An Approach to Test Train Control System Software Safety

[1]Nirmal Kumar Gupta and [2]Mohammad Qasim Rafiq,
[1,2]Computer Science Department, Jaypee University, Anoopshahr, Bulandshahr, Uttar Pradesh, India

Abstract— With the recent development of embedded system technology, automation of train control system software is being promoted. Software for safety-critical systems like the train control system software has to deal with the hazards identified by safety analysis in order to make the system safe, risk-free and fail-safe. However, systematic efforts to verify the safety of software have been rarely performed. In this paper, we propose a framework that can automatically evaluate the safety of train control system software. To do this, we analyze the related international standards and investigate existing software testing techniques. From this, we have proposed a framework based on the McCall's software quality model. The proposed framework specifically identifies the criteria corresponding to software safety in train control system software to test the key requirements required by international standards.

Keywords— *Software safety, Software Quality, TCS (Train Control System), Software testing, Safety evaluation.*

I. INTRODUCTION

The recent developments in computing technology have created a way for the development of more complex control systems which have found their place in various technologies in every domain of life. Software systems which have the potential to cause accidents are termed as safety critical software systems [1]. In recent times the Train Control Systems (TCS) have also been shifted from existing mechanical devices to computer systems and software dependencies are increasing rapidly. This dependency over software can be hazardous if it can cause other components to become hazardous.

The software of the onboard controllers is becoming more important as the automation and autonomy of the train operation has become more important. Therefore, the influence of the software on the entire train control system is also increasing [2]. On-board software has become increasingly sophisticated due to the rapid development of the microprocessor technology, and the programming languages used are also advanced high level languages.

The development of TCS software size and complexity is comparatively slower than the hardware development speed, but it is expected to gradually increase in size and complexity. As such, from the initial mechanical and manual vehicle signal system to the latest unmanned automatic train control system, various controller devices have begun to be used as on-board devices, and it has become important to verify the safety of the software installed in these devices [3]. Such devices require utmost care in their specification, design; implementation and maintenance because not adhering to these may cause injuries or loss of lives and in turn may result in financial loss.

Software safety is mainly achieved by performing safety activities at the software design stage, which is the initial stage of software development. Typical safety activities include Preliminary Hazard Analysis (PHA), Hazard & Operability Analysis (HAZOP), Fault Tree Analysis (FTA), Failure Modes, Effects and Criticality Analysis (FMECA) [4]. While these techniques are in operation since the beginning of software development, additional safety checks generally are not formalized after the development has been completed.

Recently, a model-based software development methodology [5] that implements a software model and verifies the safety of the software through model validation [5] is attracting attention as a key technique for improving software safety. However, this method remains the biggest challenge to ensure the accuracy of the model. Development tools supporting model-based development include Esterel Studio and SCADE Studio from Esterel Technologies, Rhapsody from I-Logix, Simulink and Stateflow from Mathworks Inc, Rose Real-Time from Rational [6]. However, these tools are not tools to verify the security of software but rather tools to support software development. To be able to test for software security, we need to find the qualities which shape the software architecture. There are three qualities which find their role in decision to shape the software architecture for safety-critical, real-time systems are availability, reliability and robustness.

In this study, we propose a framework based on the McCall's software quality model that specifically identifies the criteria corresponding to software safety of train control system software. The significance of this study is important. The proposed framework in this study can professionally verify the safety of software, unlike existing software development tools. Existing software development tools are development support tools for reliable software development rather than tools for evaluating safety. Also, it is designed to be developed as an authentication tool related to software safety. Therefore, in this study we have analyzed related international standards and derived evaluation measures that can be automated among various requirements required by international standards. Once the tool is developed from this proposed framework, it is expected that it can be used for the software safety verification. The composition of the presented paper is as follows.

Section II gives introduction to software quality models which provide basic quality criteria for proposed framework. Section III explains about derived testing techniques for software safety evaluation framework. Section IV describes the architecture of proposed software safety assessment framework and major testing components that can be implemented. Section V concludes the paper.

