OETCS/WP4/D4.4V0.3

openETCS

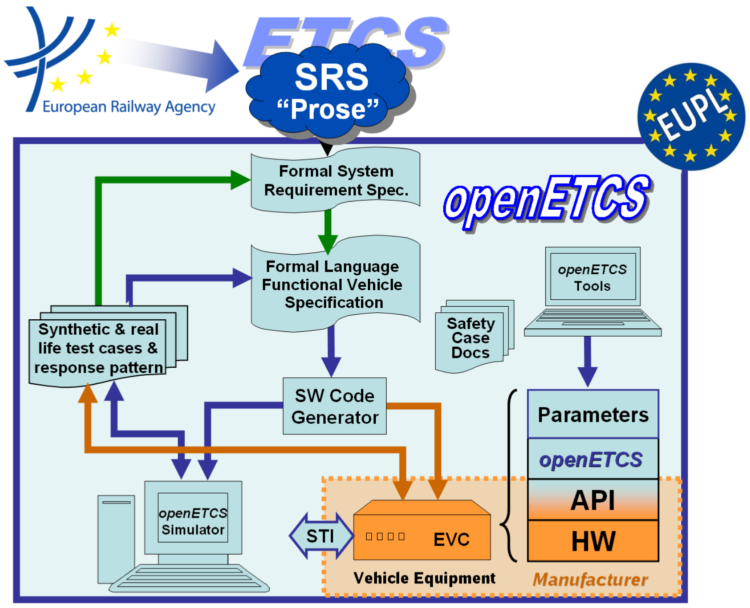
ITEA2 Project
Call 6 11025
2012 – 2015

Work-Package 4: “Verification & Validation Strategy”

openETCS Final Report on Verification and Validation

Hardi Hungar, Marc Behrens, Mirko Caspar, Michael Mönsters, Jan Peleska, Uwe Schulze, Marielle Petit-Doche, Stefan Rieger and Thorsten Schulz

December 2015



Funded by:

Federal Ministry of Education and Research

Liberté • Égalité • Fraternité RÉPUBLIQUE FRANÇAISE MINISTÈRE DE L'ENSEIGNEMENT SUPÉRIEUR ET DE LA RECHERCHE

INVESTISSEMENTS D'AVENIR

Région de Bruxelles-Capitale

GOBIERNO DE ESPAÑA MINISTERIO DE INDUSTRIA, ENERGÍA Y TURISMO

Work-Package 4: “Verification & Validation Strategy” **OETCS/WP4/D4.4V0.3**
December 2015

openETCS Final Report on Verification and Validation

Document approbation

Lead author:	Technical assessor:	Quality assessor:	Project lead:
location / date	location / date	location / date	location / date
signature Marc Behrens (Deutsches Zentrum für Luft und Raumfahrt e.V.)	signature [assessor name] ([affiliation])	signature Jan Welte (TU Braunschweig)	signature Klaus-Rüdiger Hase (DB Netz)

Hardi Hungar
DLR, main editing

Marc Behrens, Mirko Caspar, Michael Mönsters
DLR

Jan Peleska, Uwe Schulze
University Bremen

Marielle Petit-Doche
Systerel

Stefan Rieger
TWT

Thorsten Schulz
University Rostock

Final Report

Prepared for openETCS@ITEA2 Project

Abstract: This document summarizes the approach, scope and result of the verification and validation activities in the project openETCS.

Modification History

Version	Section	Modification / Description	Author
0.0	all	initial	Marc Behrens
0.1	all	revision and addition	Hardi Hungar
0.2	all	revision and addition	Hardi Hungar, contributions by partners
0.3	most	added sections for Institut Telecom, included Systerel contributions, added TBD comments	Hardi Hungar

Table of Contents

Figures and Tables

Figures

Tables

1 Introduction

According to [1, 3.1.48], verification is an activity to check whether the output of a development phase meets the requirements. This concerns formalities, traceability, and, w.r.t. the main content, completeness, correctness and consistency. Within openETCS, examples of each kind of verification have been performed. Thereby, also new methods and tools have been evaluated and adapted.

Validation concerns the compliance of the end result of the development with the user requirements. This has been done employing the demonstrator of the EVC software.

This document summarizes the activities described in more detail in separate reports. It explains how these separate activities fit into the development process of openETCS as defined in the deliverable D2.3a.

Most verification activities are actually reviews of documents (or even programs). For general review activities, a process has been defined in [2].

