Design of automatic testing tool for railway signalling systems software safety assessment

J.-G. Hwang[1], H.-J. Jo[1] & H.-S. Kim[2]
[1]*Train Control Research Team,*
Korea Railroad Research Institute (KRRI), Korea
[2]*Electrical and Computer Engineering, Chungnam University, Korea*

Abstract

Recent advances in embedded system technology have brought more dependence on automating train control. While great efforts have been reported to improve electronic hardware safety, there have been fewer systematic approaches to evaluate software safety, especially for the vital software running on board signalling systems. In this paper, we propose a new software tool to evaluate train control system software safety. We have reviewed requirements in the international standards and surveyed available tools in the market. From that, we identified necessary tests to meet the standards and proposed a tool that can be used during the whole software life cycle. We show the functional architecture and internal components of the tool. This tool is unique in that it is a comprehensive tool evaluating reliability and safety together.
Keywords: railway signalling systems, S/W testing tool, safety evaluation.

1 Introduction

The train control system has recently been converted from the existing mechanical device to a computer system, and dependence on the software has been rapidly increased. As for the representative system, the Japanese EJTC [2] ATC (Automatic Train Control) system can be cited. The ATC of EJTC is composed of anything from the vehicle control through a wayside signal exchange at Level 0 to the unattended fully automated vehicle control system at Level 3. In this way, with the transition from the mechanical, manual vehicle signalling system at the earlier stage to the recent unattended fully automated train control system, multiple computers began to be used as on-board

WIT Transactions on Information and Communication, Vol 39, © 2008 WIT Press
www.witpress.com, ISSN 1743-3517 (on-line)
doi:10.2495/RISK080501

equipment, and the validation on the reliability and safety of the software to be mounted on these computers began to gain force as an important issue. The safety of software is being accomplished primarily by carrying out safety activities at the software design stage, which is the earlier stage of development for software. As for the representative safety activities, HAZOP, FTA and FMECA [4,5], etc. can be cited. Though techniques like these are being utilized at the earlier stage of development for software, no automated or formalized method has been applied for additional validation on safety after completion of development. In case of authenticating the safety of software, when evaluating the safety activities of an authenticated institution, it is difficult to grasp the faithfulness of software safety activities if they are evaluated by relying on the documents provided by the authenticated institution, and it may be the case that it is necessary to conduct additional validation. In this case, it will be very helpful if there is a tool to evaluate the safety of software automatically because it can enhance the reliability of the evaluation. The role of software for the on-board mounted computer is becoming more important in accordance with the trend toward automation and self-regulation of train operation, and therefore, the effect of software on the whole train control system is being increased. The on-board mounted computer has been becoming gradually high-powered in accordance with the rapid development of the microprocessor market, and in the case of its programming languages being used, the Ada, which is the superior level language has also been used recently in addition to the simple assembly language [2].

This study suggests the software capable of evaluating the safety of software for train control systems automatically. For this purpose, we analyzed relative international standards and investigated existing testing tools for the software used. From this, we suggested the tool capable of testing the main requirements to meet the international standards, which can be used throughout the whole development cycle for software. Unlike other existing tools, this tool is very significant in the respect that it can verify the safety and reliability of software simultaneously. This paper describes the results of the design stage of the tool, and its composition is as follows.

In section 2, the evaluation method on the safety of the train control software suggested in this paper is explained, and the requirements of the testing tool once automated are described in section 3. In section 4, the main testing items that must be implemented for the suggested tool will be explained, and in section 5, the architecture and main functions of the testing tool will be suggested, and the conclusion will be given in section 6.

2 Evaluation of the safety of train control software

In this study, we suggested the safety test method of software by analyzing IEC61508 [3], which is the international standard for the train control field. In IEC61508, the Software Safety Integrity Level (SWSIL) is defined, and the formalized development process is suggested, and the validation techniques are presented by each process in accordance with the level of SSIL. The evaluation

of software safety suggested in this study extracted evaluation items to be performed at the implementation, verification and testing, hardware combination test, validation and assessment stages which are those following the software design stage. In this section, the software safety requirements to meet IEC61508 are described.

