openETCS

OETCS/WP2/D2.4.0 – 00/00

ITEA2 Project
2012 – 2015

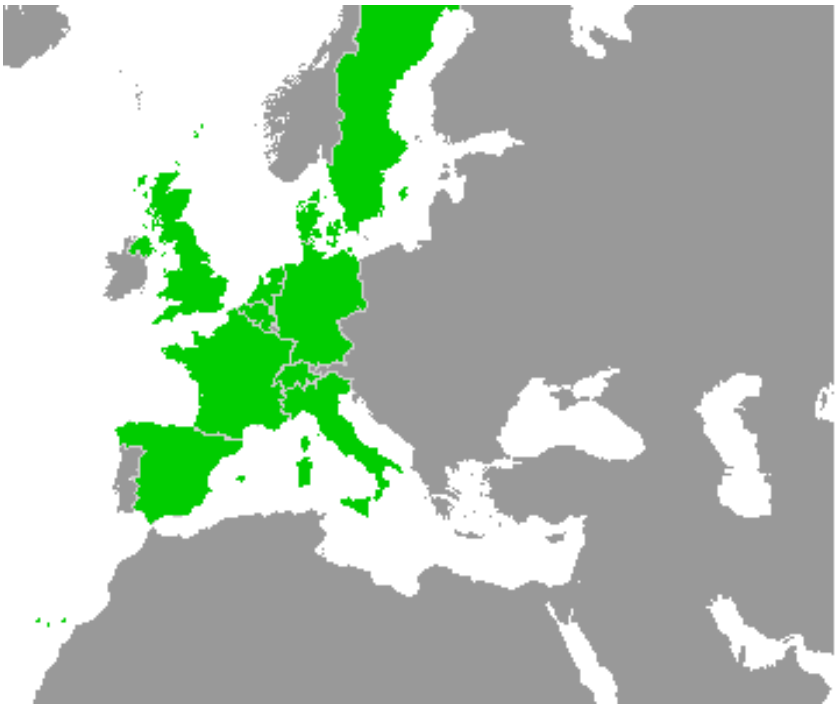
Work-Package 2: “Definition”

OpenETCS methods

Definition of the methods used to perform the formal description

Marielle Petit-Doche

February 2013



This page is intentionally left blank

OpenETCS methods

Definition of the methods used to perform the formal description

Marielle Petit-Doche

Systerel

Definition

Prepared for ITEA2 openETCS consortium
Europa

Abstract: This document give a description of the process to be applied in the OpenETCS project. It gives a description of the activities to specify and design a critical system in a first part. The second part presents an abstract description of the case study issued from subset 26.

Detailed process in regards of state of the art results and Cenelec standard requirements (detailed design plan) - For each step of the process defined in T.2.2.1 : - objectives of the step - Input/output - proposed techniques - rules on languages and tools

Table of Contents

1 Introduction

The purpose of this document is to describe, for the OpenETCS project, the activties of specification and design. However the activities of verification and validation are not in the scope of this document and will be described in WP4/D4.x.x.

These activities shall follow the requirements of EN 50126 and EN 50128 and reflect usual activities for the development of railway critical systems (see D2.1.0 and D2.2.0).

This document contents two parts :

- the description of the process to applied in the OpenETCS project
- the abstract description and the limit of the case study

2 Reference documents

- CENELEC EN 50126-1 — 01/2000 — *Railways applications — The specification and demonstration of Reliability, Availability, Maintenability and Safety (RAMS) — Part 1: Basic requirements and generic process*
- CENELEC EN 50128 — 10/2011 — *Railway applications – Communication, signalling and processing systems – Software for railway control and protection systems*
- CENELEC EN 50129 — 05/2003 — *Railway applications — Communication, signalling and processing systems — Safety related electronic systems for signalling*
- FPP — *Project Outline Full Project Proposal Annex OpenETCS* – v2.2
- SUBSET-026 3.3.0 — *System Requirement Specification*
- SUBSET-076-x 2.3.y — Test related ERTMS documentation
- SUBSET-088 2.3.0 — *ETCS Application Levels 1 & 2 - Safety Analysis*
- SUBSET-091 2.5.0 — *Safety Requirements for the Technical Interoperability of ETCS in Levels 1 & 2*
- CCS TSI — *CCS TSI for HS and CR transeuropean rail has been adopted by a Commission Decision 2012/88/EU on the 25th January 2012*

3 Conventions

The requirements are prefixed by “R-zz-x-y”, and are written in a roman typeface, where “R” stands for “Requirement”, “zz” identifies the source document,“x” is the version number and“y” is the identifier of the requirement. All the text written in italics is not a requirement: it may be a note, an open issue, an explanation of the requirements, or an example.