2 Verification and Validation in the Development Lifecycle

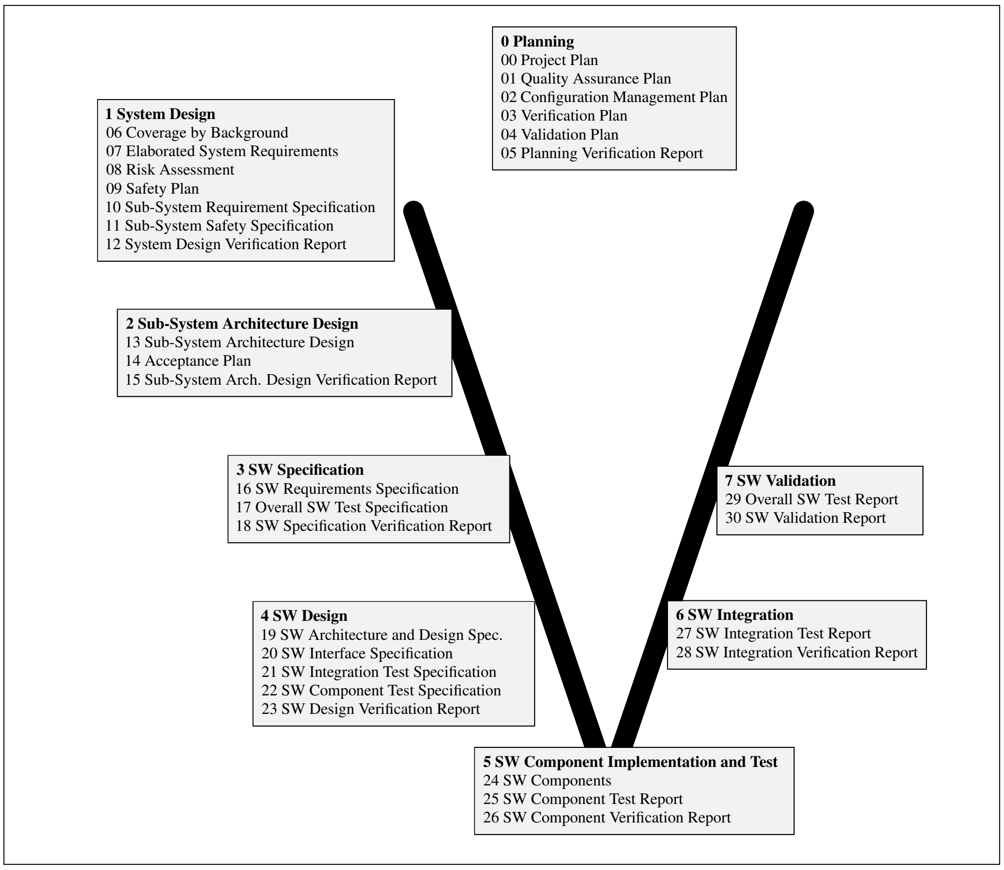

0 Planning
00 Project Plan
01 Quality Assurance Plan
02 Configuration Management Plan
03 Verification Plan
04 Validation Plan
05 Planning Verification Report
1 System Design
06 Coverage by Background
07 Elaborated System Requirements
08 Risk Assessment
09 Safety Plan
10 Sub-System Requirement Specification
11 Sub-System Safety Specification
12 System Design Verification Report
2 Sub-System Architecture Design
13 Sub-System Architecture Design
14 Acceptance Plan
15 Sub-System Arch. Design Verification Report
3 SW Specification
16 SW Requirements Specification
17 Overall SW Test Specification
18 SW Specification Verification Report
7 SW Validation
29 Overall SW Test Report
30 SW Validation Report
4 SW Design
19 SW Architecture and Design Spec.
20 SW Interface Specification
21 SW Integration Test Specification
22 SW Component Test Specification
23 SW Design Verification Report
6 SW Integration
27 SW Integration Test Report
28 SW Integration Verification Report
5 SW Component Implementation and Test
24 SW Components
25 SW Component Test Report
26 SW Component Verification Report

Figure 1. openETCS Development Lifecycle

Fig. 1 is an overview of the openETCS development lifecycle, taken from D2.3a. It depicts the process for a complete development of the EVC software, of which a part has been performed within the project. Verification, resp., validation, has to be done in each of the phases of the development.

3 Overview of Verification and Validation Activities

The verification and validation activities of openETCS fall in two categories.

- They may serve the purpose of supporting the development of the EVC SW. These are activities as defined in the process definition D2.3a, actual verification or validation of design artifacts.
- They may serve to demonstrate or evaluate methods or tools for V&V. Such methods or tools are applied either to available design artifacts, or some such artifacts are created specifically for the purpose of the demonstration/evaluation.

Both kinds of activities are reported about in this document. It is structured according to the phases of the development.

All subsubsectins:

- Add references to corresponding sections in D4.3.1 and D4.3.2
- Address maturity level of tools (¿methods too?)

3.0 Verification and Validation in the Planning Phase

There have been reviews of the planning documents. These activities are not reported in detail, here.

3.1 Verification and Validation in the System Design Phase

3.1.1 Verification of Chapter 5 of Subset 026 (TWT)

Contributing project partners

The work has been performed by TWT.

Process step

This activity is part of the verification of the Elaborated System Requirements which are based on Subset 026 [3]. It contributes to the System Design Verification Report (1-12). In formalizing and analyzing the procedures it findings contribute also to the definition of the Elaborated System Requirements themselves (1-07).

Object of verification

The object of verification are the procedures defined in Chapter 5 of Subset 026 [3, 5]. *NN* of the ¿25? procedures have been analyzed.

Available specification

The procedures are checked for consistency. They are not checked against an external specification.

Objective

The main objective w.r.t. verification is check the procedure definitions for consistency and some sanity conditions. A by-product are formalizations which can enter the Elaborated System Requirements (1-07).