2.1 Software safety integrity level

The evaluation of software safety is accomplished by validating whether the software developed can satisfy the Software Safety Integrity Level (SWSIL) given at the time of designing the software. SWSIL is not defined by software autonomously, but determined to be identical to the Safety Integrity Level (SIL) of the system applying the software. However, if it is possible to prohibit an error in the software from propagating to the system, it was made to be determined at a lower level than this. SWSIL is classified into five grades as follows in accordance with the risk of system.

2.2 Method of safety validation in the development stage

The software development process presented by IEC62279 is composed of the development process and validation process as shown in Fig. 1. The standards present requirements to be satisfied at every stage, and as for the main requirements, they present validation methods for requirement by dividing into M (Mandatory), HR (Highly recommend), Recommend and Not Recommend, etc. in accordance with SWSIL. In applying the techniques at testing, it is recommended to use automated testing tools. The following are the arrangements centred on the validation techniques required as M and HR in the case of SWSIL 4 grade, which is the highest safety grade among validation techniques presented by standards.

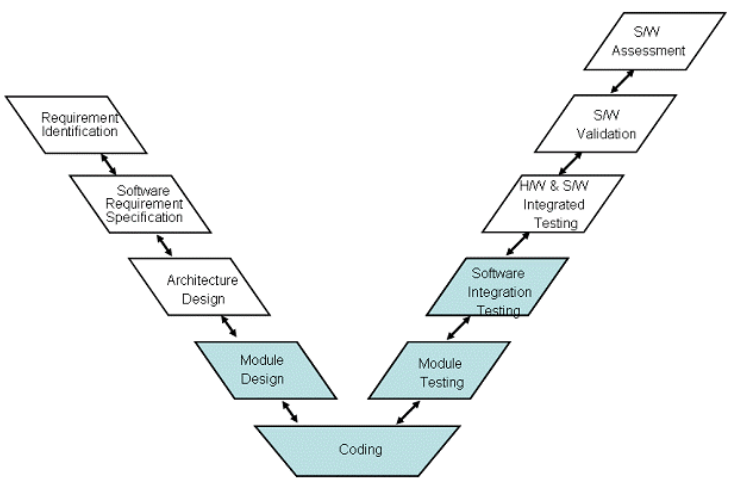

Figure 1: Software development life cycle in IEC61508.

○Design and implementation of the software: In the process for design and implementation of the software, the compliance with the coding rule and black-box testing is required as M validation technique. As for the coding rule required, the prohibition from using the dynamic objects and variables, restriction to the use of the pointer and recursive function, and the prohibition from using the unconditional jump were suggested.

○Software verification and testing: As for the verification and testing techniques, the Formal Proof, probability test, static analysis, dynamic analysis and Software Error Effect Analysis, etc. were suggested as HR items.

○Software/hardware integration test: In the software and hardware integration test stage, the function test, black-box test and the performance test are required as HR level.

○Software Validation: In the validation stage, the performance test, function test and black-box test are presented as M items, and the probability test is presented as HR item.

○Software Assessment: In this stage, the purpose is to finally validate whether development processors and developed software are satisfying the SWSIL defined at the earlier stage of design. As for the validation techniques to be used, the checklist, static analysis, dynamic, Fault Tree Analysis, Software Error Effect Analysis and the Common Cause Failure Analysis, etc. are suggested as HR items.

3 Requirements for the software safety evaluation tool

IEC62279 recommends using automated testing tools to validate software safety for the train control system. The following requirements were drawn upon to design automated tools evaluating the safety of the software.

①It must be able to validate the safety requirements of the international standards in relation to the train control software.

