OETCS/WP2/D2.6 – 01.00.00

openETCS

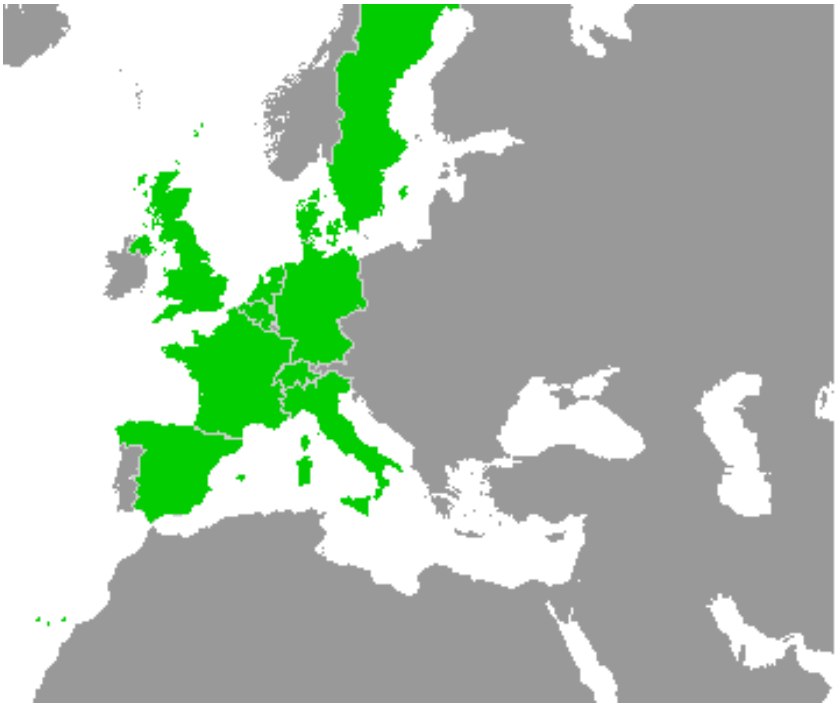
ITEA2 Project
2012 – 2015

Work-Package 2: “Requirements”

Preliminary Requirements for openETCS

Sylvain Baro

March 2013



This page is intentionally left blank

Preliminary Requirements for openETCS

Sylvain Baro
SNCF

Requirements

Prepared for ITEA2 openETCS consortium
Europa

Abstract: This document provides a preliminary view of the requirements for the openETCS project. It is meant to evolve after the initial release in order to provide the full requirement lists.

Document information	
Work Package	WP2
Deliverable ID or doc. ref.	D2.6
Document title	Preliminary Requirements for openETCS
Document version	01.00.00
Document authors (org.)	Sylvain Baro (SNCF)

Review information	
Last version reviewed	00.01.00
Main reviewers	Stéphane Callet, Cyril Cornu, David Mentre, Marielle Petit-Doche, Stanislas Pinte, Merlin Pokam, Guillaume Pottier, Uwe Steinke

Approbation			
	Name	Role	Date
Written by	Sylvain Baro		
Approved by	Gilles Dalmas	WP2 leader	

Document evolution			
Version	Date	Author(s)	Justification
00.00.00	01/01/13	S. Baro	Document creation
00.01.00	15/01/13	S. Baro	Request for comment to some partners, version for review
01.00.00	05/03/13	S. Baro	Version after first review. Req. version is 1.

Table of Contents

1 Introduction

The purpose of this document is to enumerate the meta-requirements of the projects: *i.e.* the requirements on the modeling, processes, toolchain, validation and verification. The purpose of this document is not to provide the system and safety requirements that will specify the model/software developed in the OpenETCS project.

The requirements found in the 50126 and 50128 are not rewritten here. The required plans for the project (see Sect. 7.1) shall be written according to what is required in the standards, then reviewed. Once this task is done, the plans will be the reference for the project. In the meantime, one can refer to the standards, or to the D2.2 document.

This document is initiated as a preliminary requirement list, and will evolve during the project to be completed with all the requirements on the methodology modelling, process, tool chain, and safety proof. The table herebelow sums up the area of responsibility of the contributors of WP2 on these requirements.