Method/Approach

The control flow of the procedures is modeled with colored Petri nets (CPNs) in the tool [4]. Each model is checked independently by a second person. The necessity of formalization coming with the modeling uncovers inconsistencies in textual specifications. With the help of the simulation and checking facilities of the CPN tools, sanity conditions on the models are checked.

Means/Tools

The CPN tools are ...

Results

The modeling and analysis uncovered 36 inconsistencies, ambiguities and gaps in the Subset 026 which were reported in [5]. $\boxed{\text{to be revised/completed}}$

Observations/Comments

Conclusion

$\boxed{\text{Missing procedures?}}$ The numerous specification findings illustrate the need for validating the specification. CPNs are well-suited to model the behavioral aspects described in Subset-026 chapter 5. The size of the model clearly indicates the complexity of the procedures, even at the current level of abstraction. The main benefit comes from the activity of formalization itself, and of incomplete, but valuable, simulations.

3.1.2 Verification of Management of Radio Communication function (Systerel)

Contributing project partners

The work has been performed by Systerel

Process step

This activity contributes to:

- the Elaborated System Requirements (1.07)
- the Sub-System Requirement Specification (1.10)
- the System Design Verification Report (1.12)

Object of verification

The object of verification is the Event-B model for the communication establishing at `https://github.com/openETCS/model-evaluation/tree/master/model/Event_B_Systerel/Subset_026_comm_session`. It is from the strictly formal modeling phase and represents the communication session management of the OBU.

Available specification

The model implements the requirements for the communication session management as described in Subset-026 chapter 3.5.

This section describes the establishing, maintaining and termination of a communication session of the OBU with on-track systems.

Objective

One goal is the development of a strictly formal, fully proven model of the communication session management and to provide evidence of covering the necessary requirements of Subset-026 as well as proving correctness of the model wrt. the requirements and attaining a good coverage of the model wrt. the requirements.

The second goal is to correctly implement the applicable safety requirements identified by the safety analysis. Both functional and safety requirements should be traced in the model and a requirement document in a standardized format.

The formal model will represent the described functionality on the system level, the correct functioning can be validated by step-wise simulation and model-checking of deadlock-freeness.

Method/Approach

At first, the basic functionality described in the chapter 3.5 that are identified. These serve as basis for a first abstract model, which is refined iteratively, adding the desired level of detail. The elements of Subset-026 are traced using links from Event-B to the ProR file in ReqIf format. Requirements are formalized as invariants and proven where applicable.

Means/Tools

The means used are:

- open source Rodin tool (http://www.event-b.org/), including plug-ins (for details see https://github.com/openETCS/model-evaluation/blob/master/model/Event_B_Systerel/Subset_026_comm_session/latex/subset_3_5.pdf)
- ProR requirements modeling tool http://www.pror.org
- open source ProB model checker and B model simulator http://www.stups.uni-duesseldorf.de/ProB/index.php5/Main_Page
- open source CVC3 (http://www.cs.nyu.edu/acsys/cvc3/), verIT (www.verit-solver.org) and Alt-Ergo (http://alt-ergo.lri.fr) SMT solvers

All the tools are on class T1.

Results

- The result is a fully formal model of the communication session management as described in chapter 3.5 of Subset-026.
- Each implemented element of this section is linked to the ProR requirements file, both specification elements that describe how something has to be done, as well as requirements that describe what must be achieved.
- The model can be simulated / animated, either with the AnimB or the ProB plug-in, validating the functional capabilities.
- The safety requirements are formalized as invariants in predicate logic, their proofs are for the most part fully automatic.
- It was found that while the Subset-026 communication management explicitly allows multiple communication partners (see RBC handover), there is no explicit limit of established communication connections given in chapter 3.5.
- A complete covering of the elements of Subset-026 was not realized, e.g., there is a representation of the contents of a message, but its explicit format is not implemented. This is considered an implementation detail without influence for a system level analysis. In general, Event-B models will not be refined up to the implementation level.

See https://github.com/openETCS/validation/blob/master/Reports/D4.3/D4.3.1-Final-VV-repor D4.3.1.pdf and https://github.com/openETCS/validation/blob/master/VnVUserStories/VnVUserStorySysterel/04-Results/d-EventB-VnV/EventB-Rodin-VnV.pdf.

Conclusion

Having an abstract formal model of the implemented functionality which can be simulated, allows for interesting insights into the overall functioning of a system. Formalized requirements are very helpful in both the identification of ambiguous requirements and in their clarification.

The elements of Subset-026 are of very different nature. Some describe rather low-level specification details, other describe “real” requirements. Without an analysis as done with this Event-B model, it can be difficult to decide which elements must be considered on a system level analysis and which on the lower implementation level.

3.1.3 Verification of the Chapter 3.8 of Subset 026 (Institut Télécom)

Contributing project partners

This work has been performed by (Institut Télécom)

Process step

This activity is part of the verification of the Elaborated System Requirements which are based on Subset 026 [3]. It contributes to the System Design Verification Report (1-12). In formalizing and analyzing the procedures its findings contribute also to the definition of the Elaborated System Requirements themselves (1-07).