This study drew the software safety validation requirements for the international standards by analyzing IEC61508 and IEC62279. The techniques that could be automated were drawn by analyzing the testing and validation techniques that are required from the development stage of the software lifecycle. It should be possible to use the test tool at every development stage for the software, and it must be able to apply validation items required by standards at each stage. Thus, if we use the tool to be developed, the reliability of the development process can be enhanced since we may validate whether it complies with IEC61508 or not objectively. The drawn evaluation items are described in section 4.

②Various forms of test combinations must be performed in accordance with the choice of valuator.

The tools to be developed are supposed to be used by software developers, valuators and also certification institutions, and they must not only perform the formalized test, but be able to compose tests in accordance with the

choice of users, so that certification institutions can apply the test techniques used at the development process for the purpose of verification. It must be able to combine necessary tests in accordance with the safety level being tested, and in accordance with the development stage.

③The static analysis and dynamic analysis on the software must be available.

Most of the existing automatic testing tools for software are those for static analysis based on the source code analysis. However, the importance of dynamic analysis has been brought to the fore recently for the validation on runtime characteristics that might not be verified with static analysis. For the dynamic analysis, it is necessary to develop agent programs that are viable at the embedded target board with high universality.

④The results of the main safety activities at the design stage must be utilized by the test tool.

The existing automatic software testing tools have been used for the purpose of enhancing reliability, and though they can be used for safety validation, they do not provide the additional function of safety validation. Since testing tools were designed to input test cases directly by users or create them randomly, it is difficult to ensure the linkage between the results of safety activities to be performed at the design stage and developed products. To solve this problem, it is essential to have the function enabling to validate the results of the Fault Tree Analysis, or Hazard and Operability (HAZOP) analysis being used at the design stage of the development stage. Therefore, in the newly developed safety evaluation tool, we enabled the function to create and validate test cases by receiving the results of safety activities at the design stage as inputs automatically through differentiating them from existing software reliability testing tools.

⑤Testing tools must correspond with the international safety standards.

The safety test tool to be developed in this study can be used for the certification of international standards, and for this purpose, the tool itself must be certified also. Thus, the tool to be developed must also obtain the certification of IEC61508 or DO-178B, etc.

4 Evaluation items for the software safety evaluation tool

In this section, the evaluation items selected to implement those among the safety validation requirements defined in IEC 61508 and IEC62279 of the safety evaluation tool are introduced. We selected those possible to implement as automation tools among the validation methods required by standards preferentially, and extracted those among the various techniques centered on the validation methods prescribed as M and HR. To draw evaluation items, we divided the techniques largely into a total of six steps from ST1 to ST6 by focusing on the stages in relation to behaviours in which the implementations of software are to be prepared or validated after being prepared, which are listed as follows:

ST1: Software module testing stage
ST2: Software integration testing stage

ST3: Integration stage between hardware and software
ST4: Software validation stage
ST5: Software change validation stage
ST6: Software evaluation stage

Each stage is the extracts of stages containing items to evaluate software from software development stages of IEC61508. This standard, associated directly with software for the train control system, requires automated tests at the main software development stage, and the main development techniques and testing techniques to be applied are defined in accordance with the safety integrity level of the software. In the safety evaluation tool for software, this can be applied to the software safety integrity level 4 for the highest safety among the integrity levels, and we drew it as a requirement to be implemented with the tool for the corresponding technologies whose usage was classified as M or HR only.

Table 1: Selected testing items.

Test techniques	Performance stage						Classification of techniques
	ST1	ST2	ST3	ST4	ST5	ST6	
Performance testing	x	x	x		x		Non-Functional
Boundary value analysis	x	x	x	x	x	x	Black-box
Equivalent classes	x	x	x	x	x	x	Black-box
Design & coding standard	x	x					Maintenance
Control flow testing	x	x	x	x	x	x	White-box
Data flow testing	x	x	x	x	x	x	White-box
Fagan inspection					x	x	Analysis
Symbolic execution					x	x	Analysis
Checklist						x	Analysis
Metrics	x	x				x	Analysis
Decision table						x	Analysis
FTA						x	Analysis

