This paper is part of the Proceedings of the 15th International Conference on Railway Engineering Design and Operation (CR 2016)
www.witconferences.com

Validation tool of functional safety for train control secure software

J.-G. Hwang & H.-J. Jo
Korea Railroad Research Institute (KRRI), Korea

Abstract

A train control system is a vital piece of control equipment which is responsible for the operation of trains, and its functional safety must be validated before real field application. Most of the existing studies on functional safety testing for the train control system secure software have focused on verifying the functional safety through the monitoring of the internal memory embedded system. However, the train control system is one of the typical embedded control systems in the railway sector, and the embedded secure software has a characteristic of generating appropriate outputs through the combination of internal processing in consideration of the current internal status and external input. Therefore, the test approach of using the interface communication channel can be an effective way for the functional testing for railway signalling system software in consideration of these characteristics. Since a communication interface specification of the train control system has the properties of the sequence input and output signals, a test-case for software testing is the most effective methodology by MSC language, one of the graphic languages. The MSC-based testing tool for functional safety of train control system secure software was developed and its applicability to the prototype of a train control system under development was confirmed.
Keywords: secure software, train control system, software testing.

1 Introduction

A railway signalling system is one of the representative embedded systems in the railway area and it enables mutual exchange of input/output information with other control systems through interfaces. In addition, with the active employment

of communication technologies in railway signalling systems in recent years, the railway signalling system has evolved into a system which performs railway signalling functions through interfaces with more control devices. In other words, each railway signalling device has had unique functions in the past, but a railway signalling system able to perform more various and complex functions through interfaces with other systems is being developed and employed [1].

The railway signalling system's embedded software receives external input signals, processes them in combination with internal status information at that time and produces more than one appropriate output. Therefore, the most effective method for validation of functional safety may be to perform a functional safety test through external interface communication channels actually used by the railway signalling system under consideration of such embedded software's operational characteristics. Various test tools are available for the validation of embedded system software, such as a railway signalling system, but they do not sufficiently consider the embedded software's characteristics described above. Accordingly, taking into account such characteristics, research has been conducted to utilize interface communication channels actually used by an embedded system to input test data and provide feedback information [2, 3].

The most effective method for validation of functional safety may be to perform a functional safety test through external interface communication channels actually used by the railway signalling system under consideration of such embedded software's operational characteristics. However, the testing methods based on interfaces actually used are considered appropriate because of consideration of the embedded software's characteristics, but they have some limitations in that interfaces actually have communication characteristics. In general, interfaces have the input/output signal sequence characteristic, but testing tools under development utilize input/output interface signals in testing under insufficient consideration of such a sequence characteristic. In other words, test cases are created using TTCN-3 (Testing and Test Control Notation Ver. 3) script languages developed for suitability testing of communication protocols. Since TTCN-3 standard is used in generating test cases, the sequence characteristic appears to be considered, but end users (testers) may think that this characteristic is not fully taken into account [4, 5]. TTCN-3 is an international standard testing script language standardized by ISO and ITU for suitability testing of mobile communication protocols and Internet protocols. It may be suitable for the testing of embedded software based on actual interfaces previously researched, but end users who do not have sufficient knowledge in TTCN-3 script language itself may have difficulties.

Moreover, since the railway signalling system's external interfaces have the input/output signal sequence characteristic, using MSC (Message Sequence Chart), one of graphic languages to express sequences by scenarios, for test cases to test embedded software through input/output signals is a method that can be easily understood by users and that can effectively and accurately express communication protocols [6–8].

For all MSC-based supporting tools, the input of test cases is based on MSC, but test results are in script format, making it difficult for testers to use them. The

test case generation and testing method through generation of MSC-based test cases and automatic conversion into TTCN-3 script language has many advantages, such as easy use and generation of test cases with higher accuracy for validation of railway signalling system's embedded software characterized by an output based on external input data and internal conditions.

2 Test cases using MSC

In general, one process in embedded software is based on values obtained from prior operations. For example, in order to calculate and output the speed limit for the current train, data on locations of preceding trains is needed. Therefore, such data on locations of preceding trains corresponds to results from prior operations needed for calculation of speed limit for the current train. In other words, in order to perform a process, results from prior processes have to be sequentially utilized. Results from a process are used for another process.

MSC is a Z.120-standardized, graphic-based specification language, which is widely used in specification of communication protocol. MSC is a graphic method to easily express communication participants, time (the axis of ordinates), data flow between communication participants at each time point and data. It can be easily used for the effective expression of communication protocol.

Communication protocol specification is for sequential data exchange between two objects and MSC enables description of data to be transmitted. As described above, test cases for validation of embedded software based on interfaces actually used in a railway signalling system can be effectively created in MSC. A method for automatic conversion of such MSC-based test cases into TTCN-3 (international standard testing language) script language for the purpose of testing embedded software was proposed and the tool for this testing method was developed.