Subtask ID	Requirements	Subtask leader
D2.6	Set of requirements on modeling	TUBS
D2.7	Set of requirements on API	ALSTOM
D2.8	Set of requirements on tools	ERTMS Solutions
D2.9	Set of requirements on V&V	SNCF

2 Reference documents

2.1 Standards & ERA 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*
- 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*
- 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 3.2.0 — *Safety Requirements for the Technical Interoperability of ETCS in Levels 1 & 2*

2.2 Project documents

- FPP — *Project Outline Full Project Proposal Annex OpenETCS* – v2.2

- D2.1 — *Report on existing methodologies* — Jan Welte and Hansjörg Manz
- D2.2 — *Report on CENELEC standards* — Merlin Pokam and Norbert Schäfer

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 Assurance

RBC Radio Block Center

RTM RunTime Model

SIL Safety Integrity Level

SRS System Requirement Specification

SSRS SubSystem Requirement Specification

THR Tolerable Hazard Rate

V&V Verification & Validation

5 Goals

5.1 Goal 1: Formalization of the SRS in an executable formal and high level model

The first goal of the project is to propose a formalization of a subset of the on-board subsystem, as defined in the SUBSET-026, for the chosen reference baseline.

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 and analyzing purpose;
- to provide information on the completeness and soundness of the SUBSET-026;

- to be used as a reference formal specification for the implementation of an OBU (by the OpenETCS project team and by industrial actors);
- ...

In order to conduct this formalization, a part of the tool chain and methodology defined (see Goal 2) will be used.

The output of this goal is a formal specification, translatable to other tools (SCADE, Simulink, B tools, OpenETCS tool chain...) that can be given to all railway actors, and if possible associated to SRS documents in the ERA database.

The final goal is that industrial actors work with this formal specification instead of natural language specification.

5.2 Goal 2: Definition of a process/methodologies for developing on onboard software that can fulfill the EN 50128 requirements

This process/methodology must allow to produce SIL4 software. The full safety process needed for the OpenETCS to be *certifiable* according to CENELEC 50126 and 50128 shall be described in details. This safety plan will detail precisely which activities are required or not, why, and the choices that are made that allows to claim that safety is guaranteed.

Because the full design, development, validation and safety analysis process for a SIL4 OBU is a huge task far beyond the project possibilities, the full safety activities will not be conducted on the whole subsystem (see below). Nevertheless the safety process description shall be complete according to CENELEC requirements.

The process has to ensure that toolchain, formalized SUBSET-026 specification and models are certifiable according to these standards while emerging from an open source environment.

5.3 Goal 3: Definition of a tool chain for developing on onboard software that can fulfill the EN 50128 requirements

These tools must be "certifiable" according to the corresponding tool level of EN 50128 (considering the process for which they are used), but will not be certified as part of the project.

By the combination of Goal 1, 2 and 3, it should be possible for the industry to build an ETCS onboard software:

- By using OpenETCS model and proving the implementation satisfies the model;
- By using the OpenETCS toolchain with their own model;
- By using the OpenETCS model and toolchain.

5.4 Goal 4: Building an implementation of the subset of an onboard ETCS using the model and the tool chain defined in Goal 1 & 2

It is the demonstration that all the work done in the OpenETCS project is coherent, and that the tool chain is operational.

5.5 Goal 5: Define the safety properties at the model level

In order to comply the CENELEC standards, it is necessary to conduct safety activities to identify errors and anomalies in the process. One important step for this is to define safety properties which are on the same level than the formal model.

These safety properties:

- will be used for the validation of the model itself;
- will be used as reference proof obligations for the subsequent activies.

However the full safety analysis is out of the scope of the OpenETCS project. We will consider that SUBSET-088 and SUBSET-091 will provide the roots elements for safety analysis, and this elements will be refined and allocated into the modelled subsystem.

Please note that this is unsufficient in order to guarantee the safety of a system that would embed an OpenETCS OBU. The complete assertion of safety is not in the scope of the project.

5.6 Goal 6: Provide a subset of the safety case

Selected part of the safety process shall be applied (either by applying the full process on a small subset of the development, or by applying a part of the activities on the whole project development).