The placeholder “ %%xxx%% ” is used to indicates that a paragraph or section is not finished, to be defined or to be confirmed.

4 Glossary

API Application Programming Interface

FME(C)A Failure Mode Effect (and Criticity) Analysis

I/O Input/Output

OBU OnBoard Unit

QA Quality Analysis

RBC Radio Block Center

RTM RunTime Model

SIL Safety Integrity Level

THR Tolerable Hazard Rate

V&V Verification & Validation

5 Short introduction on formal approaches to design and validate critical systems

5.1 What is a formal approach ?

semi-formal, formal, syntax, semantics,...

5.2 When formal approaches are recommended according to CENELEC standard ?

5.3 Which constraints are required on the use of formal approaches ?

R-WP2/D2.3.0-X-0.1 The techniques applied to the software will be compliant regarding the SIL.

R-WP2/D2.3.0-X-0.2 The tools used shall be developed in order to be certifiable according to EN 50128.

Comment. No requirement on the way of doing this. E.g. *to have a certified (certifiable?) code generator, two generators and comparison of the result, one generation and one verification chain...*

5.4 Which are the benefits to use formal approaches ?

Comment. From D.2.5 :

The purpose of the formalization is:

- *to enhance the understanding of modelled subset;*
- *to allow formal analysis of the modelled subset;*
- *to be able to animate the model for testing an analyzing purpose;*
- *to provide information on the completeness and soundness of the SUBSET-26;*
- *to be used as a reference formal specification for the implementation of an OBU (by the OpenETCS project team and by industrial actors);*
- ...

5.5 What means are involved behind a formal approach ?

5.5.1 Model edition

5.5.2 Mathematical analyses

static checking, proof, model-checking, ...

5.5.3 Simulation and code generation

6 Formal approaches for the design and development of a system

Comment. This section will describe how formal methods are involved in the design of critical system.

6.1 System level model

6.2 Software level model

6.3 Functional architecture

6.4 Safety properties expression

R-WP2/D2.3.0-X-1 The model-level safety properties shall be written in a formal language.

7 Formal approaches for VnV of a critical system

Comment. This section will describe how formal methods are involved in the VnV of critical system in particular how they ensure the proof of safety properties.

7.1 Formal approaches for verification

Comment. completitude, consistence, static checking, type checking,...

7.2 Formal methods for validation

Comment. automatic test generation, simulation or animation, low level model validation against unit test activities,...

7.3 Formal methods for safety

Comment. proof of safety requirements, static analysis, safety analysis, traceability,...

8 Guidelines on the approaches used for OpenETCS

8.1 Definition of an OpenETCS methodology

R-WP2/D2.3.0-X-2 The model formalism shall be easily understandable by the domain experts.

R-WP2/D2.3.0-X-3 The safety properties should be provided in a declarative, simple and formal language.

R-WP2/D2.3.0-X-4 The formal model shall be understandable by or exportable to many tools (SCADE, Simulink, B tools, OpenETCS tool chain...)

R-WP2/D2.3.0-X-5 Formal specifications should be able to formalize:

R-WP2/D2.3.0-X-5.1 State machines,

R-WP2/D2.3.0-X-5.2 Time-outs,

R-WP2/D2.3.0-X-5.3 Truth tables,

R-WP2/D2.3.0-X-5.4 Arithmetics,

R-WP2/D2.3.0-X-5.5 Braking curves,

R-WP2/D2.3.0-X-5.6 Logical statements.

Comment. This requirement does not state that all these objects need to be first order objects *of the language. It only state that it should be possible (easy?) to formalize and manipulate them.*

It is to be noted that if (for example) braking curves are objects of the language, it shall be proved that they are sound, and that the code generation for these objects is also sound.

R-WP2/D2.3.0-X-6 The formal model shall be executable.