Object of verification

The object of verification is the concept of the movement authority (MA) as defined in Chapter 3.8 of Subset 026 [3, Sec. 3.8] and a model derived from that description.

Available specification

The specification consists of safety properties relating to the movement authority.

Objective

This activity shall demonstrate the applicability and usefulness of the IF model checker for analyzing abstract models. As a by-product, parts of the specification in Subset 026 are checked.

Method/Approach

The control mechanism of movement authorities and the movement of the train in a simplified environment is modeled by an extended timed state machine. The safety properties express that the train stays within the bounds set by the movement authorities.

Means/Tools

The approach has been realized with the IF-Tools [6].

Results

The safety properties have been verified on the model. During modeling, three inconsistencies, ambiguities and gaps in the specification have been identified. They have been reported in [7].

Observations/Comments

The activity has been published in [8].

Conclusion

Formal modeling and formal verification can be applied to improve the informal ETCS specification.

3.4 Verification and Validation in the SW Design Phase

3.4.1 Verification of the openETCS Architecture and Design Specification

Contributing project partners

This work has been performed by the DLR.

Process step

This activity is part of the verification of the openETCS SW Architecture and Design Specification (4-19), ADD. It contributes to the SW Design Verification Report (4-23).

Object of verification

The object of verification is D3.5.3, the openETCS Architecture and Design Specification.

Available specification

The ADD is checked against Subset 026 [3].

Objective

The objective is to check that the procedures of ETCS OBU are completely, correctly and consistently mapped to the components of the SW as described in the ADD document.

Method/Approach

The verification has been performed by comparing the corresponding specifications of Subset 026 with the ADD document for each relevant paragraph.

Means/Tools

The verification has been performed manually.

Results

The verification uncovered some minor inconsistencies. These have been reported to be removed in D3.5.4 which revises D3.5.3.

Conclusion

TBD

3.4.2 Verification of the Procedure On-Sight (Systerel)

Contributing project partners

The work has been performed by Systerel.

Process step

This activity contributes to:

- the Software Requirement Specification (3.16)
- the Software Specification Verification Report (3.18)
- the Software Architecture and Design Specification (4.19)
- the SW Component Test Specification (4-22)
- the Software Design Verification Report (4.23)
- the Software Components (5.24)
- the Software Component Verification Report (5.26)

Object of verification

The object of verification is a formal model in Classical B for the procedure On-Sight. It is available at `https://github.com/openETCS/model-evaluation/tree/master/model/Classical_B_Systerel/obu_classicalB`.

Available specification

The model implements the requirements for the procedure On-Sight, as described in Subset 026 [3, Sec. 5].

Objective

The goal is to produce a B model which implement the On-Sight procedure. The B model shall be formally proved (verified) and shall allow to generate an executable C code. The main objective of the activity is to show the applicability of the B method and its implementing and supporting tools to the development task.

Method/Approach

At first, a formal model is defined with the B method using the Atelier B tool. Then the model is formally verified by proof (typing and value constraints) and model-checking (additional checks of invariants and freedom of deadlocks). Functional properties can also be defined and validated by proof or model-checking on the B model.

Finally, C code is automatically generated from the B model. To check consistency of a modified or additionally provided implementation model, tests for checking conformance to the formal model are generated

Means/Tools

The means used are:

- Atelier B to design, check, verify and prove the model (qualified by industrial railway actors)
- ProB to perform model-checking
- Atelier B translators to produce C code (Code translator shall be T3 level to obtain certified code).

Results

- The result is a fully formal model of the procedure On-Sight.
- The C code has been automatically generated.
- typing and value constraints have been formally proved for the model.
- Some properties have been checked by model checking.

See `https://github.com/openETCS/validation/blob/master/Reports/D4.3/D4.3.1-Final-VV-repor` `D4.3.1.pdf` and `https://github.com/openETCS/validation/blob/master/VnVUserStories/VnVUserStorySysterel/04-Results/b-ClassicalB-VnV/BmodelVnV.pdf`.

Conclusion

The B method, along with its verification processes and tools, meets the goals and activities of the openETCS project in terms of quality, rigor, safety and credibility.
To also satisfy the requirement of open proof, an open-source proof obligation generator is to be developed, and a framework for proving has to be built. There are but this is compensated by the fact that work on the subject is ongoing, and ProB is an effective tool for verification.

3.4.3 Model-based Test Generation for the ETCS Ceiling Speed Monitor

Contributing project partners

Main contribution by University of Bremen, additional contributors: DLR and Siemens

Process step

This activity is part of the SW Design (Phase 4). It contributes to the SW Component Test Specification (4-22).

Object of verification

The object of verification are implementations of the ETCS Ceiling Speed Monitor (CSM).

Available specification

The specification of speed and distance monitoring in [3, Sec. 3.13].

Objective

The main objective is to evaluate and demonstrate the new input equivalence class partition test generation method developed by the team of the University of Bremen. The method guarantees 100 per cent error detection inside a fault domain, and is expected to provide high coverage outside the domain. Its results on the CSM are compared with the relevant system test cases as defined in then ETCS standard conformity test specification, Subset 076.