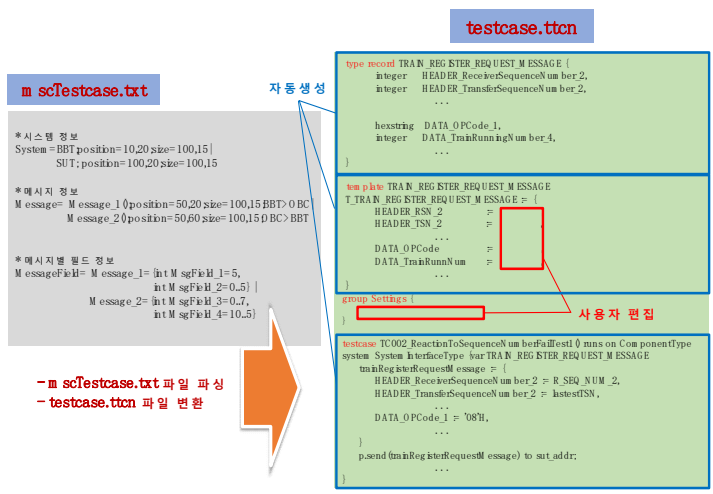

Figure 1: Conversion of TTCN-3 script file.

Test cases for previously developed testing tools are based on a TTCN-3 script. As shown in the figure, a test case consists of three groups; message, settings and test case. The message group defines the message structure and message input value's ranges. The settings group defines communication information on the interfaced system (IP and port number). The test case group includes transmission field's values for actual testing. For test cases in TTCN-3 script, it may be difficult for a tester to correct or edit them as necessary. In addition, it is difficult to intuitively understand a test case. Therefore, much effort and time are needed to create test cases.

In this study, in order to improve the user's convenience and assure an intuitive understanding of test cases, a module was developed to input test cases based on MSC easily understood by users and optimized for communication protocol specification and to automatically convert them into TTCN-3 scripts. For MSC test cases, automatic matching is employed to individually recognize and save message names, system names and message parameters. As shown in the figures, coordinates where system names and message names are located are recognized, messages are sequenced on the basis of such coordinates' information and then, they are converted into text files. Fig. 1 shows the process of converting such MSC-based test cases into text files.

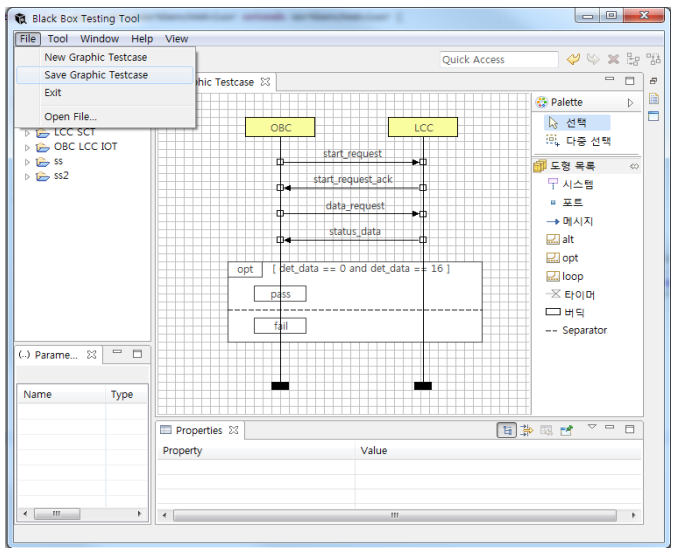

Figure 2: Windows for MSC-based test-case editing.

The process of inputting MSC-edited test cases into testing tools was based on a prior study [9]. MSC consists of events with a defined sequence between sending and receiving objects and such events correspond to message information. The sequence of such events is controlled and saved as a separate parameter and each sending message is referenced by this parameter. More information is described in the reference [9].

When an MSC file is saved as a text file, the developed automatic conversion module performs parsing of such text file to automatically convert it into the relevant part in TTCN-3 script. As shown in this figure, when values are entered into individual fields at MSC, they are automatically converted into script, which can be edited in the resultant TTCN-3 script.

3 Development and application of tool

An automatic test case creation module based on MSC input of a testing tool for black box testing of railway signalling system software described above was developed and it was applied to the previously researched tool. Fig. 2 shows procedures for the input of an MSC-based test case, automatic conversion into and editing of script language, performance of testing using such test case and display of test results like Fig. 3. As shown in this figure, a test case is inputted in MSC format and output is also made in the same MSC format. In other words, the outcome developed in this study is the MSC-based test case input and output module, allowing testers to easily and intuitively understand input and output.