II. SOFTWARE QUALITY MODELS

Since the last three decades software quality has received widespread attention within the software engineering community. There have been two remarkable models of software quality [7]. Both McCall and Boehm have described quality using a decompositional approach [8][9].

A. *McCall's Software Quality Model*

The McCall's model identifies and determines software product quality by addressing three perspectives: (i) Product Operation – It is the product's ability to be understood quickly, working and capable of delivering the desired results by the user. It also covers reliability, accuracy, efficiency, integrity and ease of use. (ii) Product Revision – It is about the ability of the product to review changes, including error detection and correction. It also covers the maintenance, flexibility and testability. (iii) Product transition – It is the ability of the product to adapt to new environments, distributed processing, together with the rapid change in hardware. This framework is useful for an integrated approach to quality. In this context, we classify software quality attributes into the hierarchy of three levels. At the top level are the so-called "quality factors" from the point of view of customers or users: reliability, precision and efficiency, integrity and ease of use, maintenance and testability, flexibility, portability, reusability and interoperability [10]. At the second level, these are "quality standards", which represent technical concepts. At the third level, "quality standards" measure the attributes of software products.

B. Framework for TCCS

McCall's quality model is modified to address software safety [11]. Based on it, software safety model is proposed by Ben Swarup Medikonda et. al [12], which includes six quality criteria described below:

Q_1: System hazard analysis
Q_2: Completeness of requirements
Q_3: Identification of safety critical requirements
Q_4: Design based on safety constraints
Q_5: Run-time issues management
Q_6: Safety critical testing

A set of lower level quality metrics can be derived from the above criteria, which can be measured directly.

The evaluation of software safety is done by verifying that the developed software satisfies the level of software safety integrity level (SSIL) given at the time of software design. SSIL is not defined by the software itself, but is determined to be equal to the Safety Integrity Level (SIL) of the system to which the software is applied. However, if it is decided to prevent software errors from propagating to the system, it can be set at a lower level.

SSIL is classified into 5 levels according to the risk of the system as follows.

TABLE I. SSIL LEVELS

SSIL Level	Qualitative Consequence
4	Potential for fatalities in the community
3	Potential for multiple on-site fatalities
2	Potential for major on-site injuries or a fatality
1	Potential for minor on-site injuries
0	Non-safety rating

The software development process proposed in IEC62279 [13] consists of the development process and the verification process as shown in Figure 1. The standard provides the requirements to be satisfied at each stage of development. The use of automated testing tools is recommended to measure the quality metrics while applying techniques to testing.

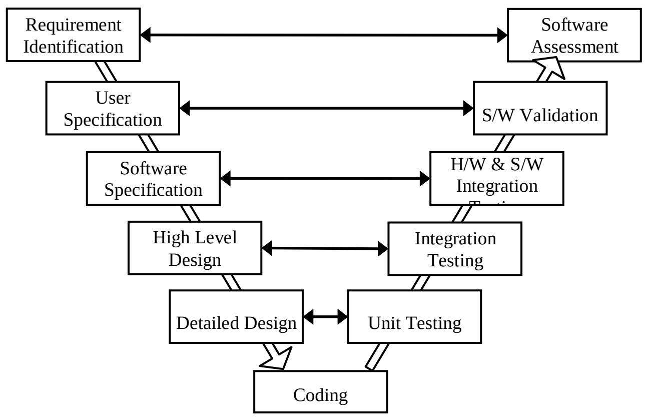

Fig. 1. Software Development Life Cycle in IEC61508-3

III. TESTING TECHNIQUES FOR SOFTWARE SAFETY EVALUATION FRAMEWORK

This section introduces the selected testing techniques for implementation in the safety evaluation framework among the safety verification requirements defined in IEC61508 and IEC62279 [2]. In this paper, we propose a framework that can be implemented as an automated tool among the verification methods.

In order to derive the testing techniques to be applied, the six steps ST1 to ST6 are derived, which are related to the steps related to the actions to be verified after the implementation of the software is written or created.

