OETCS/WP4/D4.4V0.1

openETCS

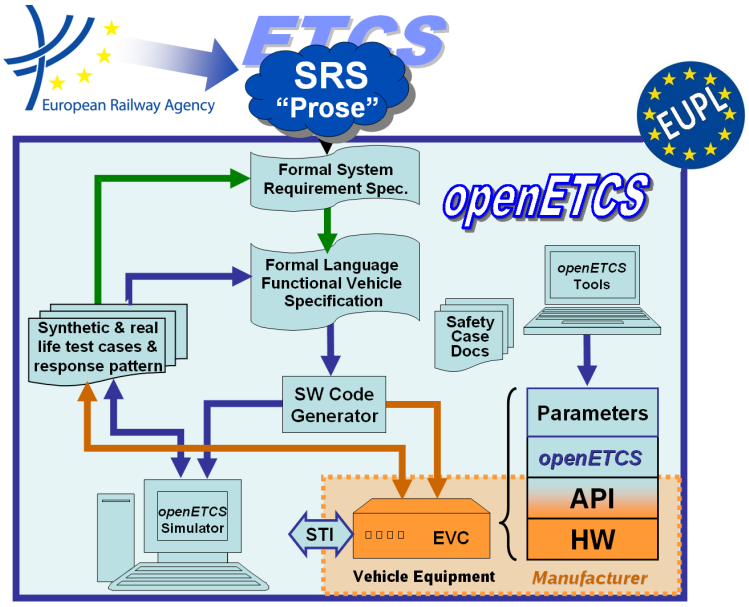
ITEA2 Project
Call 6 11025
2012 – 2015

Work-Package 4: “Verification & Validation Strategy”

openETCS Final Report on Verification and Validation

Marc Behrens and Hardi Hungar

December 2015



Funded by:

Federal Ministry of Education and Research

Région de Bruxelles-Capitale

GOBIERNO DE ESPAÑA
MINISTERIO DE INDUSTRIA, ENERGÍA Y TURISMO

This page is intentionally left blank

Work-Package 4: “Verification & Validation Strategy” **OETCS/WP4/D4.4V0.1**
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)

Marc Behrens and Hardi Hungar

DLR
Lilienthalplatz 7
38108 Brunswick, Germany

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

Table of Contents

Figures and Tables

Figures

Tables

1 Introduction

According to [**?** , 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 Verification and Validation in the Development Lifecycle