5.7 Goal 7: Promote OpenETCS

Promote this work to push it to become a *de facto* standard for the industrial actors, (like *e.g.* the AUTOSAR standard in the automotive world).

6 Project outline

In order to pursue these goals, the development cycle for the project may be presented as follow.

Please note that this is just an outline of the activities, not the project plan, nor the Q&A, nor the Validation plan. The verification arrows where not represented on this outline in order not to clutter the drawing. Also note that the activities needed for the toolchain are not covered here.

Fig. 1 shows the main part of the development process. This process may be seen as a “triple-V”. The smaller V corresponds to the development of the formal model.

It starts by the SRS which is not part of the project (SUBSET-026), then outlines the boundaries and the applicable requirements from the SUBSET-026 that will be used in the formal model. This outline is provided in the Subsystem Requirement Specification. This document describe the subsystem that will be modelled. It describes the architecture of the subsystem (functions and their I/O) and the requirements allocated to these functions. If necessary, the requirements are rewritten in order to address the I/O and to correspond to the allocation. This document also provide the Safety/Non Safety classification of the requirements. The architecture part is described in a formal or semi-formal language, and the requirements are described in natural language.

The next step is the creation of the formal model itself, from the SSRS. Because this model is executable, it can be validated as itself, thus the first “closing branch” of the V.

From the model can be derived some “abstract” code. The word “abstract” is used to emphasize that this code is not necessarily capable of running on a full SIL4 platform. This code can be validated in the second “closing branch”, possibly using some of the work done in the first branch.

A project demonstrator may be derived from this code (or may be the “abstract” code itself).

The third “closing branch” corresponds to the production of code capable of running on a given SIL4 platform, and the associated validation activities. This is not part of the project.

The yellow boxes correspond to activities that should be covered completely in order to produce a certifiable product, but of which only a subset will be conducted in order to demonstrate the capabilities of the product. We consider that doing the full set of activities for the project to be fully compliant to CENELEC standards (*i.e.* to be certifiable) is a huge task. Hence we should isolate a subset of this tasks, as complete in terms of tasks as possible in order to be convinced of the feasibility of the whole activities.

Therefore we have to discriminate three kinds of tasks:

blue: the tasks that will be completely achieved in the scope of the OpenETCS project;

red: the tasks that will be not be achieved, because they are out of the scope of the OpenETCS project;

yellow: the tasks for which a sample will be achieved, because they are in the scope of the OpenETCS but doing them completely is unrealistic considering the resources of the project.

Fig. 2 shows activities that are needed for the safety analyses. It should be considered in parallel of the descending branch of the V, but has been put on a separate diagram for the sake of clarity. The validation tasks corresponds to the safety validation of the model w.r.t. the safety properties. Verification tasks corresponds to the verification that the properties are each step are correct and complete w.r.t. the higher level properties. Indeed for both these tasks, arrows should really be bidirectional.

High level safety properties are provided, which must be refined side-to-side with each step on the descending branch of the V. These properties are then used for the safety analysis of the model. The validation (safety analyses) boxes are yellow because the full activity will not be conducted. Only a subset of the safety properties will be proved.

Regarding the process, WP2 shall issue (through this document) the requirements on V&V, including safety activities. From these requirements, WP4 shall propose the corresponding plans (Safety, Verification and Validation). This plans will then be reviewed by WP2 to check conformance to the requirements. The reviewed plans will then be used as reference for the activities of the WPs.

7 Requirements

7.1 Standards

R-WP2/D2.6-01-001 The project shall comply the CENELEC EN 50126, 50128 and 50129 standards.

