# MODEL-BASED TESTING FOR SAFETY-CRITICAL SOFTWARE OF RAILWAY SIGNAL

**CHUNFENG HU[1], YADONG ZHANG[1*], JIN GUO[1], WUDONG YANG[1], HEZE ZHAO[2]**

*[1] The School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China*
*[2] The communication workshop in Chawu, Communication section of Taiyuan railway bureau, Beijing, China*
**Corresponding author's email: ydzhang@swjtu.edu.cn*

***Abstract:*** *The safety of train operation is affected directly by railway signal safety-critical software, which shoulds be tested strictly and completely. Model-based software testing is emerged as a promising way for software analysis. This paper introduces the characteristics and difficulties of high-speed railway signal software in testing, and the scheme of model-based testing for safety-critical software of railway signal system. Several stages of the methods including analyzing the requirements of system to be tested, establishing abstract models, generating abstract test cases, concretizing abstract test cases, performing specific test and analyzing test results are introduced. An instance of using this method to test track circuit receiver (TCR) is described, and then the problems and characteristics for testing more complex systems with this approach are also proposed.*

***Keywords:*** *railway signal, safety-critical software; model-based testing*

## I. INTRODUCTION

China has the world's longest High-Speed Rail (HSR) network. As of September 2016, the HSR network has over 20,000 kilometers of route in service, which is more than the rest of the worlds' high-speed rail tracks combined [1]. A lot of signal software are used in HSR, and these software is safety-critical. In safety-critical field, such as aerospace, aviation and railway signal system, software plays a critical role for system safety, whose any failure may cause serious accident and disaster. This sort of software is called safety-critical software, and has to be tested by a strict and comprehensive way before putting into practice. Model-based testing has been became an important technology to assure software quality. Due to the features of improving testing automation and efficiency, easier to detect faults and generating test cases that can be reused, this method has been widely used in many fields, such as electric power industry, internet technology, automobile industry.

In this paper, we introduce the scheme of model-based testing for safety-critical software. Furthermore, and we present the problems and characteristics for testing more complex systems with our approach.

The rest of the paper is organized as follows. Section II introduces signal software used in HSR. Section III analyzes our scheme of model-based testing for safety-critical software of

railway signal. Section IV describes an instance of using this method to test TCR. Section V summarizes characteristics of the methods and presents future work.

## II. SIGNAL SOFTWARE USED IN HSR

Signal software used in HSR controls communication between trains, tracks, stations and signals. What is more, a software error can cause a signal to control confusion, or even a train crash, and they are all safety-critical software. Therefore, it is necessary to strictly test these railway signal software. However, before testing, we must first understand the characteristics and difficulties of the signal software used in HSR to be tested.

The safety software of railway signal has four characteristics, as follows.

1. The software is safety-critical and has to be highly reliable.

2. The software is usually embedded with hardware.

3. Unlike regular software, safety-critical software is expected to respond in real-time. While trains are running with a high speed, any response latency could lead to train crashes.

4. The signal software is only deployed on specific places such as rails and trains.

Therefore, only specific users operate the software. The safety-critical software of railway signal has the responsibility of controlling the operation of the train and ensuring the safety of the interval, which is very strict in terms of real time and safety. Due to those characteristics, these testing particularities of railway signal safety-critical software are caused. When testing safety-critical signal software, we need to pay attention to four aspects.

First, setting up a special test environment is required. On a test platform, we may need to either use real equipment or simulate real objects such as trains.

Second, generating test inputs can be complex. When testing embedded software, testability is low. That is to say, not every interface can be accessed. Moreover, tests include not only regular system actions, but also timing requirements. We must test the software in different contexts.

Third, unlike normal IT companies that test software by them, safety-critical software requires a third-party to verify before deploying the software into the high-speed rails.

Last but not least, same software system could be provided by different vendors. We need to make sure we can test the software from different vendors, with the same testing method.

## III. SCHEME OF MODEL-BASED TESTING FOR SAFETY-CRITICAL SOFTWARE

The introduction of the model for safety-critical software in railway signal is conducive to create more understandable and complete test cases, and simplifies the communication process within development teams. The graphical way to generate test cases can improve the quality and

efficiency of testing, and easily understand the software requirements and its impact of changes. Figure 1 shows a typical process of model-based for safety-critical software testing in railway signal.