R-WP2/D2.3.0-X-6.1 The formal model shall be executable in debug mode (step-by-step), allowing inspection of states, variables and I/O.

R-WP2/D2.3.0-X-6.2 The environment shall be emulated by high level construction of the inputs.

Justification. "High level" means that it will not be necessary to define bitwise the inputs at each cycle. On the contrary, some motorization will be available to define the behavior of the inputs.

R-WP2/D2.3.0-X-7 It shall be possible to assert logical properties on the model (*i.e. invariants*).

R-WP2/D2.3.0-X-7.1 It shall be able to check the conformance of these properties at runtime.

R-WP2/D2.3.0-X-7.2 It shall be able to prove the conformance of the model to these properties.

8.2 Use of tools

R-WP2/D2.3.0-X-8 The tool chain shall be sufficiently robust to allow large software management.

R-WP2/D2.3.0-X-8.1 It shall allow modularity at any level (proof, model, software).

R-WP2/D2.3.0-X-8.2 It shall allow the management of documentation within the same tool.

R-WP2/D2.3.0-X-8.3 It shall allow distributed software development.

R-WP2/D2.3.0-X-8.4 It shall include an *issue-tracking system*, in order to allow change management and errors/bugs management.

R-WP2/D2.3.0-X-8.5 It shall allow to document/track the differences between the model and the ERTMS reference.

*Justification. In case where errors are found in the specification, or reducting choices are to be made in the model (*e.g. *in case of non-determinism).*

R-WP2/D2.3.0-X-8.6 It shall allow concurrent version development, or be compatible with tools allowing concurrent version development.

R-WP2/D2.3.0-X-8.7 In particular, it shall be made easy [1] to track the differences between two releases of a model and to manage conflicts.

R-WP2/D2.3.0-X-8.8 In particular it shall allow to track the roles and responsabilities of each participant on a configuration item, at each step of the project lifecycle.

R-WP2/D2.3.0-X-8.9 In particular, version management shall allow to track version of the safety properties together with the model.

R-WP2/D2.3.0-X-9 The tool chain shall allow traceability between the documentation (in particular the specification) and the models and safety properties.

R-WP2/D2.3.0-X-10 The tool chain shall allow traceability between the different layers of model and safety properties.

R-WP2/D2.3.0-X-11 The tools used in the tool chain shall be able to cooperate, *i.e.* the outputs of one tool will be suitable to be used as the inputs of the other tool.

R-WP2/D2.3.0-X-12 The tool chain shall conform to 50128 requirements, for the corresponding SIL and tool class.

[1] Especially in the case of a graphical language.

R-WP2/D2.3.0-X-12.1 For T2 and T3 tools[2], the choice of tools shall be justified, and the justification shall include how the tool's failures are covered, avoided or taken into account (ref. to EN 50128 6.7.4.2).

R-WP2/D2.3.0-X-12.2 All T2 and T3 tools must be provided with their user manuals.

R-WP2/D2.3.0-X-12.3 For all T3 tool, the proof of correctness or the measure taken to guarantee the correctness of the output w.r.t. their specification and the inputs shall be provided.

R-WP2/D2.3.0-X-12.3.1 . . . for data transformation,

R-WP2/D2.3.0-X-12.3.2 . . . for software transformation (*e.g.* translation, compilation. . .).

R-WP2/D2.3.0-X-13 The tool chain shall allow to write and store *test cases* and *use cases* for the model.

R-WP2/D2.3.0-X-13.1 Version management will allow to map test cases version to model versions.

R-WP2/D2.3.0-X-14 The tool chain shall allow to generate test cases for the model.

Open Issue. Is it really necessary? If we have formal proofs on the models, the tests should stay at a functional level. Therefore generated test cases should not be interesting in this context.

Open Issue. TBD requirements on the prover. Should it verify De Bruijn's criterion[3]*. Should at least the proof tree be exportable and checkable in another tool? (if the proof tree itself is mandatory).*

[2] T2: Tools contributing to the test or verification of the code or design *e.g.* static analyzers, test generators. . .)
T3: tools contributing directly or indirectly to the final code or data (*e.g.* compilers, code translator. . .)

[3] *I.e.* to be able to produce a proof tree that could be verified by a simple proof checker.