Method/Approach

A test model specifying the expected behaviour of the CSM has been developed in SysML, using state machines and block diagrams. The model elements have been linked to the associated ETCS system requirements. Since this SysML language subset can be associated with a formal semantics, it is possible to execute algorithms that automatically generate sets of executable test cases from the model. These sets of test cases permit to check implementations for compliance with the model. The tracing information enable to derive detailed coverage and fault identification information.

The existing SUBSET-076 test cases were formalised using linear temporal logic (LTL), so that the same test data generation concept could be applied as for the test cases that were automatically identified: SUBSET-076 test cases do not provide concrete test data for every test step, but specify the general constraints from which concrete data can be elaborated. This approach also allows to trace the model coverage achieved by the SUBSET-076 test cases.

All tests were executed against software mutants derived from a reference implementation, using 3 different mutation generators in order to avoid a mutation bias. For each testing strategy applied it was checked

- which parts of the test model were covered by the test execution, and
- which fault coverage (percentage of “killed” mutants) was achieved.

Means/Tools

The whole approach is fully supported by RT-Tester and its model-based testing component RTT-MBT. Test cases are described by LTL formulas. An integrated SMT-solver generate solutions for the LTL formulas which add concrete data and makes the test cases executable. From the SysML test model, the tool automatically derives LTL formulas which describe the test cases. For the SUBSET-076 test cases, the LTL formulas have been provided manually and completed by the solver.

Results

The results can be summarised as follows.

1. The new equivalence class testing method shows significantly higher test strength than all other methods used in the comparison. It achieved nearly 100% fault coverage for mutants outside the fault domain (mutants inside the fault domain are always killed, due to the guaranteed fault detection properties).
2. The new method is very well suited for software testing and HW/SW integration testing, where the high number of test cases (approx. 5000 cases) can easily be executed, in particular, because the test suite is fully automated. The new method, however, yields too many test cases to be applied on system testing level with real trains on real tracks.
3. The SUBSET-076 test cases are missing 2 cases for the CSM in order to achieve requirements coverage. These can be easily identified and added. As a result, these test comprise 11 cases.
4. With the missing test cases added, the SUBSET-076 achieve only a fault coverage of 62% – this would certainly not suffice to obtain certification credit. It is possible, however, to add an acceptable number of test cases to the SUBSET-076 suite for the CSM which would significantly increase its test strength.

The results have been published in [9, 10, 11, 12, 13].

Observations/Comments

It is interesting to note that typical model-coverage driven test cases (e.g. transition coverage, MC/DC coverage), while achieving higher model coverage than the SUBSET-076 tests, do not achieve much higher fault coverage (approx. 68%). The reason is that these test cases are not invariant under syntactic model transformations: with another – through semantically equivalent – model, higher or lower test strength would be achieved with the coverage-driven test cases derived from that model.

In contrast to that, the new equivalence class testing strategy is elaborated from the *semantic* representation of the model and is therefore invariant (i.e. always maximal) under all syntactic model transformations that leave the behavioural semantics unchanged.

Verified Systems International GmbH who maintain the commercial version of RT-Tester have won the runner-up trophy of the EU Innovation Radar Innovation Prize[1] for implementing the equivalence class testing strategy described above in the commercial version of RT-Tester.

[1]see `https://www.verified.de/publications/papers-2015/eu-innovation-radar-price-runner-up-trophy-for-verified-systems-international/`

Conclusion

The new test strategy has shown to provide superior test strength when compared to SUBSET-076 test cases and conventional model-coverage driven test cases that are typically provided by other model-based testing tools. As of today, RT-Tester is the only testing tool where the new test strategy is implemented.

3.4.4 Model-based Testing of the ETCS Target Speed Monitor

Contributing project partners

Main contribution by University of Bremen, additional contributors: DLR and Siemens

Process step

This activity is part of the SW Design (Phase 4). It contributes to the SW Component Test Specification (4-22).

Object of verification

The object of verification is an implementation of the target speed monitoring function of the EVC, see [10]

Available specification

ETCS system specification, SUBSET-026-3; model parts are also available in [10]. The whole target speed monitoring model will be made available on `http://www.mbt-benchmarks.org`.

Objective

For creating a SysML test model of the target speed monitoring function, both time-discrete (e.g. trigger of the emergency brakes) and time-continuous (e.g. time-dependent train location, speed, and acceleration) variables need to be considered. SysML state machines are suitable for modelling concurrent real-time behaviour of time-discrete control functions. For time-continuous aspects, the report [1] describes how to use parametric constraints and associated diagrams for modelling. It is also explained how the parametric specifications are made available to the SMT solver creating concrete test data from models. As a result, the solver generates data that complies with the time-continuous physical constraints of the model.

Method/Approach

Parametric constraints represent a language aspect of the SysML which has not yet been fully investigated in the research communities. Using so-called constraint blocks, these constraints can be specified. Typically, parametric constraints represent system invariants or – this is the relevant aspect for the target speed monitor – physical laws, such as acceleration-dependent speed and speed-dependent location. For our application, these laws also comprise the ETCS braking curves modelling the speed changes of the braking train. Parametric constraints can be specified using general physical variables; these are bound to concrete model variables using parametric diagrams.