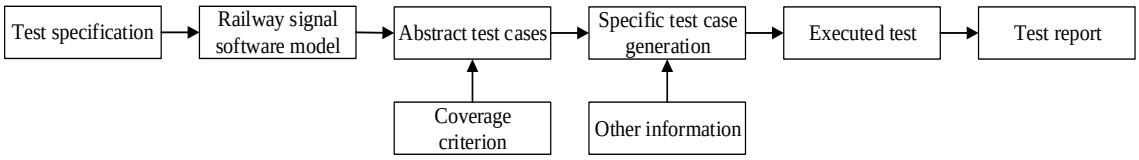


***Figure. 1.*** *Model-based for Software Testing Process*

The process is consisted of 5 stages: analyzing the system to be tested, establishing abstract models, generating abstract test cases, concretizing abstract test cases, performing specific test and analyzing test results.

**1. Analyzing the system to be tested**

In this stage, the system characteristics of the safety-critical software of railway signal are analyzed. According to these characteristics, the appropriate model is selected, which is used to generate test cases. Requirements specifications of software is usually used to analyze the software in railway signal. The optional types of modeling methods are finite state machines, UML (Unified Modeling Language), and so on.

**2. Establishing abstract models**

After determining the characteristics of the railway signal safety-critical software and the appropriate model, we can establish the corresponding model to reflect the demand of function. Moreover, we can further analyze whether the model is suitable for the safety-critical software of railway signal in the process of establishing model. The selection and establishment of the model for the safety-critical software of railway signal may be an iterative process. Only when we fully understand the characteristics of the safety-critical software of railway signal and the characteristics of each model, we can establish the most appropriate model for the safety-critical software of railway signal. This stage requires the modeler to be more familiar with the safety-critical software of railway signal and to be able to express the requirements of the safety-critical software as detailed as possible and without two meanings. The model is the basis of the subsequent test analysis in model-based for safety-critical software testing in railway signal.

Finite state machine including FSM (Finite State Machine), EFSM (Extended FSM), State machine, uses the trigger event and state transition to describe the behavior of the system. In terms of UML model, it is mainly focused on the UML state diagram, sequence diagram. However, other diagrams (such as deployment diagram, component diagram, etc.) are used less. In state diagram, status indicates data input timing or location, and transition conditions point out the changes of status, specially, inputs and the current state that determine target state and output.

### 3. Generating abstract test cases

After the abstract model for the safety-critical software of railway signal is built, abstract test cases can be generated. However, there may be infinitely many possible test cases in the process, so we have to choose some criteria. Assume that the selected model is UML state graph. First, we need to obtain the modeling data from information module, and identify some model elements, such as edges, nodes, regions, and so on. And then, testers select the corresponding selection criteria, combine with the model and coverage criteria to generate abstract test path, which is a series of nodes in the model for the safety-critical software in railway signal and the test case sequence in turn. Finally, the input interface information is located according to the corresponding transfer condition to decide the abstract test cases.

### 4. Concretizing abstract test cases

In this phase, the abstract test cases are transformed into executable specific test cases. In terms of UML state diagram, other information that includes abstract test values and data mapping applies the combination mode to achieve the process that the abstract test cases convert to specific final executable test cases.

### 5. Performing testing and analyzing testing results

Through executing the test cases generated, the test results are recorded and kinds of software problems are identified, and analyze the cause of the problem and take corrective actions. Finally, according to this analysis results, the safety of the railway signal safety-critical software could be evaluated, and a test report is presented to point out the mistakes and puts forward the improvement suggestions.

## IV. CASE STUDY

TCR is one of the Safety-Critical Software of Railway Signal. We will use TCR to illustrate our approach. Figure 2 shows the structure of TCR. TCR consists of hardware and software. The core component of the hardware is a digital signal processor (DSP). The TCR software is embedded in DSP. The only inputs to DSP are analog signals and time. First, TCR receives an analog frequency-shift keying (FSK) signal from the circuit installed on tracks. A FSK signal is an analog carrier signal with data. Second, TCR converts the FSK signal into a digital signal. Third, TCR extracts input parameters from the digital signal. Fourth, TCR processes the parameters in the DSP. Last, the output driver generates a signal to control the relay to be up or down [1].

