorb the output variations caused by the hardware included in the ECU for HILS.

By providing these programs into the automatic testing tool, this system is configured to covert the signal input timing in real time during the HILS verification without creating the files for the test patterns changed from MILS to HILS.

As a result, only the basic test pattern files need to be stored, and multiple test pattern files, including those for various signal input timings, are not required any more. This improvement enables us to reduce the time for file management and enhance work efficiency.

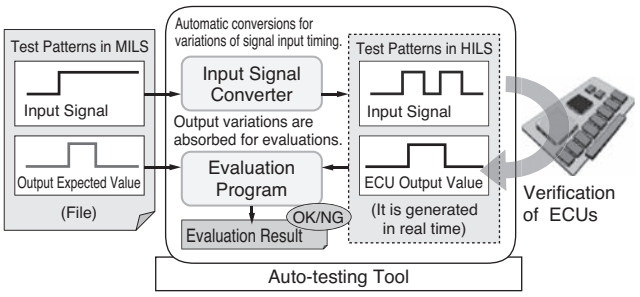


Fig. 7. Structure of Test-Supporting Tools

3-3 Application to model development and effects

By applying this tool to the procedure of model-based development, we evaluated the applicability of the tool for the actual development project. The evaluation was performed for a function with more than 20 types of the input signals and 8 types of output signals.

By assuming that the auto-testing tool was used, we estimated that man-hours required for testing were reduced by approximately 50% compare to those with the conventional tool.

By completing the design simulations in upstream process and applying the test patterns to the product testing, we verified that the product quality was improved in a

short time and the rework rate at the product testing phase was minimized.

4. Conclusion

In this paper, we described development of a test-supporting tool for more efficient testing and its effectiveness in dealing with sophisticated software.

First of all, we established a technology that uses common test patterns for simulation before and after ECU design change to secure product quality at the common software development. As a result, we achieved 20% reduction of man-hours required for changing the test patterns.

Secondly, as part of our efforts towards the model-based development, we established a technology that uses the common test patterns at a phase of design simulation and ECU testing. Using the auto-testing tools, we achieved approximately 50% reduction of man-hours compared to previous conditions. We believe that it is possible to secure product quality in a short time and minimize the rework rate at the product testing phase.

We continue to make further efforts for improvement of the evaluation process for the model-based development, in order to achieve high efficiency and quality at the development of vehicle-embedded software.

Technical Terms

∗1 ECU (Electronic Control Unit): An electrical control unit mounted on a vehicle.
∗2 MILS (Model in the Loop Simulation): Software verification process intended for a designed model.
∗3 HILS (Hardware in the Loop Simulation): ECU verification process in a simulated operating environment.
∗4 Direct connection: Cable connection to transmit a single signal.
∗5 CAN (Controller Area Network): A network standard for connecting electrical circuits or devices.
∗6 Body control: Electronic control of car interior equipment such as lights or door lock devices.
∗7 SILS (Software in the Loop Simulation): Software verification process intended for the source code which is automatically generated from a sample model.

References

(1) Akira KANAZAWA, "Extensive Verification of Activation Timing of In-vehicle Software," The 172nd Issue of SEI Technical Review, pp106-111 (2008)
(2) Information-technology Promotion Agency (IPA), Japan, "Investigation Report on 'Leading Model Based Development for Embedded Systems'" (2012) (Japanese only)
(3) Kenichi HORIKAWA, "Evaluation of Automotive Software Standard 'AUTOSAR'," The 175th Issue of SEI Technical Review, pp92-97 (2009)

Contributors (The lead author is indicated by an asterisk (∗).)

T. KATAOKA*
• Assistant Manager, Power Device R&D Department, Auto Networks Technologies, Ltd.

I. SAKA
• Western Customers ECU Department, Electronics Division, Sumitomo Wiring Systems, Ltd.

K. FURUTO
• Manager, Power Device R&D Department, Auto Networks Technologies, Ltd.

T. MATSUMOTO
• General Manager, Power Device R&D Department, Auto Networks Technologies, Ltd.