TABLE II. DERIVED TESTING TECHNIQUES

Testing Steps	Quality Criteria	Testing Technique											
		T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	T_{12}
ST_1	Q_6	Y	Y	Y	Y	Y	Y	N	N	N	Y	N	N
ST_2	Q_5	Y	Y	Y	Y	Y	Y	N	N	N	Y	N	N
ST_3	Q_3	Y	Y	Y	N	Y	Y	N	N	N	N	N	N
ST_4	Q_2	N	Y	Y	N	Y	Y	N	N	N	N	N	N
ST_5	Q_4	Y	Y	Y	N	Y	Y	Y	Y	N	N	N	N
ST_6	Q_1	N	Y	Y	N	Y	Y	Y	Y	Y	Y	Y	Y

T1: Performance Testing	T7: Fagan Inspection
T2: Boundary Value Analysis	T8: Symbolic Execution
T3: Equivalence Class Testing	T9: Checklist
T4: Design & Coding Standard	T10: Metrics
T5: Control Flow Testing	T11: Decision Table
T6: Data Flow Testing	T12: FTA

ST_1: Software module testing stage
ST_2: Software integration testing stage
ST_3: Integration stage between hardware and software
ST_4: Software validation stage
ST_5: Software change validation stage
ST_6: Software evaluation stage

Each step extracts information that includes measures for software evaluation from the software development stages, which is directly related to the train control system software which requires, automated testing during major software development stages. Major software testing techniques to be applied are defined according to the level of safety integrity of the software and quality criteria incorporated in that stage.

Table 1 shows the 12 core testing techniques derived in this way. Table 1 specifies the applied quality criteria and test techniques that can be applied to each stage of software development. For example, in software module testing stage, safety critical testing is the criteria for quality and for this Performance Testing, Boundary Value Analysis, Equivalence Class Testing, Design & Coding Standard, Control Flow Testing, Data Flow Testing are the derived testing techniques.

The performance test is to perform the hardware processing capability and the resources required in software implementation which is performed in the form of dynamic testing. Control and data flow testing tracks the control flow and data flow generated by the software to test whether unused code or data areas exists or not. Applying these derived test techniques provide the software test metrics which can be measured directly to be used in generation of automatic test data.

IV. Architecture of Software Safety Assessment Framework

This section describes the architectural design of the software safety evaluation framework. The train control system software safety evaluation technique consists of an automatic test case generator, an automatic test execution and monitor, and a target testing agent. Since the train control system has characteristics of the embedded control system, the structure of the S/W test tool to be tested and monitored through the testing agent program of the actual target board in which the application software is ported should be designed. Therefore, the test tool converts the safety analysis data of the evaluation target software by using the source code and the input data conversion module, receives the input data, and generates the test data automatically based on the input source code and the safety analysis data. Generate test data and scenarios using generation module. The generated test cases are automatically executed and tested, and the test results are analyzed by the monitoring module and the target testing agent, and the result is stored as a screen and a file. Figure 2 shows the use of the proposed safety testing framework. The test framework accepts the source-code and input safety analysis data of evaluation target software. Using the input data conversion module it generates the test data automatically based on the input source code and the safety analysis data.

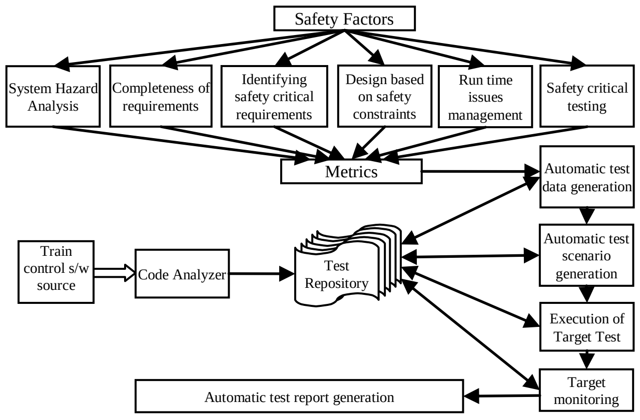
Safety Factors
System Hazard Analysis
Completeness of requirements
Identifying safety critical requirements
Design based on safety constraints
Run time issues management
Safety critical testing
Metrics
Automatic test data generation
Train control s/w source
Code Analyzer
Test Repository
Automatic test scenario generation
Execution of Target Test
Automatic test report generation
Target monitoring

Fig. 2. Train control software safety evaluation tester

The main functional definition of the testing framework is as follows.