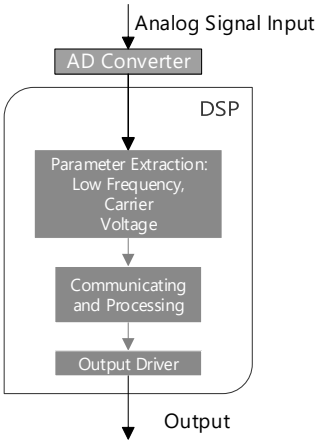


***Figure. 2.*** *The structure of TCR*

The train control system detects the presence of the train, and then sends an analog signal to TCR to control the relay to be up or down. Through our analysis, the input of the TCR software is made up of a signal and the duration of the signal that lasts, and the signal is composed of three parameters, *low frequency* (LF), *carrier frequency* (CF), and *voltage* (VO). Table 1 shows specification of parameters LF, CF, VO.

***Table. 1.*** *Specification of parameters LF, CF, VO*

| Parameters | Valid Range | Invalid Range |
|---|---|---|
| LF | $f_l$−0.3$Hz$≤$f$≤$f_l$+0.3$Hz$ | $f$<$f_l$−0.3$Hz$&$f$>$f_l$+0.3$Hz$ |
| CF | $f_c$−1.1$Hz$≤$f$≤$f_c$+1.1$Hz$ | $f$<$f_c$−1.1$Hz$ $f$> $f_c$+1.1$Hz$ |
| VO | (0V,7V] | (7V,10V] |

With combinatorial testing (CT), we identified parameters that constitute the signal (an input to TCR) and created Input Domain Models (IDMs) [1, 6]. While waiting for the action of the relay, TCR may receive more signals. Therefore, we created UML FSM to model the system behaviors in the presence of multiple signals, with the consideration of timing. Figure 3 shows FSM for relay of TCR from down to up.

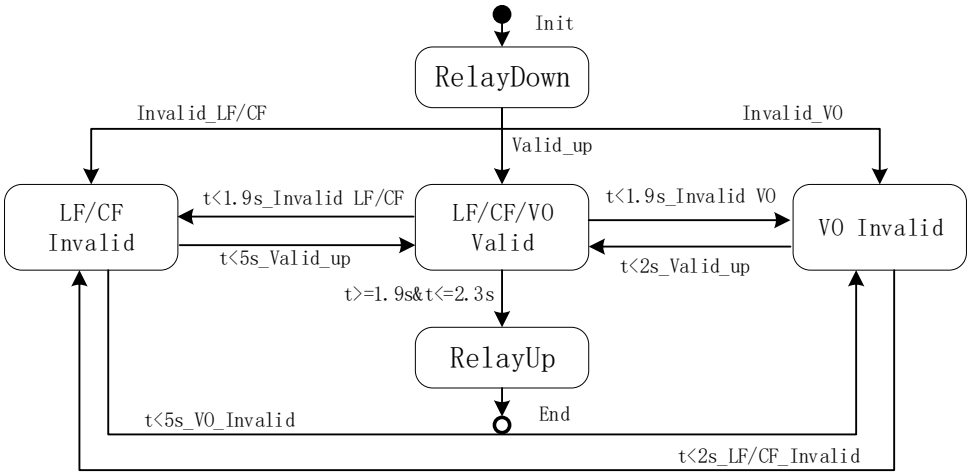


***Figure. 3.*** *FSM for relay of TCR from down to up*

This FSM models different scenarios in which the relay could go up. When we pass three types of signal to TCR: Invalid_VO, Invalid_LF/CF, and Valid_up, TCR will be in VO Invalid, LF/CF Invalid, or LF/CF/VO Valid state, respectively. The relay is down in the VO Invalid, LF/CF Invalid, and LF/CF/VO Valid states. When TCR is in one of these three with no additional signals received, TCR will stay with the relay down or reach the RelayUp state after a few seconds. For example, if TCR is in LF/CF/VO Valid, it will reach the RelayUp state in between 1.9 to 2.3 seconds.

In IDMs, we partitioned the input domain of each parameter, based on their relation to important values, which include the standard values, lower and upper bounds of the standard

values, and boundaries of invalid values. We applied a combinatorial coverage criterion (pair-wise coverage) to the IDMs, generating a suite of tests. On the other hand, We used a Model-Based Testing (MBT) tool, Skyfire 2, to generate test paths from the FSMs. These test paths are called abstract tests since they cannot be executed.

The next step of MBT is to concretize the abstract tests, assigning actual test values to the signal transitions. For each transition in an abstract test, we selected signal values from the first test suite generated from the IDMs. Next, we treated each signal as an independent parameter and applied pairwise coverage to them. Each test represents a series of signals.

After executing these test cases, we have some conclusions and the test results are shown in a table below. The highlights of the results are that we generated fewer tests but found four new faults in a released version of TCR. Table 2 shows the overall approach comparison.