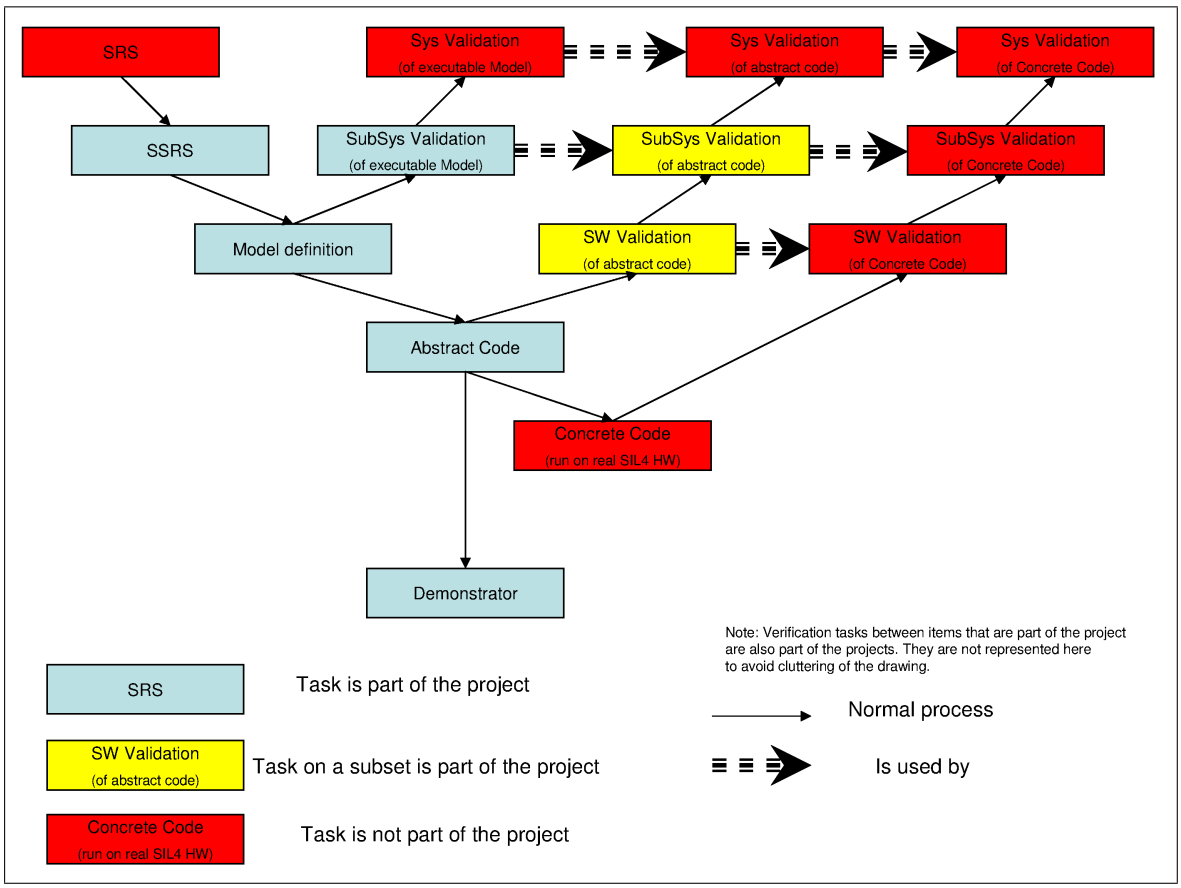


Figure 1. Main process

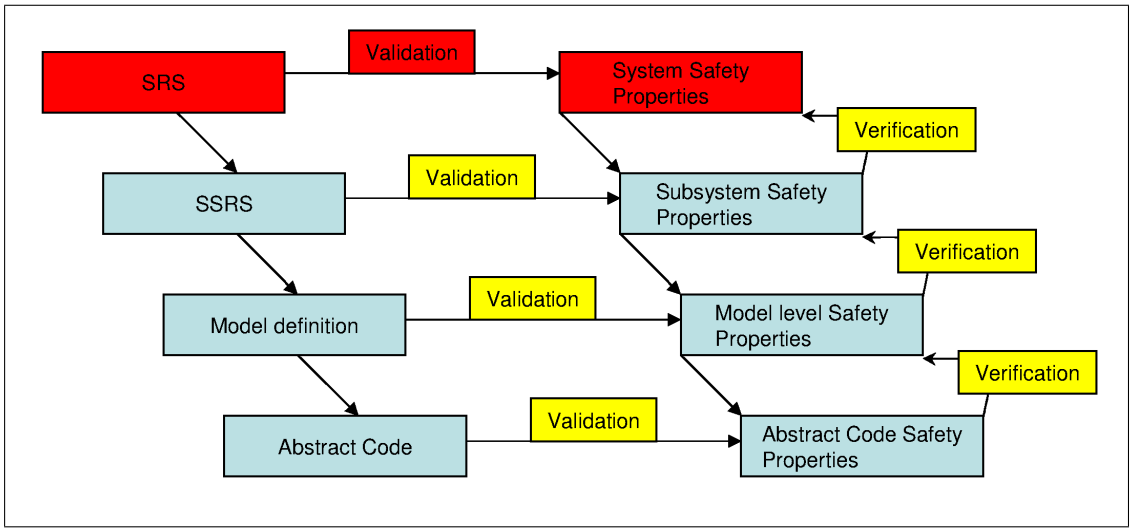


Figure 2. Safety analyses

Comment. This requirements pull some documentation issue (for example a project plan and a safety plan [1] *which describe what to do in the context of an Open Source critical software), but it also pull other requirements (compliance to ISO 9001, skill of the people in contact with critical items). These points must be considered* [2] *in the QA Plan.*

EN 50129 goes down to the hardware, but it also provides information on the system safety activities. In the scope of the project (which does not include the hardware), we will follow the EN 50129 for the components which are within the scope, and justify that some activities are not conducted because they corresponds to items outside this scope. E.g. *we do not have an implementation on a real hardware, therefore we will not require to go as down as safety analysis of the hardware itself. Hence the scope of the safety activities must be provided clearly in the safety plan, and will exclude some activities as "out of scope". Nevertheless this is not equivalent to only applying EN 50128.*

R-WP2/D2.6-01-001.01 A QA Plan shall be issued and complied with.

R-WP2/D2.6-01-001.02 The QA Plan shall describe the management of versions, OpenETCS baselines, connexion with ERTMS baselines, project life cycle, maintenance and evolution plan, organisation and roles.

R-WP2/D2.6-01-001.03 A Verification plan shall be issued and complied with.

R-WP2/D2.6-01-001.04 A Validation Plan shall be issued and complied with.

R-WP2/D2.6-01-001.05 The techniques applied to the software will be compliant regarding the SIL.

R-WP2/D2.6-01-001.06 All the output documents required by the EN 50126, EN 50128 and EN 50129 for each step of the lifecycle shall be issued, or their lack of shall be justified.

R-WP2/D2.6-01-001.07 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...*