It is shown in [1] how parametric constraints can be used to calculate physically meaningful train behaviours, that is, meaningful changes of speed and location over time, taking into account the braking actions. The method follows a 2-step approach: first, a model abstraction is created, and the equivalence class testing strategy described in Section 3.4.3 is used to identify test cases with guaranteed fault detection properties. Next, the calculated tests are refined with respect to time-dependent behaviour, so that still the same equivalence classes are used, but the representatives for location and speed are selected in a way that complies with the physical laws.

Means/Tools

The method has been implemented in the RT-Tester tool as part of the WP7-related activities of the University of Bremen team.

Results

The results show that the method can be automatically performed with acceptable computation time.

Observations/Comments

To our best knowledge, this is the first SysML-based method for calculating test data with guaranteed fault detection properties in presence of both time-discrete and time-continuous observables.

Conclusion

The method developed here is highly relevant for testing cyber-physical systems in general. Verified Systems International GmbH who maintain the commercial version of RT-Tester has already decided to make this method available in 2016.

3.4.5 Test Generation Applied to the Movement Authority Mechanism (Institut Télécom)

Contributing project partners

This work has been performed by (Institut Télécom)

Process step

The techniques used in this activity would fit best the phase SW Specification, contributing to the Overall SW Test Specification (3-17), or the phase SW Design with results used in the SW Component Test Specification (4-22).

Object of verification

The object of verification is an executable which animates the handling of the movement authority (MA) by the train.

Available specification

The specification consists of a formal model of the handling of the movement authority in the ETCS system (from Sec. 3.1.3).

Objective

This activity shall demonstrate the applicability and usefulness of test generation from a formal model to check an implementation.

Method/Approach

Tests are generated which would constitute a part of a test suite covering the behavior the formal model. The tests are applied to a simulation of the system. Thereby, is conformity of the implementation behavior with that of the model is checked.

Means/Tools

The approach has been realized with TestGen-IF, the test generation facilities of the IF-Tools [6]. For demonstrating the application to an executable system simulation, such is generated by deriving JAVA simulators of the components of the model.

Results

The tests have been generated and applied to the JAVA simulation of the system. By that, the applicability of the model-based test generation approach has been demonstrated.

Conclusion

it has been demonstrated that test generation from a formal model can be applied successfully on parts of the ETCS system. This could be applied for the Overall SW Test Specification (3-17) or the SW Component Test Specification (4-22).

3.5 Verification and Validation in the SW Component Phase

3.5.1 Basic Component Verification of Components Implemented in SCADE

Each component which has been implemented with the SCADE Suite Advanced Modeler has been subjected to basic verification by the implementer. The objective of the basic verification is to establish that the component implements the functionality as required in standard (non-exceptional) usage situations. This is part of the process Phase 5, SW Component Implementation and Test. A component which has passed the basic test may be integrated with other components (Phase 6, SW Integration).

The basic test is only a part of the full test according to the SW Component Test Specification, which requires, among other, code coverage criteria to be met. That test must be performed on the final version of the component, and it is to be performed by the Tester.

As an example of the basic component verifications performed, the one of the component implementing the Speed and Distance Monitoring from [3, 3.13] is documented in the following.

3.5.2 Basic Verification of the Implementation of “Speed and Distance Monitoring”

Contributing project partners

This work has been performed by the University of Rostock.

Process step

This activity is part of Phase 5, SW Component Implementation and Test. It addresses the requirement 5.1 (basic tests performed by the Implementer).

Object of verification

The object of verification is the SCADE package `SpeedSupervision_Integration` incorporating the sub packages `CalcBrakingCurves`, `SDM_Commands`, `SDM_GradientAcceleration`, `SDM_Models`, `SDM_TargetLimits` and `TargetManagement`, together with the helper and type package `SDM_Types`.

Available specification

The specification of speed and distance monitoring is given in [3, Sec. 3.13].

Objective

The objective is to establish that the code performs the expected functionality in standard (expected) usage scenarios. It is not meant to be exhaustive. Also, it shall check for performance and memory usage abnormalities.

Method/Approach

The test has been performed by integrating the package `SpeedSupervision_UnitTest_Pkg` into the main module which provides the implementation code with inputs of a usage scenario. The test simulates a linear movement via generated odometry inputs, providing constant default national values, reasonable train data and a generic track that consists of one constant speed profile, a single, non-extending movement authority and a null-gradient. The correctness of the implementation reaction is checked mainly manually.

Means/Tools

The test has been performed with the simulation functions (SCADE Suite Simulator) of the SCADE Suite Advanced Modeler, Version 6.1. Since the basic verification is to complemented by a full verification, the tool qualification level is (only) T1.

Results

The test has been applied to each version of the implementation. Only implementation versions which passed the test had been given clearance for integration.

Conclusion

The test established readiness for integration testing. It does not cover the full verification of the code according to the Component Test Specification (4-22).

3.5.3 Code Reviews

The SW components have been subjected to code reviews to detect design errors prior to testing, and to complement testing. An example of the code reviews is presented in the ensuing section.

3.5.4 Code Review of the SCADE Package trainData

Contributing project partners

This work has been performed by the DLR.