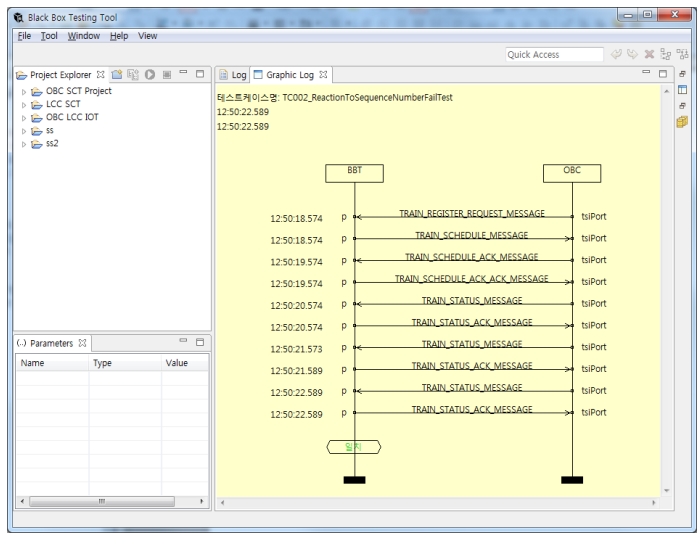

Figure 3: Windows for MSC-based test results output.

In conclusion, when compared to the conventional method for creating a test case in script format, users are able to easily create test cases, leading to the decrease in errors in the course of the creation process. In Fig. 3, the right part shows various objects in MSC. Fig. 4 shows the output of test results in MSC format, including the test case name, testing start time and end time. The figure's left part shows the message sending/receiving time during testing. "Consistency" at the bottom means the consistency with the test case, indicating that, when

testing was conducted according to the relevant test case, the target system was found to be in normal operation.

The black box testing tool developed in this study was applied to an actual train control system to confirm its applicability. Applicable train control systems include radio communication-based train control system's onboard system and ground control and monitoring system and their prototypes are being developed in Korea. Functional safety of software for such a system under development was validated using the testing tool developed in this study in order to confirm its applicability. Results illustrated in this paper are derived from such application.

```
control : {13:34:01.848} : // Time: 13:34:01.848. Date: 27/Oct/2015. MOT version: TC: 4.2.5.

********************************************************************************
*** RUNNING TEST CASE LCC_SCT_02

mtc : {13:34:01.927} : // CASE LCC_SCT_02 STARTED
mtc : {13:34:01.927} : log("This executable code has been generated by evaluation copy of OpenTTCN compiler. Can be
used for non-commercial evaluation purposes only");
mtc : {13:34:01.927} : map(mtc:p1, system:tsiPort1);
mtc : {13:34:01.958} : p1.send(TRAIN_REGISTER_REQUEST_MESSAGE : { HEADER_ReceiverClass_1 := 2,
HEADER_ReceiverId_1 := 1, HEADER_SenderClass_1 := 1, HEADER_SenderId_1 := 1,
HEADER_ReceiverSequenceNumber_2 := 0, HEADER_TransferSequenceNumber_2 := 0, HEADER_DataLength_2 := 14,
DATA_OPCode_1 := '08'H, DATA_TrainRunningNumber_4 := 825241649, DATA_ICT_ID_3 := 3223600,
DATA_DriverID_4 := 825241648, DATA_TrainCategory_1 := 2, DATA_TrainLength_1 := 100 }) to { host := "127.0.0.1",
portField := 7002 };
mtc : {13:34:01.958} : timer_1.start(30.0); // Timer is started: duration 30 s.
mtc : {13:34:02.020} : p1.receive(TRAIN_SCHEDULE_MESSAGE LCC_SCT_02.T_TRAIN_SCHEDULE_MESSAGE :=
{ HEADER_ReceiverClass_1 := 1, HEADER_ReceiverId_1 := 1, HEADER_SenderClass_1 := 2, HEADER_SenderId_1 :=
1, HEADER_ReceiverSequenceNumber_2 := 0, HEADER_TransferSequenceNumber_2 := 1, HEADER_DataLength_2 :=
11, DATA_OPCode_1 := '0F'H, DATA_TotalPaths_1 := 9, DATA_PathIDn_I_1 := 10, DATA_PathIDn_II_1 := 20 }) from
{ host := "127.0.0.1", portField := 7002 };
mtc : {13:34:02.020} : setverdict(pass);
mtc : {13:34:02.020} : log("TRAIN_SCHEDULE_MESSAGE received by vaild ReceiverSequenceNumber of
TRAIN_REGISTER_REQUEST_MESSAGE.");
mtc : {13:34:02.020} : unmap(mtc:p1, system:tsiPort1);
control : {13:34:02.036} : // CASE LCC_SCT_02 FINISHED WITH PASS

////////////////////////////////////////////////////////////////////////////////
/// TEST CASE LCC_SCT_02 COMPLETE VERDICT PASS


********************************************************************************
*** TEST EXECUTION SUMMARY

Pass Fail Inconc None Error Total Duration
1 0 0 0 0 1 00:00:01
```

Figure 4: Output of test results in text file (example).