R-WP2/D2.6-01-002 All Roles, responsabilities and more generally participators to any task (development, documentation, validation...) on any step of the project development shall be tracked, in particular to show independence of the development and testing teams. The way of tracking shall be explained in the QA Plan.

[1] Each plan is in the scope of the Work Package responsible for the corresponding tasks. *I.e.* V&V plan shall be issued by WP4.

[2] The plan has to describe either how the requirement is applied in the context of OpenETCS project, or why it is not needed.

7.2 Runtime Model & API

The framework needs to provide a list of properties and functions. If we take the parallel of the Java environment, some of these properties/functions will be provided by the *abstract machine* properties, and some of them will be provided by the API.

If we consider for example "allocation of memory", in Java usually it is just provided by the creation of an object (thus in the "Runtime model"). In C it is given by the *malloc* function, which is part of the API (of course, we will find eventually that it leads to the runtime model too, but in the user point of view, it is definitely part of the API).

What I consider is that at requirement level, we do not want to know which property will come from API and whichm will come from the runtime model. We are only interested in the properties themselves.

In order to avoid ambiguities, we will define the following.

Runtime model. This is the abstract layer required to "run" the formal model. It shall provide in the formalism part or all of the following (but not restricted to):

- memory management,
- execution of state machines (or of the chosen formal objects),
- failures,
- communication between processes and concurrence.

All these can be provided with or without safety properties. This corresponds in fact to the services provided by the "abstract machine(s)" which runs the models.

API. This is the functions/primitives required to complete the *Runtime model*. It shall provide the remaining of the features listed hereabove which are not provided by the Runtime model.

All these can be provided with or without safety properties.

