TTCN-3 Testing of Hoorn-Kersenboogerd Railway Interlocking

Jens R. Calamé
CWI, Amsterdam
The Netherlands
e-mail: jens.calame@cwi.nl

Nicolae Goga
Technical University Eindhoven,
Eindhoven
The Netherlands
e-mail: n.goga@tue.nl

Natalia Ioustinova, Jaco van de Pol
CWI, Amsterdam
The Netherlands
e-mail: {ustin,vdpol}@cwi.nl

Abstract

Railway control systems are safety-critical, so we have to ensure that they are designed and implemented correctly. Testing these systems is a key issue. Prior to system testing, the software of a railway control system is tested separately from the hardware. The interlocking is a layer of railway control systems that guarantees safety. It allows to execute commands given by a user only if they are safe; unsafe commands are rejected.

Railway interlockings are central to efficient and safe traffic management for railway infrastructure managers and operators. European integration requires new standards for specification and testing interlockings. Here we propose an approach to testing interlockings with TTCN-3 and give an example for its application.

The code of interlockings is simulated during test execution. For assessing the quality of the tests, we propose an approach inspired by the Classification Tree Method.

Keywords *— Railway interlocking, TTCN-3, test coverage*

1 Introduction

Railway control systems, and here especially interlockings, are safety-critical. For this reason, testing is a key issue to ensure the correctness of their design and implementation. A special point about testing these systems is, that the software is tested separately from the hardware in a simulated environment.

Railway interlockings are central to efficient and safe traffic management for railway infrastructure managers and operators [6] and new standards for the specification and test of these interlockings are required by the European integration. In this paper, we propose an approach to testing interlockings with TTCN-3. Together with engineers from ProRail, a Dutch railway management company, we have applied this approach to testing the interlocking of the train station Hoorn-Kersenboogerd in North-Holland.

Starting from the general requirements for safety, reliability, maintainability etc. (CENELEC standards EN 50126, 50128, 50129) for railway control systems, we have developed a test suite for testing the interlocking of Hoorn-Kersenboogerd station. The standard requirements are formulated for a station with a general configuration. All requirements are of the form: initial situation, action, expected results. To develop test cases for the Hoorn-Kersenboogerd station, we had to (1) map the general configuration to a particular configuration of the station; (2) map the initial situation to the stimuli for the system under test (SUT); (3) map the final situation to the output values expected from the SUT; (4) define default values for objects of the station that are not involved in the tested situation but still can influence it; (5) formulate time requirements for tested actions. We specified the test cases in TTCN-3, which is a language for specification and automated execution of test suites. The language, its operational semantics, the general structure and interfaces of a TTCN-3 test system are standardized by ETSI [5]. Originally developed for testing telecommunication system, TTCN-3 supports real-time testing. A TTCN-3 test executable has predefined standard interfaces that allow to offer TTCN-3 solutions that do not depend on the implementation details of an SUT. In the TT-Medal project [10], we applied TTCN-3 to testing railway interlockings.

The software part of a railway interlocking is a program that consists of a large number of guarded assignments. The program defines a control cycle that is repeated by the system. The length of the control cycle is fixed by design. Although the environment of the system changes continuously, the system only sees snapshots of the environment made at the beginning of each control cycle. Thus the environment is discrete from the system's point of view. The system is timed, delays are used to guarantee safety.

We test the interlocking's software without the corresponding hardware by simulating the code of the interlocking during the test execution. To ensure repeatability of testing results, we need to establish control over time in the SUT and in the test system. We have to guarantee that the SUT and the test system agree on time. We propose a solution with simulated time where time is modeled as a discrete clock. Since TTCN-3 was originally developed for real time testing, we have to make extra efforts to implement simulated time in the TTCN-3 framework. Our solution for testing with simulated time [3] is based on an extension of Dijkstra's well-known algorithm for distributed termination detection [4].

Applying our approach to testing the interlocking of Hoorn-Kersenboogerd station, we cover the whole test-process starting from the development of test cases, proceeding with the implementation of the test system, and finally with the automated execution of tests and the interpretation of results. The approach allows detecting violations of safety requirements in interlocking software. The solution for testing with simulated time in TTCN-3 is also applicable to other systems with similar characteristics. To assess the quality of the test suite, we propose an approach inspired by the Classification Tree Method (CTM [8]).

This research is supported by the ITEA project *Test and Testing Methodologies for Advanced Languages* (TT-Medal, [10])

2 Vital Processor Interlockings

In this section, we first outline the general railway control system architecture. Then, we describe in more details one of the layers of the architecture, *Vital Processor Interlocking* (VPI) layer.