Table 1 shows 12 key testing items drawn in this manner. Table 1 prescribes test items to be applied at each development stage of the software. For example, the performance testing means that it can be used for the software module tests, integration testing, hardware integration testing, and change validation stages. The evaluation items can be classified into three techniques such as white-box testing, black-box testing and source code analysis in accordance with the evaluation technique. The white-box testing and black-box testing are dynamic testing methods executing and analyzing software at the target, and the source code analysis technique is the static analysis method at the state of not executed software. The white-box technique corresponds to the case where it is possible to use the information on the internal architecture of the software that is the object of the test at the time of testing, and the black-box is the opposite case. The test

items to be implemented by white-box testing are the performance test, control flow and data flow test. The performance test is the one carrying out the hardware processing capability and resource state required at the time of implementing the software in the form of dynamic testing software. In the control and data flow testing, it tests whether any unused code or data area has occurred by tracking down the control flow and data flow of the software.

The black-box testing is composed of the boundary value analysis and equivalent class testing. The boundary value analysis testing is the test that has to be applied to all of the six software development stages defined in this section, and it inspects the software errors occurred at the limitation or boundary of the parameters. The equivalent class testing is the test detecting any error based on the input variable by using the minimum test data. To do this, it is important to regulate test data so that the whole range of input values can be included. As for the static analysis method, it includes general functions provided by existing analysis tools for software. In Fagan inspection, it is the automation of the inspection on general software performed by external specialists, and the Metric analysis refers to the measurement of unique characteristics of software such as reliability or complexity by analyzing the structural characteristics of software.

5 Architecture of the software safety evaluation tool

In section 4, we presented evaluation items for software safety that are required to be automated by analyzing relative international standards. In this section, we explain the architecture design of the safety evaluation tool for software.

The software safety evaluation tool for the train control system is composed of the automatic creation tool for the test case, the automatic test performing and monitoring tool, and the target testing agent. Since the train control system has the characteristics of an embedded control system, it is necessary to design the architecture for the software test tool that is being tested and monitored through the testing agent program of the actual target board where the applied software was ported. Therefore, the test tool is being inputted by converting the safety analysis data of the software targeted for evaluation by using the conversion module for source code and input data, and creating test data and scenarios by using the automatic test data creation module and automatic test scenario creation module on the basis of the input source code and safety analysis data.

The test results will be analyzed by performing the testing through the automatic test performance and monitoring module and target testing agent, and it has the architecture to store the results on screen and on file. Fig. 2 shows the process of using the safety testing tool being suggested. Fig. 3 shows the main function of the testing tool by stage, and Table 2 shows the function of the test tool and output.

The definitions for the function of the main stage of the testing tool are as follows:

- Program analysis: Creates function information, type information, control flow and call information among functions necessary for the test through program analysis

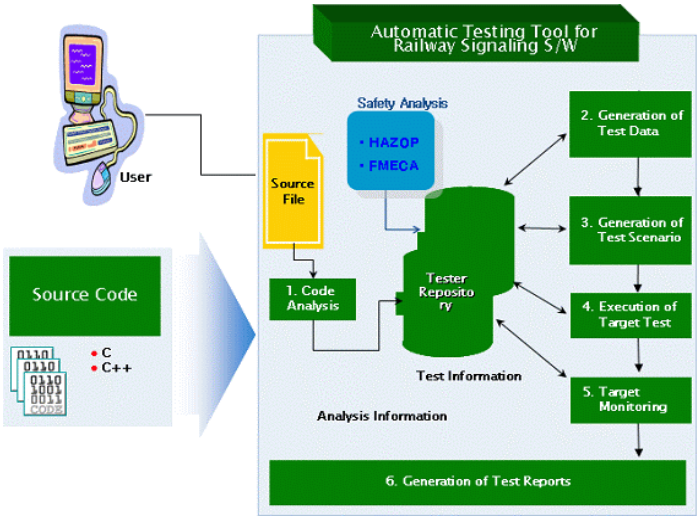

Figure 2: Software safety testing flow.