RTM/API. This corresponds to the Runtime Model *plus* API. Therefore it should provide all the services needed to emulate at abstract level the hardware platform that could run the software.

Functional Architecture. This corresponds to the functional boundaries between the ETCS KERNEL and the other functional components (JRU, DMI, Odometry, Eurobalise, Eurora-dio...). These boundaries are described in the FIS or FFFIS. It also includes the parting of the KERNEL into different functions.

In the following requirements, we will not discriminate what is required from the API and from the RTM. This is the definition of these components that will allocate the requirements to the different parts. Hence we will only state requirements on the RTM/API.

7.2.1 RTM/API

This chapter has to be completed by the leader of D2.7

R-WP2/D2.6-01-003 The RTM/API model shall provide an abstraction layer of the hardware architecture.

R-WP2/D2.6-01-004 The RTM/API shall make possible to refine the software into final code able to run on hardware complying the EN 50129 standard for the requested SIL.

R-WP2/D2.6-01-005 The RTM/API shall allow discriminating Vital processing, data and I/O from Non Vital.

R-WP2/D2.6-01-006 The RTM/API shall provide a way of communication between Vital processes and Non Vital processes.

Justification. The purpose of these requirements is to be able to discriminate the safety part from the non safety part. It should be made possible to have it run on a proprietary architecture with both software on the same computer (with for example 2oo3, or coded monoprocessor) or on two different computers. One way of doing this, for example is to have some critical state machines with their data on one side, and the non critical part on the other side, with API channels to make them communicate.

R-WP2/D2.6-01-007 The RTM/API shall allow fault injection.

R-WP2/D2.6-01-008 The RTM/API shall allow logging and tracing.

R-WP2/D2.6-01-009 The RTM/API shall provide a way of reading configuration data (*e.g.* constants,. . .)

R-WP2/D2.6-01-010 The RTM/API shall provide an abstraction layer of the communication and interfaces with other components.

*Justification. Even if the FIS or FFFIS requires a specific protocol (*e.g. *Profibus), this protocol will not be implemented in the high level model. It will be considered that low level communication issues are taken into account (= emulated) by the RTM/API.*

7.2.2 Model and Architecture

This chapter has to be completed by the leaders of D2.6 and D2.7

R-WP2/D2.6-01-011 The reference ETCS baseline shall be modified only by project decision, according to the QA Plan.

R-WP2/D2.6-01-012 The SRS (SUBSET-026 for the reference baseline) shall be refined into a SSRS.

R-WP2/D2.6-01-012.01 The SSRS shall provide a functional architecture of the OBU.

R-WP2/D2.6-01-012.01.01 The SSRS shall split the KERNEL into independent functions.

R-WP2/D2.6-01-012.01.02 This architecture shall be at least semi-formal.

R-WP2/D2.6-01-012.01.03 This architecture shall provides the functions and the data streams between them.

R-WP2/D2.6-01-012.01.04 The SSRS shall describe which part of this architecture will be modelled.

R-WP2/D2.6-01-012.01.05 The SSRS shall provide the interfaces between the considered subsystem and its environment.

R-WP2/D2.6-01-012.01.06 When the boundary of the formalized subsystem corresponds to a FIS or FFFIS, the SSRS shall try to comply to it even when it is not mandatory.

R-WP2/D2.6-01-012.02 The SSRS shall allocates the requirement of the SRS to the functions and their I/O.

R-WP2/D2.6-01-012.03 Full traceability between the SRS and SSRS shall be provided.

R-WP2/D2.6-01-012.03.01 Interpretations, additions and omissions shall be tracked and justified.

R-WP2/D2.6-01-012.03.02 The requirements allocated to other subsystemes (*e.g.* RBC) shall be explicited.

R-WP2/D2.6-01-013 The model shall comply to the SSRS.

R-WP2/D2.6-01-013.01 The model shall be consistent with the SSRS level.

R-WP2/D2.6-01-013.02 Full traceability between the SSRS and the model shall be provided.

R-WP2/D2.6-01-013.02.01 Interpretations, additions and omissions shall be tracked and justified.

R-WP2/D2.6-01-014 The model design shall allow a universal method of adding functions, removing functions and overwriting functions (modularity and model extensibility).