Railway control systems consist of three layers, the logistic, the interlocking and the infrastructure layer. The *infrastructure layer* represents a railway yard that is a collection of objects such as railway tracks supplied with signals, points and level crossings, the *logistic layer* provides the user-interface for experts in the railway control center. The *interlocking layer* connects both and is responsible for the safety of the whole system, i.e. it ensures that no train derailments or collisions happen. All commands between logistic and infrastructure are intercepted by the interlocking. The VPI checks whether it is safe to execute commands and rejects or suspends unsafe ones.

A VPI is a machine that executes hardware checks and executes a program. This program consists of a number of guarded assignments and specifies a control cycle that is repeated with a fixed frequency by the hardware. The program operates on three kinds of variables: read-only inputs, auxiliary variables for computations and writable output variables. Within one cycle, the input is read and some calculations on it are done. Shortly before the end of the time slice, the outputs are written and thus accessible by the infrastructure and logistic. Reading, calculating and writing belong to the active phase of the program, while there is an idle phase between calculation and output. A VPI is a timed system and a lot of safety requirements for it are timed. They often describe dependencies between objects of the infrastructure in time, like “when freed a train track should not become occupied for 120 seconds”. The time requirements are specified using delays.

The requirements are defined as scenarios, derived by ProRail and their third party manufacturers following the guidelines in the CENELEC standards (EN 50126, 50128 and 50129). These scenarios have a common structure, describing an initial situation, an action and the expected results. A scenario is not limited to one such description, but can consist of a sequence of situations with actions and results.

For example, the scenario *Cancel Yellow* describes the procedure to establish a route over certain segments in the station by setting a signal to *yellow* and afterwards setting this signal to *red*. The scenario is defined for an abstract railway yard consisting of six track segments (1,3,5,7,9,11) and 5 signals (2,4,6,8,10; see Figure 1). Below, we describe the scenario in more detail:

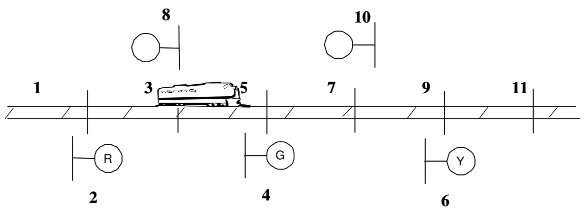

Figure 1: Cancel Yellow Station-independent setup

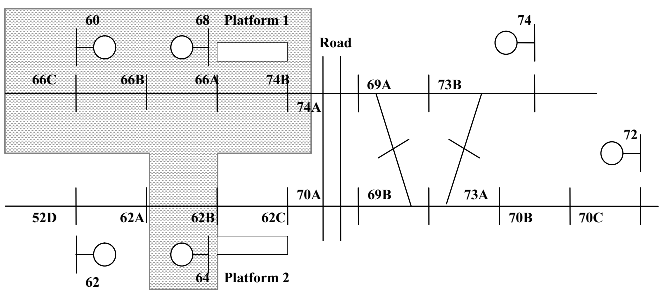

Figure 2: Configuration of station Hoorn-Kersenboogerd

No.	Initial situation	Action	Expected result
1.	Rest: No route has been set and there is no train.	Establish route 4 to 6.	The route from signals 4 to 6 is locked and signal 4 becomes *yellow*.
2.	Continuation	Cancel signal 4.	Signal 4 becomes red and the route is released without delay.

Note that only signals 2, 4 and 6 are influenced by this scenario. Further we provide a mapping from this general scenario for an infrastructure to a concrete scenario for our case study. Based on this mapping we then describe the case study itself.

3 Case Study for Hoorn-Kersenboogerd

The goal of the case study is to evaluate the applicability of TTCN-3 for testing VPIs. Here we apply TTCN-3 to test the VPI of the Hoorn-Kersenboogerd station. We first map selected scenarios for a general infrastructure to concrete scenarios for Hoorn-Kersenboogerd. Then we proceed with translating the concrete scenarios into TTCN-3 test cases, developing a TTCN-3 test system for testing the VPI and finally with executing the test cases and analyzing the test results. We illustrate this process for the *Cancel Yellow* scenario mentioned in the previous section.

Hoorn-Kersenboogerd is a railway station in the Netherlands. Its railway yard consists of two railways, one at each of the platforms, which are interconnected in both directions of traffic. Furthermore, both railways are crossed by a road. From the safety viewpoint, this situation requires the management of six signals in the area of the station, three signals per railway track.

The railway tracks at the station are subdivided into seven or nine segments, resp. Each of these segments is either in normal position, if there is no train passing this section, or in reverse position, if it is occupied by a train.