Process step

This activity is part of the verification of the SW components. It contributes to the SW Component Verification Report (5-26).

Object of verification

The SCADE package `trainData` incorporating the sub packages `trainData_ pkg` and `trainData_Types_pkg`.

Available specification

The package was checked against the ADD document (D3.5.4) and the ETCS specification Subset-026, [3, Sec. 3.18.3].

Objective

The objective is to establish plausibility that the SCADE package conforms to the specification.

Method/Approach

The verification has been performed by comparing implementation given by the SCADE model with the corresponding specifications of Subset 026 and the ADD document.

Means/Tools

The verification has been performed using the editing capabilities of the SCADE Suite to display the model, and to search and navigate.

Results

The review uncovered some incomplete coverage of the requirements due to the absence of some implementation packages. The ramifications need to be analyzed.

3.5.5 Verification of the Modes and Levels Management Function (Systerel)

Contributing project partners

The work has been performed by Systerel

Process step

This activity contributes to:

- the Software Requirement Specification (3.16)
- the Software Specification Verification Report (3.18)
- the Software Architecture and Design Specification (4.19)
- the Software Design Verification Report (4.23)
- the Software Components (5.24)
- the Software Component Verification Report (5.26)

Object of verification

The object of verification is the Scade model for the modes and levels management function at `https://github.com/openETCS/modeling/tree/master/model/Scade/System/ObuFunctions/ManageLevelsAndModes`.

Available specification

The model implements the requirements of the modes and levels management function, as described in [3, Sec. 4,5]. Only those parts of the sections which are related to the definition of the current mode and level are covered.

Objective

The goal is to produce a SCADE model, to automatically generate executable C code from it, and to verify properties of SCADE model by model-checking.

Method/Approach

At first, a formal model is defined with the Scade suite tool. Then functional properties are formally verified on the model by model-checking.

Finally, C code is automatically translated from the Scade model.

Means/Tools

The means used are:

- SCADE suite to design, check and simulate the model

- Systerel Smart Solver (S3) to perform model-checking (can be certified as T2 tool)
- KCG translator to produce C code (Code translator shall be T3 level to obtain certified code).

Results

- The result is a Scade model of the mode and level management function integrated in the whole EVC scade model.
- The C code has been automatically generated.
- Some properties have been checked by model checking.

See `https://github.com/openETCS/validation/blob/master/Reports/D4.3/D4.3.1-Final-VV-repor` `D4.3.1.pdf` and `https://github.com/openETCS/validation/blob/master/VnVUserStories/VnVUserStorySysterel/04-Results/e-Scade_S3/Scade_S3_VnV.pdf`.

Conclusion

The following benefits of formal methods in an a posteriori verification process of critical systems have been recognized by our industrial customers.

- Contrary to a human generated test-based verification solution, a formal safety verification is intrinsically complete. It is equivalent to an exhaustive search for every possible falsification.
- It clearly identifies the complete list of assumptions upon which the safety relies.
- A certified solution allows for a reduction of the testing and review efforts (only the generic safety specification has to be reviewed).
- The use of formal verification in the qualification of critical software sends a strong and positive message to the market, and is sometimes even a requirement for some customers.

To be included: Formal code verification by Fraunhofer

3.6 Verification and Validation in the SW Integration Phase

There have been automated integration tests on the SW components.

3.7 Verification and Validation in the SW Validation Phase

There have been validations on

- the integrated software within the ¿SCADE simulation environment?, subjecting the SW with a simulated environment to operational use cases.
- an integration of the SW on a reference hardware, applying operational use cases.

bgcmmntto be included: validation by SCADE simulation

3.7.1 Validation of the Implemented and Integrated Demonstration System

Contributing project partners

The implementation and the validation of the integrated demonstration system has been performed by the DLR with support of Fraunhofer FOCUS and GE.

Process step

This activity is part of the SW Validation (Phase 7). It contributes to the Overall SW Test Report (7-29).

Object of validation

The integrated demonstration system is validated. It consists of 2 basic subsystems: the EVC and the DMI. The implementations of both subsystems are generated from the according operators of the SCADE model using the Esterel code generator KCG. The complete integrated demonstration system includes the target platforms and communication systems as well. The triggering of the EVC as well as of the DMI needs to be realised by a platform dependent wrapper. This wrapper has also to handle the communication channels and resources. The wrapper for EVC and DMI depend completely on the chosen target platform and are implemented manually. However, the basic structure and basic schemes are identical since both wrappers have the same tasks.

Available specification

The modelled SCADE simulation of the EVC and DMI running from Amsterdam to Utrecht is used as behavioural reference specification. All driver relevant outputs - such as speeds, distances, and state changes - are relevant for the validation.

Objective

Basic assumption for model driven code generation is the correctness of the code generator. This assumption is also stated for the KCG-generated model code. Hence, the functionality of the implemented EVC and the implemented DMI is not verified, since the verification is already done on model level.

Furthermore, the interaction of each subsystem on a physical hardware platform needs to be validated for correct functional behaviour. Especially platform related resource restrictions or timing issues may influence the overall behaviour of the generated implementations of the subsystems.