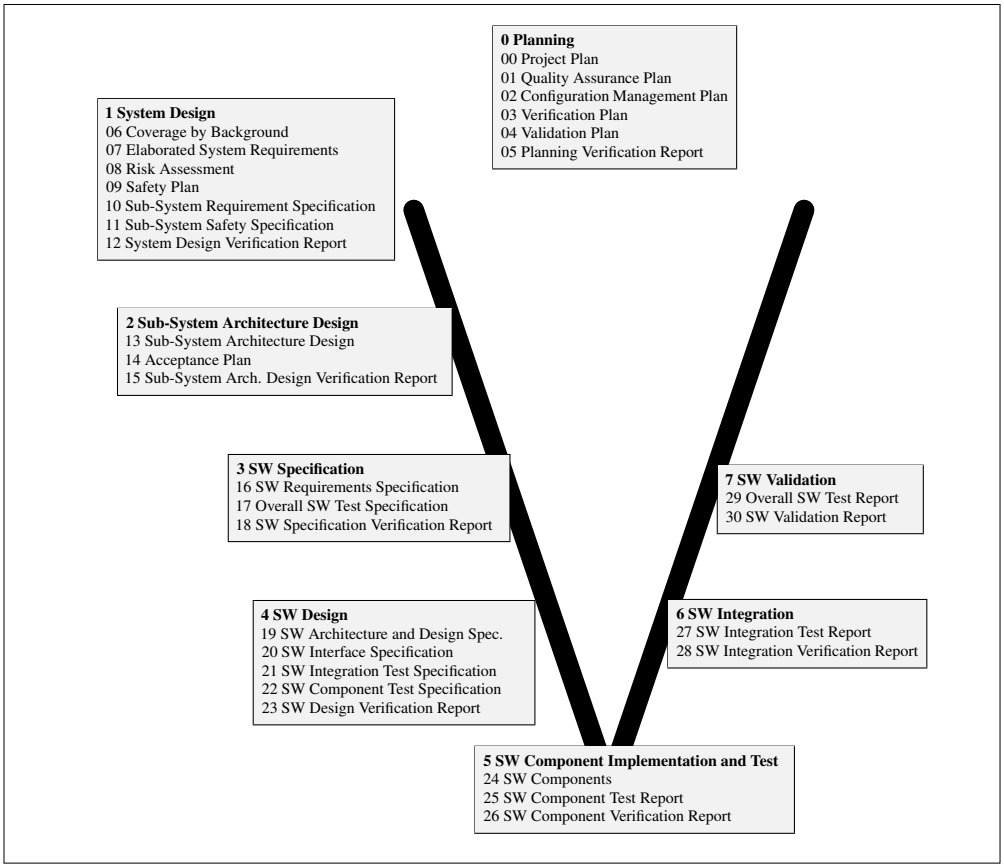


Figure 1. openETCS Development Lifecycle

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

Some sample notes are included in the subsections. To be checked for correct assignment to the phases, extended to become self-contained summaries of the activities with results and contributions. Note: Also evaluating a new verification method is a contribution to be mentioned, if this is a side or main effect of the activity. Do not forget to add yourself as an author if you contribute.

3.1 Verification and Validation in the Planning Phase

There have been reviews of the planning documents compile a list.

Template Start

3.1.1 Template Verification [Validation] of [what]

Contributing project partners

Process step

Object of verification

Available specification

Objective

Method/Approach

Means/Tools

Results

Observations/Comments

Conclusion

Template End

3.2 Verification and Validation in the System Design Phase

3.2.1 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 [**?**]. 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 [**?** , 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 [**?**]. 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 [**?**]. to be revised/completed

Observations/Comments

Conclusion

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.

3.2.2 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`.

Observations/Comments

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.

3.3 Verification and Validation in the Sub-System Architecture Design Phase

The DLR verified the Sub-System Architecture Design citations. ¿correct phase?

3.4 Verification and Validation in the SW Specification Phase

3.4.1 Verification of 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 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 Classical B model for the procedure On-Sight at `https://github.com/openETCS/model-evaluation/tree/master/model/Classical_B_Systerel/obu_classicalB`.

Available specification

The model implements the requirements for the The Procedure On-Sight, as described in *System Requirements Specification, Chapter 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.

Method/Approach

At first, a formal model is defined with the B method using the AtelierB tool. Then the model is formally verified by proof and model-checking. Functional properties can also be defined and validate by proof or model-checking on the B model.

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

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
- Atelierb 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 translated.

- The model is formally proved.
- 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.

Observations/Comments

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.
There is yet to develop open-source POG and build a framework for proving, but this is compensated by the fact that work on the subject is ongoing, and ProB is an effective tool for verification.

3.5 Verification and Validation in the SW Design Phase

Model-based testing applied to design models ¿U Bremen?

3.5.1 Verification of 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 modes and levels management function, as described in *System Requirements Specification, Chapter 4 and 5*. All the chapter are not covered, only the part related to the definition of current mode and level.

Objective

The goal is to produce a Scade model, to generate automatically executable C code, and to verify properties on 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`.

Observations/Comments

Conclusion

Benefits of formal methods in an a posteriori verification process of critical systems has been recognized by our industrial customers:

- Contrarily to a human generated test-based verification solution, a formal safety verification is intrinsically complete. It is equivalent to 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 sometime even a requirement for some customers.

3.6 Verification and Validation in the SW Component Phase

- Dedicated tests on single components ¿DB?
- Formal code verification (FRAMA C on the bitwalker)

3.7 Verification and Validation in the SW Integration Phase

Automatized integration tests on the SW components.

3.8 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.

4 Conclusion