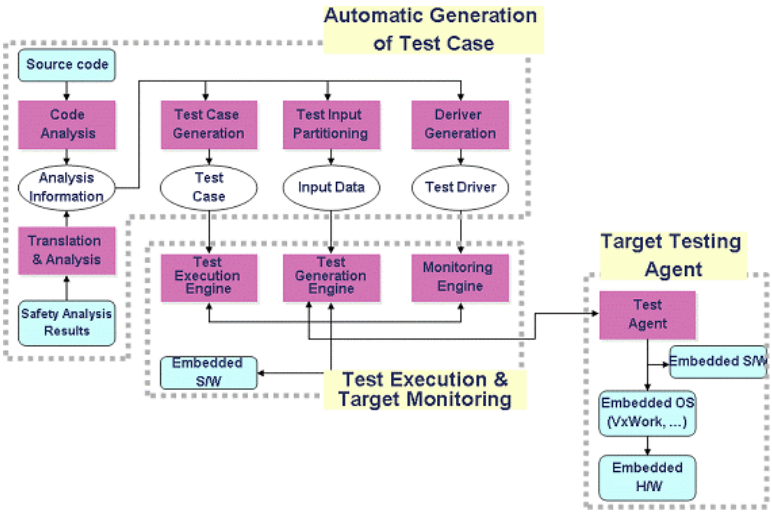

Figure 3: Software testing tool system design.

- Input division: Creates information on the input data division by data type on the basis of information obtained through program analysis
- Test scenario creation: Creates a test scenario automatically, and it also enables the user to create a test scenario (script)
- Driver creation: Creates drivers connecting the test target code with the test engine and the program where the test will be carried out

Table 2: Functions of software testing tool.

Classification	Detailed classification	Output
Program analysis	Analysis	Function and API list
		Control flow graph by function
		Call graph among functions
		Data architecture
Test case Creation	Type division	Division by data type
	Variable division	Division by variable
	Scenario	Test scenario
	Driver	Test driver
Compile & Build	Build	Test program
		Target image
Test performance	Execution	Result of performance by test case
		Test coverage
		Error diagnosis
Analysis on results	Report	Test report

- Execution: Performs the test and summarizes test coverage, details the test by each section and test results, and presents error locations, and gives the tool the function to report detailed results by test case to the user.
- Preparation for report: Creates the report on all the test information and results in accordance with the user's option

6 Conclusion

This paper suggested the safety evaluation software tool for a train control system. The suggested evaluation tool has the form of expanding the existing automated software test tool, and the evaluation items required by standards are performed in the form of a dynamic test using the results of safety activities derived from the software development cycle as the input. We made it include key evaluation items required by international standards, and used them during the development lifecycle of the software. We added the function to validate whether the safety is maintained or not continuously by using the results of safety activities performed at the software design stage as the input into the testing tools.

If the testing tool for embedded software having the suggested architecture is developed, it is anticipated that it will be very helpful for the evaluation of the software for train control systems.

References

[1] Matsumo, M., "The Revolution of Train Control System in Japan", Autonomous Decentralized Systems, ISADS Proceedings, 2005.

[2] Lawson H. W. et al, “Twenty Years of Safe Train Control in Sweden”, Engineering of Computer Based Systems, Proceedings. 8 Annual IEEE International Conference and Workshop on the, 2001.
[3] International Electrotechnical Commission (IEC), “61508 - Functional Safety of Electrical/Electronic/Programmable Electronic Safety-Related Systems”, 1999.
[4] Younju Oh et al., “Software Safety Analysis of Function Block Diagrams using Fault Trees”, Reliability Engineering & System Safety, 2005.
[5] Robyn R. Lutz and Robert M. Woodhouse, “Bi-directional Analysis for Certification of Safety-Critical Software”, proceedings of 1st International Software Assurance Certification Conference, Dulles, Virginia, February, 1999.