Depending on this information, the signals are set to stop (red light), slow ride (yellow light) or ride (green light).

We have selected three of the scenarios provided by ProRail, which were potentially applicable to Hoorn-Kersenboogerd: *Cancel Yellow*, *Normal Route* and *Normal Train Depart*. The scenario *Cancel Yellow*, which describes the procedure to establish a route over certain segments at the station by setting a signal to *yellow* and afterwards setting this signal to *red*, is discussed in further detail in this section.

To implement the test cases for this scenario regarding Hoorn-Kersenboogerd, we first map the affected tracks as well as the signals from the general scenario to the conditions at Hoorn-Kersenboogerd. The situation is highlighted in Figure 2 by selecting the railway track segments 66A/B/C and 74A/B and signals 60, 68 and 64. We provide the following mapping: the signals 4, 6 and 10 of the general scenario (Figure 1) are mapped to the signals 60, 68 and 64, and the segments 5, 7, 9, and 11 of the general scenario are mapped to the segments 66C, 66B/A, 74B and 74A of Hoorn-Kersenboogerd.

As the next step, we build a TTCN-3 test case reflecting the scenario. Its initial situation is mapped to corresponding values of input variables of the VPI program representing objects involved in the scenario. The expected situations are handled analogously and mapped to the values of the output variables of the VPI. The test case reflects the control cycle of the VPI: The general pattern is to send input values to the VPI, wait for the outputs and check whether the obtained outputs match the expected ones. Outputs not only depend on values of inputs but also on time. Some inputs have to remain the same for a certain time span in order to change the corresponding output values. Therefore, the test case takes several control cycles in order to reach the expected situation starting from the initial one. For instance the TTCN-3 test case implementing *Cancel Yellow* lasts for six cycles.

Since we cannot utilize the VPI hardware, we have to simulate the VPI in order to test it. Developing a simulator for one case study is too expensive. Therefore we provide a automatic translation of the VPI program into a specification in μCRL [2] for which we already have a simulator. The translation preserves all the behaviors of the VPI program. μCRL is a process algebraic language for specifying large distributed systems. In this language, recursive processes are specified, which invoke actions with data parameters. The invocation of actions can optionally be guarded by conditions.

To execute test cases, we developed a TTCN-3 test system. Time also led to another problem in test case implementation. Real-time execution of test cases was not possible due to the fact that the software for the railway interlocking is tested apart from its hardware. We discovered that real-time or scaled-time semantics were not efficient enough to be applied to this task, so we developed the approach of *simulated time* [3]. Technically, this approach introduces some components in the test system, which synchronizes the progress of simulated time during the test for all participating components. Therefore, not only time progression steps must be implemented, but also system idleness must be communicated to the additional timing components. This must be taken care of during test case implementation.

As a test execution engine we used the TTCN-3 tool TTworkbench [1] that runs the TTCN-3 test cases and communicates with the SUT via the test adapters, which we have implemented. Independently from each other, ProRail and we were able to find the same failure in the *Normal Train Depart* scenario. In the scenario, a signal should stay *yellow* at a certain point, but actually turned to *green* after one cycle. Previous works on verification of the VPI [9] have missed this failure. Since ProRail had found the failure as well, the actual VPI program running at Hoorn-Kersenboogerd does not contain it.

4 Testing coverage

For the computation of the coverage of a test suite we employ a methodology inspired by CTM [8]. CTM is usually used for the selection of test data, by defining equivalence classes for the data and using a uniformity hypothesis [7]: picking few representatives for a class is sufficient to check the whole equivalence class. This methodology naturally leads to a notion of a test coverage based on the percentage of equivalence classes that are checked by a test suite. We adapted a comparable methodology for our case study.

If the behavior of a VPI is specified in a formal language, formal methods can be used to check, which of the situations in infrastructure are reachable and which of them are impossible. However, we had no specification provided therefore we had to rely on the knowledge of experts in order to decide which situations in the infrastructure are possible and mapped them into combinations of inputs of VPI. This set of possible input combinations we further consider as the basis for calculating coverage of our test suite. Together with information about the points in time, in which several inputs are fed into the SUT, we can derive the number of possible traces through the system. The coverage is then the percentage of traces, which have actually been executed by the test suite.