Method/Approach

The validation is realised by running the test track (Amsterdam - Utrecht). A concrete sequence of activities is defined in order to start-up and initiate the distributed system. External inputs, e.g. for TIU, odometry, and balise information, are provided by a simulation platform (SEFEV, proprietary software for executing Subset076 sequences) which is connected via TCP-sockets to the EVC. The behaviour of the integrated demonstration system is compared to the behaviour of the SCADE-simulation model. Tolerances for time and distances have been used as specified in Subset076.

The log files of each subsystem were used in order to check concrete behaviour.

Results

The validation was done based on Win32-implementations of each subsystem (EVC, DMI). Both subsystems were executed as single processes on the same machine. The communication was realised via TCP-sockets. The correct behaviour of the implementation compared to the simulated model was shown for a first part of the test drive of around 4km.

Observations/Comments

A second implementation of the demonstrations system was realised on an embedded realtime platform. The EVC was executed on this platform whereas the DMI needed to be executed on a Win32-platform. The generated code for EVC and DMI were identical to the code of the initial Win32-implementation. Only the platform dependent wrappers and the communication management needed to be adopted.

Due to timing and resource restrictions of the real-time platform, several synchronisation issues needed to be solved. It can be stated that the execution times of each part of the subsystem may influence the overall functional behavior.

Conclusion

The generated implementation of the SCADE model and the basic wrapping systems work as expected. Further investigations are necessary in order to validate runtime and synchronisation effects - mainly on heterogeneous target platforms.

4 Conclusion

The conclusion will be written after the completion of the V&V activities.

References

[1] Railway applications – Communication, signalling and processing systems – software for railway control and protection systems. Norm EN 50128:2011, CENELEC, Brussels, Belgium, 2011.

[2] Hardi Hungar. openETCS valdiation & verification plan. Technical Report D4.1.1.02, openETCS, July 2014.

[3] UNISIG. SUBSET-026 - System Requirements Specification. Technical Report 3.3.0, ERA, March 2012.

[4] Michael Westergaard. CPN tools 4: Multi-formalism and extensibility. In *PETRI NETS 2013*, volume 7927 of *Lecture Notes in Computer Science*, pages 400–409. Springer, 2013.

[5] Stefan Rieger. ETCS specification findings. `https://github.com/openETCS/validation/tree/master/VnVUserStories/ModelVerificationTWT/05-Work/SpecificationFindings`, January 2014.

[6] Marius Bozga, Susanne Graf, Ileana Ober, Iulian Ober, and Joseph Sifakis. The if toolset. In *Formal Methods for the Design of Real-Time Systems*, pages 237–267. Springer Berlin Heidelberg, 2004.

[7] Huu Nghia Nguyen, Joaa Santos, and Ana Cavalli. Etcs specification findings. `https://github.com/openETCS/validation/tree/master/VnVUserStories/VnVUserStoryMinesTelecom/05-Work/SpecificationFindings`, January 2014.

[8] Huu Nghiaa Nguyen and Ana R. Cavalli. Formal verification of coordination systems' requirements - a case study on the european train control system. In Halping Xu, editor, *SEKE 2014*, pages 393–396, 2014.

[9] Wen ling Huang and Jan Peleska@InCollectionraey, Title = Experimental Evaluation of a Novel Equivalence Class Partition Testing Strategy, Author = Hübner, Felix and Huang, Wen-ling and Peleska, Jan, Booktitle = Tests and Proofs, Publisher = Springer International Publishing, Year = 2015, Editor = Blanchette, Jasmin Christian and Kosmatov, Nikolai, Pages = 155-172, Series = Lecture Notes in Computer Science, Volume = 9154, Doi = 10.1007/978-3-319-21215-9_10, ISBN = 978-3-319-21214-2, Keywords = Model-based testing; Equivalence class partition testing; Adaptive random testing; SysML; State Transition Systems, Language = English, Url = http://dx.doi.org/10.1007/978-3-319-21215-9_10 . Complete model-based equivalence class testing. *International Journal on Software Tools for Technology Transfer*, pages 1–19, 2014.

[10] Felix Hübner, Christoph Hilken, and Jan Peleska. Combination of behavioral and parametric diagrams for model-based testing – application to etcs target speed monitoring. Technical report, openETCS, 2014.

[11] Cécile Braunstein, AnneE. Haxthausen, Wen-ling Huang, Felix Hübner, Jan Peleska, Uwe Schulze, and Linh Vu Hong. Complete model-based equivalence class testing for the etcs ceiling speed monitor. In Stephan Merz and Jun Pang, editors, *Formal Methods and Software Engineering*, volume 8829 of *Lecture Notes in Computer Science*, pages 380–395. Springer International Publishing, 2014.

[12] Cécile Braunstein, Jan Peleska, Uwe Schulze, Felix Hübner, Wen-ling Huang, Anne E. Haxthausen, and Linh Vu Hong. A SysML test model and test suite for the etcs ceiling speed monitor. Technical report, University of Bremen, 2014.

[13] Cécile Braunstein, Wen-ling Huang, Felix Hübner, Jan Peleska, and Uwe Schulze. Evaluation of model-based testing strategies for the etcs ceiling speed monitor. submitted to the Journal of Software Testing, Verification and Reliability, 2015.