Fig. 5 shows the structure of the target system and interface for black-box testing. As shown in this figure, the new black-box testing tool inputs test case data and receives feedback data through the interface channel and snooping device. The interface channel was developed to allow expansion of connectivity with other communication protocols. The interface channel is a device for data communication between the testing tool and target system. It sends data from the testing tool to the target system's interface and it receives data from the target system to the testing tool's interface.

4 Conclusion

The conventional railway signalling system's black box testing tool creates and edits test cases based on a TTCN-3 script. However, if test cases are created in script language, errors may be introduced. In addition, users may have difficulties in creating and editing test cases in script language. A test case for communication channel-based testing, such as a railway signalling system is generally based on specification in MSC format. Therefore, when compared to the creation and edition of test cases in script language, the creation and edition of a test case in MSC format may have various advantages, such as a decrease in errors, because testers are able to intuitively understand them. Accordingly, a module was developed in this study to allow editing and creation of test cases in MSC, a graphic language familiar to users, and to automatically convert them into script language for testing. In addition, it was designed to output test results (test log) in both MSC and script. Editing of test cases through a graphic editor enables the re-use of test cases and the text-based log is organized to assure user's easy analysis. The module's applicability was tested. In this test, conversion of an MSC-based test case into script language and performance of testing based on such automatically converted test case were confirmed. Test results were also expressed in two formats (MSC and script) and normal output was confirmed through comparison of two test logs.

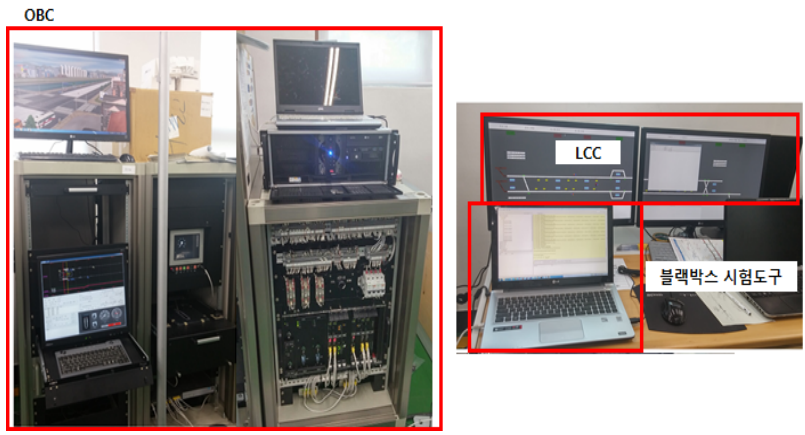

Figure 5: Testing for applicability of tool to train control system.

References

[1] Korea Railroad Research Institute, "Development of technology for safety and efficiency improvement of train operation", KRRI research report, Dec. 2014.

[2] J. G. Hwang, H. J. Jeong and B. H. Kim, “Results of Coding Rules Testing of Train Control System Software”, International Journal of Software Engineering and Its Applications, Vol. 7, No. 3, pp. 249-257, 2013.
[3] J. G. Hwang, J. H. Baek, and H. J. Jo, “Interoperability test methodology for train control system using interface channels”, CMEM 2015 International Conference Proceeding, May 1015.
[4] C. Willcock, T. Deiss, S. Tobies, S. Keil, F. Engler, S. Schulz and A. Wiles, An Introduction to TTCN-3, 2nd Edition, John Wiley & Sons Ltd., 2011.
[5] http://www.ttcn-3.org/index.php/downloads/standards
[6] E. Rudolph, J. Grabowski, and P. Graubmann, “Tutorial on Message Sequence Charts (MSC’96)”, Tutorial of FORTE/PSTV’96 conference proceedings, Oct. 1996.
[7] ITU-T, “Recommendation X.120: Message Sequence Chart (MSC)”, Setp. 1994.
[8] J. Graowski, D. Hogrefe, I. Nussbaumer, and A. Spichiger, “Test case specifications based on MSCs and ASN.a,” Proc. of the Seventh SDL Forum 1995, pp. 307-322, 1995.
[9] H. S Bae and Y. R. Kwon, “Validation of timing and communication constraints in real-time parallel programs”, Phd. Dissertation in Computer Science, KAIST, 1999.