- **Code Analyzer**: Generates function information, type information, control flow, and call information between functions through program analysis.
- **Create test scenarios**: Automatically generate test scenarios. It also allows users to create additional test scenarios.
- **Test data generation**: Appropriate test cases are generated based on selected test criterion.
- **Driver creation**: The driver that connects the test target code to the test engine and the program to be tested are created.
- **Execution**: Performs tests, summarizes test coverage, breakdown of sections, and test results. Provide the location of the error and have the ability to report detailed results to the user by test case.
- **Test report generation**: Generates reports based on the options for all test information and results.

Conclusion

In this paper, a safety evaluation framework has been proposed for train control system software. The proposed evaluation framework extends the existing automated software test tool and uses the results of the safety activity derived from the software development cycle as inputs to test the evaluation items required by the standard in a dynamic test form. It includes the core evaluation items required by the international standard and makes it available during the software development lifecycle. We also added the ability to continuously verify the safety by using the results of the safety activities performed at the software design stage as input to the testing tool.

It is expected that if the embedded software test tool with the proposed structure is developed, it will help to evaluate the software safety of the train control system.

References

[1] D. L. Parnas, A. J. van Schouwen, and S. P. Kwan, “Evaluation of safety-critical software”, Commun. ACM vol. 33, pp. 636-648, 1990.

[2] J.G. Hwang, H.J. Jo & H.S. Kim, "Design of automatic testing tool for railway signalling systems software safety assessment", WIT Transactions on Information and Communication Technologies, vol. 39, pp. 513 - 522, 2008.

[3] A. Zimmermann, G. Hommel, "A train control system case study in model-based real time system design", Proc. International Workshop on Parallel and Distributed Real-Time Systems (WPDRTS03), Nice, France, 2003.

[4] H.C. Liu, L. Liu, Q.H. Bian, Q.L. Lin, N. Dong, & P.C. Xu, "Failure mode and effects analysis using fuzzy evidential reasoning approach and grey theory", Expert Systems with Applications, vol. 38, pp. 4403-4415, 2011.

[5] R. France and B. Rumpe, "Model-driven development of complex systems: A research roadmap", Future of Software Engineering 2007, IEEE-CS Press, pp. 37-54, 2007.

[6] D. Harel, "Statecharts: a visual formalism for complex systems", Science of Computer Programming, vol. 8, pp. 231–274, 1987.

[7] J.P. Cavano & J.A. McCall, "A framework for the measurement of software quality", In ACM SIGMETRICS Performance Evaluation Review, vol. 7, No. 3-4, pp. 133-139, 1978.

[8] F. Norman, and J. Bieman, "Software metrics: a rigorous and practical approach", CRC Press, 2014.

[9] A. Rawashdeh & B. Matalkah, "A new software quality model for evaluating COTS components", Journal of Computer Science, vol.2, no. 4, pp. 373-381, 2006.

[10] J. A. McCall, "Factors in Software Quality: Preliminary Handbook on Software Quality for an Acquisiton Manager", Information Systems Programs, General Electric Company, 1977.

[11] R. Singh, “A Systematic Approach to Software Safety”, Proceedings of Sixth Asia Pacific Software Engineering Conference (APSEC), Takamatsu, Japan ,1999.

[12] M.B. Swarup & P.S. Ramaiah, "A software safety model for safety critical applications", International Journal of Software Engineering and Its Applications, vol. 3, no. 4, pp. 21-32, 2009.

[13] K.D. Shim & J.W. Lee, "Software Quality Assurance Activities of Automatic Train Control System to meet Requirements of the IEC 62279 Standard", Journal of the Korean society for railway, vol. 13, no. 4, pp. 412-418, 2010.