8 Verification, Validation and Safety issues

This chapter has to be completed by the leader of D2.9

8.1 Safety

Justification. Side to side with the model (which should be a dynamic model), should lay a set of static safety properties on the model. The higher level properties will be

provided by the SUBSET-091 document, that we will consider here as a preliminary hazard analysis, because it provides us the higher level dread events.

Thise dread events will be refined by the safety analysis process into events of the same level than the model. These low level events will in turn be transformed as safety properties. The process of doing so shall be described in the Safety Plan.

This will provide Safety Properties [3] *on the model (or Dread Events). The lower level Safety Properties/Dread Events shall address variables, state and interfaces used in the formal model.*

Formal proof would then be used to prove that the OpenETCS model never enter a Dread State, as long as the other subsystem (RBC, communication layer...) fulfill their own safety properties (axiom describing the environment).

R-WP2/D2.6-01-015 A safety plan shall be provided and complied with.

R-WP2/D2.6-01-016 The subsystem shall be compatible with the THR required in the SUBSET-091.

R-WP2/D2.6-01-017 The safety analysis shall consider the Dread Events of the SUBSET-091, restricted to the scope of the subsystem.

R-WP2/D2.6-01-018 The safety properties of the SUBSET-091 shall be refined to the model level, and allocated to the functions.

R-WP2/D2.6-01-019 The Functional Architecture shall identify the Vital and Non Vital functions [4].

R-WP2/D2.6-01-020 The model-level safety properties shall be written in a formal language.

8.2 Verification and Validation

Comment. The requirements in Sect. 7.1 requires the CENELEC EN 50126, 50128 and 50129 to be followed. This pulls a number of requirements on V&V, including Verification and Validation plans. On the topic of compliance to EN 50128, one can also refer to the D2.2 document.

R-WP2/D2.6-01-021 The test plan shall comply the mandatory documents of the SUBSET-076, restricted to the scope of the OpenETCS project.

Open Issue. Before stating the V&V requirements, decisions should be taken regarding the safety issues raised in the previous section. The V&V process will be heavily impacted by the choice to do safety validation or not. It will also be impacted by the tools and methodology chosen (formal proof or not? test generation or not?).

[3] The document "Safety properties for OpenETCS through two examples" provides a proposal on this subject. `https://github.com/openETCS/requirements/tree/master/WorkDocuments/SafetyRequirementsExamples`

[4] Possibly in the SSRS.

*If code is generated with refinement proof obligations, their will not be "verification testing", but only "validation testing" (*i.e. *functional tests).*

*If we use a formal method with automatic code generation, there is no need of unit testing. There would be only the need of validation (functional) tests (*e.g. *Subset 076), and integration tests.*

It is therefore not possible to dive more into details on the requirements on V&V. This part will be updated after the choices on methodology and tools are made.

9 Language and formalism

This chapter has to be completed by the leader of D2.6

Some of the requirements in this section could suit in the Tool Chain section, but for the sake of clarity I preferred to keep them near other language requirements.

R-WP2/D2.6-01-022 The model formalism shall be easily understandable by the domain experts.

R-WP2/D2.6-01-023 The safety properties should be provided in a declarative, simple and formal language.

R-WP2/D2.6-01-024 The formal model shall be translatable to other tools (SCADE, Simulink, B tools, OpenETCS tool chain. . .)

R-WP2/D2.6-01-025 Formal specifications should be able to formalize:

R-WP2/D2.6-01-025.01 State machines,

R-WP2/D2.6-01-025.02 Time-outs,

R-WP2/D2.6-01-025.03 Truth tables,

R-WP2/D2.6-01-025.04 Arithmetics,

R-WP2/D2.6-01-025.05 Braking curves,

R-WP2/D2.6-01-025.06 Logical statements

R-WP2/D2.6-01-025.07 Messages and fields.

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/D2.6-01-026 The formal model shall be executable.

Comment. This requirement is in the section "language and formalism" because in order for the model to be executable, it has to be able to yield some algorithmic content and determinism (or a way of determining a non-deterministic model), which is indeed a property on the formal aspects of the model. In the other hand, it is also a requirement on the tool chain, hence there are also some requirements on this topic in Sect. 10.