Our methodology can be described as follows: Given a particular railway yard, we calculate (using experts' knowledge) all possible combinations of traces through the SUT, induced by the yard's objects' states together with possible inputs from logistic and their order. Further we refer to the number of these combinations as M. During test execution, we keep track of the traces covered by a particular test suite, referred to as n. The coverage provided a test suite TS is then computed using the formula

$$cov_{TS}(n, M) = \frac{n}{M}\,.$$

For our case study there are in total 80 possible combinations, from which we had checked, after performing three test cases, 10. The coverage is

$$cov(10, 80) = \frac{10}{80} = 0.125 = 12.5\% .$$

In our experiments, we observed that the coverage respects a natural property of the testing coverage, respectively the monotonic property by increasing after performing more tests. It can be also noted that many other combinations that are not checked by our test suite can easily be verified by arranging in a suitable way the inputs such that the desired combinations of inputs and input orders will occur. In this way the testing coverage can be easily increased.

A possible combination of inputs and timers corresponds to a set of behaviors. For example if a possible combination contains two inputs that are both 1, there will be at least three behaviors that can model that situation: one in which both inputs are 1 in the same time and two in which the value 1 of an input will precede in time the other value 1 of another input. The proposed coverage considers that checking at least one behavior from the correspondent set of behaviors will be sufficient to check all the behaviors from the set (consequently, the correspondent possible combination is also checked). This *uniformity* hypothesis is quite strong but makes the computations simple. For differentiating between different behaviors within the correspondent set, more sophisticated techniques for the computation of the coverage needs to be employed, such as test coverage and selection based on distances [7], but this is an option for further research.

5 Conclusion

In this paper, we proposed an approach to test railway control systems, in particular railway interlockings, with TTCN-3. We outlined the scenario mapping for our case study Hoorn-Kersenboogerd and discussed the results of the test experiment. Particularly timing was a non-trivial issue, since we could not test the railway interlocking in its original environment, but only in a simulated one. For this reason, time also had to be simulated, which formed a special challenge in this work.

Finally, we discussed how the test coverage for a railway interlocking test can be obtained to achieve an optimal selection of input values and timing them. The coverage measurement itself is based on the idea of CTM, while the actual selection of optimal test data values and timings is object of further research.

We showed that TTCN-3 can potentially be used as a standard language to specify test suites for railway applications. It is suitable to provide reusable platform independent test suites for railway interlockings. Clear semantics and high maintainability of TTCN-3 allow not only to provide high-quality test suites but also to reduce costs for testing and maintenance of test systems on the long run. This work was done within TT-Medal project (Test and Testing Methodologies for advanced languages) where TTCN-3 was also applied by a team of Daimler Chrysler to test automotive systems and by teams from NetHawk, Nokia and VTT to test modern telecommunication systems. More information about TTCN-3 and using it for various domains is available on the TT-Medal web-page [10]. We thank Fraunhofer FOKUS and Testing Technologies (`www.testingtech.de`) for providing us with TTCN-3 tools and support on TTCN-3.

References

[1] Testing Technologies. `www.testingtech.de`.

[2] S.C.C. Blom, W.J. Fokkink, J.F. Groote, I.A. van Langeveld, B. Lisser, and J.C. van de Pol. μCRL: a Toolset for Analysing Algebraic Specifications. In G. Berry, H. Comon, and A. Finkel, editors, *Proc. of the 13th Conference on Computer Aided Verification (CAV'01)*, pages 250–254. Springer-Verlag, 2001.

[3] S.C.C. Blom, N. Ioustinova, J.C. van de Pol, A. Rennoch, and N. Sidorova. Simulated Time for Testing Railway Interlockings with TTCN-3. In *Proc. of 5th International Workshop on Formal Approaches to Testing of Software (FATES 2005)*, Lecture Notes in Computer Science, 2005.

[4] E.W. Dijkstra, W.H.J. Feijen, and A.J.M. van Gasteren. Derivation of a Termination Detection Algorithm for Distributed Computations. *Information Processing Letters*, 16(5):217–219, June 1983.

[5] ETSI. *Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Parts 1 to 6.* ETSI ES 201 873, `www.etsi.org`.

[6] European Standards for Railways Interlocking Systems. `www.euro-interlocking.org`.

[7] N. Goga. *Control and Selection Techniques for the Automated Testing of Reactive Systems.* PhD thesis, Technical University of Eindhoven, 2004.

[8] M. Grochtmann and K. Grimm. Classification Trees for Partition Testing. *Software Testing, Verification and Reliability*, 3(2):63–82, 1993.

[9] J.F. Groote, J.W.C. Koorn, and S.F.M. van Vlijmen. The Safety Guaranteeing System at Station Hoorn-Kersenboogerd. In *Proc. 10th Annual Conference on Computer Assurance (COMPASS'95)*, pages 57–68, 1995.

[10] Test and Testing Methodologies for Advanced Languages (TT-Medal). `www.tt-medal.org`.