***Table. 2.*** *The overall approach comparison*

| Metrics | Our Tests | Original Tests |
|---|---|---|
| Requirements coverage | All | Part |
| Applying MBT | Yes | No |
| No. of tests | 13,971 | 32,840 |
| New faults found | 4 | 0 |

## V. CONCLUSIONS

The main problem of model-based software testing in railway signal field is that the requirements for safety increase greatly the complexity of software models. The main indexes to evaluate the test methods are the size of the test set and the ability of fault detection. Therefore, model-based methods for safety-critical software testing in railway signal hope to solve the problem how to use models accurately describe the safety-critical software to be tested to generate test sets that meet the demand of all tests and have strong enough fault detecting ability.

The model-based method for safety-critical software testing in railway signal has high efficiency in testing, which is irreplaceable to substitute for other testing methods. Of course, it does not replace existing testing techniques since it is unable to just depend on this way for all faults in safety-critical software of railway single. This method has been widely used in railway signal by research team, such as ZPW2000, ATP, and Train Control Centre. For example, the lines of code of and the number of functionalities of ATP is about 10 times more than that of TCR. The number of documented test scenarios of ATP is about 15 times more than that of TCR [6]. Specially, some problems were found and obtained confirmation from software provider. We plan to study the algorithm of test case generation, in order to generate more compact test cases with guaranteed coverage.

*(Continuously see page 77)*

method using extended Kalman filter for attitude and heading reference system". Position, Location and Navigation Symposium, USA, 2008.
[3]. W. Premerlani and S. Balutal, "DCM tutorial - an introduction to orientation kinematics", Stalino Electronics, USA 2009.
[4]. G. Baldwin, R. Mahony, J. Trumpf, T. Hamel, and T. Cheviron, "Complementary filter design on the special euclidean group SE(3)". European Control Conference (ECC), 2007, pp. 3763–3770.
[5]. R. Mahony, T. Hamel, and J. M. Pflimlin, "Nonlinear complementary filters on the special orthogonal group". IEEE Transactions on Automatic Control, vol. 53, no. 5, pp. 1203–1218, 2008.
[6]. J. Hartikainen, A. Solin, and S. Sarkka, "Optimal filtering with Kalman filters and smoothers - A manual for the matlab toolbox ekf-ukf," GNU, FINLAND, 2011 ♦

---

## MODEL-BASED TESTING FOR SAFETY-CRITICAL …

*(Following page 83)*

**ACKNOWLEDGEMENT**

This research was supported by National Natural Science Foundation of China (Grant No. 61703349), Key Research Projects of China Railway Corporation (Grant No. 2017X007-D), Fundamental Research Funds for the Central Universities (Grant No. 2682017CX101, 2682017ZDPY10) and Opening Foundation of Gansu Provincial Key Laboratory of Traffic Information Engineering and Control (Grant No. 20161103).

**References**

[1]. C. Rao, J. Guo, N. Li, Y. Lei, Y. Zhang, Y. Li, and Y. Cao, "Applying combinatorial testing to high-speed railway track circuit receiver," in 2017 6th Internaltional Workshop on Combinatorial Testing (IWCT2017),Tokyo, Japan, March 2017, pp. 199–207.
[2]. A. Petrenko, A. Simao, and J.C Maldonado, "Model-based testing of software and systems: recent advances and challenges," International Journal on Software Tools for Technology Transfer, no.14, pp.383-386, June 2012.
[3]. T.S Chow, "Testing design modeled by finite-state machines," IEEE Transactions on Software Engineering, vol.4, no.3, pp.178-187, May 1978.
[4]. S.H. Li, J. Wang, Y.C. Qi, "Model-Based Methods for Real Time System Testing," COMPUTER ENGINEERING & SCIENCE, vol.28, no.4, pp.119-123, 2006
[5]. J. Yang, J. Wang, H.W Chen, "A review of Model-Based for Software Testing," COMPUTER SCIENCE, vol.31, no.2, pp.184-187, 2004
[6]. C F Hu, J Guo, N Li, et al. Towards Effective and Scalable Testing for Complex High-Speed Railway Signal Software[C]// IEEE International Conference on Software Quality, Reliability and Security Companion. IEEE, 2017:571-572.
[7]. N. Li, A. Escalona, and T. Kamal, "Skyfire: Model-based testing with cucumber," in 2016 IEEE International Conference on Software Testing, Verification and Validation (ICST), April 2016, pp. 393–400.