R-WP2/D2.6-01-027 It shall be possible to assert logical properties on the model (*i.e. invariants*).

R-WP2/D2.6-01-027.01 It shall be possible to check the conformance of these properties at runtime.

R-WP2/D2.6-01-027.02 It shall be possible to prove the conformance of the model to these properties.

R-WP2/D2.6-01-028 The language and formalism should be evolutive.

10 Tool chain

This chapter has to be completed by the leader of D2.7

10.1 Usage

R-WP2/D2.6-01-029 The tool chain shall be composed only of Open Source components licensed under a license compatible with the EUPL license.

R-WP2/D2.6-01-029.01 Closed source components may be used, but only if their use is not mandatory in the process, or if an open source counterpart is provided.

R-WP2/D2.6-01-030 The tool chain shall be portable to main operating systems.

R-WP2/D2.6-01-031 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/D2.6-01-032 If tools are required to handle the configuration, they will be considered as part of the tool chain.

R-WP2/D2.6-01-033 The tool chain shall allow to generate executable code from the model.

10.2 Information management

R-WP2/D2.6-01-034 The tool chain shall be sufficiently robust to allow large software management (at least covering the onboard part of the SUBSET-026).

R-WP2/D2.6-01-034.01 It shall allow modularity at any level (proof, model, software).

R-WP2/D2.6-01-034.02 It shall allow the management of documentation within the same tool.

R-WP2/D2.6-01-034.03 It shall allow distributed software development.

R-WP2/D2.6-01-034.04 It shall include an *issue-tracking system*, in order to allow change management and errors/bugs management.

R-WP2/D2.6-01-034.05 It shall allow to document/track the differences between the model and the ERTMS reference.

R-WP2/D2.6-01-034.06 It shall support management of subsequent Subset-026 versions, as well as differences tracking between Subset-026 versions.

R-WP2/D2.6-01-034.07 It shall allow concurrent version development, or be compatible with tools allowing concurrent version development.

R-WP2/D2.6-01-034.08 The version management tools shall use model-based version control instead of text-based version control, when appropriate.

R-WP2/D2.6-01-034.09 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/D2.6-01-034.10 In particular, version management shall allow to track version of the safety properties together with the model.

R-WP2/D2.6-01-035 The tool chain shall allow traceability between:

R-WP2/D2.6-01-035.01 the documentation and the model,

R-WP2/D2.6-01-035.02 the documentation and the tests,

R-WP2/D2.6-01-035.03 the model and the tests,

R-WP2/D2.6-01-035.04 the documentation and the model,

R-WP2/D2.6-01-035.05 the documentation and the safety properties,

R-WP2/D2.6-01-035.06 the model and the safety properties,

R-WP2/D2.6-01-035.07 the tests and the safety properties.

10.3 Testing

R-WP2/D2.6-01-036 The formal model shall be executable in debug mode (step-by-step), allowing inspection of states, variables and I/O.

R-WP2/D2.6-01-037 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 automation will be available to define the behavior of the inputs.

R-WP2/D2.6-01-038 The tool chain shall allow to write, execute and store *test cases* and *use cases* for the model.

R-WP2/D2.6-01-039 Version management will allow to map test cases version to model versions.

R-WP2/D2.6-01-040 The tool chain shall allow to generate test cases for the model from a test model.

10.4 Conformance to standards

R-WP2/D2.6-01-041 Each tool in the tool chain shall be classified among T1, T2 and T3.

R-WP2/D2.6-01-042 The tool chain shall conform to EN 50128 requirements, for the corresponding SIL and tool class [5].

R-WP2/D2.6-01-042.01 For T2 and T3 tools [6], 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/D2.6-01-042.02 All T2 and T3 tools must be provided with their user manuals.

R-WP2/D2.6-01-042.03 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/D2.6-01-042.03.01 ... for data transformation,

R-WP2/D2.6-01-042.03.02 ... for software transformation (*e.g.* translation, compilation...).

[5] Refer in particular to D2.2.

[6] 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...)

