OETCS/WP2/D2.6 – 1.1.0

openETCS

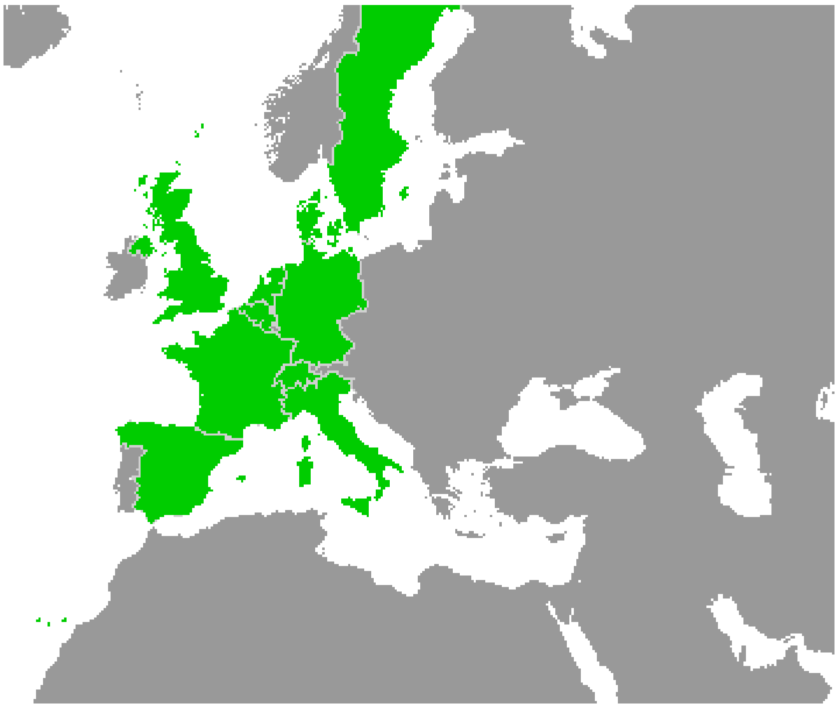
ITEA2 Project
2012 – 2015

Work-Package 2: “Requirements”

Requirements for openETCS

Sylvain Baro and Jan Welte

April 2013



Requirements for openETCS

Sylvain Baro
SNCF

Jan Welte
TU-BS

Requirements

Prepared for ITEA2 openETCS consortium
Europa

Abstract: This document provides the list of higher level requirements for the openETCS project.

Document information	
Work Package	WP2
Deliverable ID or doc. ref.	D2.6
Document title	Requirements for openETCS
Document version	1.1.0
Document authors (org.)	Sylvain Baro (SNCF)

Review information	
Last version reviewed	0.1.0
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
0.0.0	01/01/13	S. Baro	Document creation
0.1.0	15/01/13	S. Baro	Request for comment to some partners, version for review
1.0.0	05/03/13	S. Baro	Version after first review. Req. version is 1.
1.1.0	19/04/13	S. Baro, J. Welte	Update after workshop on safety and tools. Updated req. version is X.

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 modelling, 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 CENELEC standards as well as those found in the ERTMS specifications are not rewritten here. The required plans for the project shall be written according to what is required in the standards (see Sect. 2.1), 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 was initiated as a preliminary requirement list, and is now evolving during the project to be completed with all the requirements on the methodology modelling, process, tool chain, and safety proof.

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
FFM	Fully-Formal Model
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
SFM	Semi-Formal Model
SRS	System Requirement Specification (in this document, this is equivalent to the SUBSET-026 of the TSI)
SSRS	SubSystem Requirement Specification
THR	Tolerable Hazard Rate
V&V	Verification & Validation

5 Goals

The following goals for the projects were considered for the writing of this document. Please note that they are *not* listed by order of priority, but in an arbitrary order.

5.1 Goal 1: Formalization of the SRS into a high level, executable, semi-formal and formal 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.

There will be two models created to satisfy this goal. One semi-formal executable (but not in real time) model, for which we aim to model all the requirements considered in the scope by the system design, covering as much features as possible of the Onboard Unit.

On some parts of the system, this model will be completed with a strictly formal model. This will allow to improve the understanding of the system, and will provide elements for verification and validation using formal proof.

This will also allow to demonstrate the feasibility of the process using both semi-formal and formal modelization and the capabilities of analysis on the formal model using formal methods.

5.2 Goal 2: Definition of process/methodologies required for the safety validation of the model

The full safety process needed for the OpenETCS to be *certifiable* according to the CENELEC standards shall be described. The safety plan and safety case concept will detail precisely which activities are required or not with regard to the scope and the context of the project.

The process has in particular 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: Apply the safety activities on a subset of the project

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.

The safety analyses as described in the safety plan and safety case concept shall be conducted on a selected subset of the system, and on a selected subset of safety properties. It will start with system-level safety activities, then go down into the semi-formal, then formal model.

5.4 Goal 4: Definition of a tool chain for developing on onboard software that can fulfill the EN 50128 requirements

These tools must be *qualified* 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[1] and proving the implementation satisfies the model;
- By using the OpenETCS toolchain with their own model;
- By using the OpenETCS model and toolchain.

5.5 Goal 5: Building a non Vital executable demonstrator/simulator of the model

From the model, an executable software will be derived. This software will be used for simulation and demonstration. This software will be completely non Vital. It will be able to execute in real time and to interface with other components.

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 "double-V". The smaller V corresponds to the development and validation of the high level 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 model. This outline is provided in the Subsystem Requirement Specification (SSRS). 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 Vital/Non Vital classification of the requirements and data stream. The architecture part is described in a semi-formal language, and the requirements are described in natural language.

The next step is the creation of the model itself, from the SSRS. Because this model is executable, it can be validated as itself, thus the first "closing branch" of the V. This model will be semi-formal, and aim to cover the full SSRS requirements. This semi-formal model will be sidekicked with a formal model for some of the functions of the model.

The first "closing branch" corresponds to the validation of the subsystem, using the semi-formal model (with tests) and the formal model (with proof).

The second "closing branch" corresponds to the production of real time code capable of running on an onboard computer. This code shall be derived from the semi-formal model (possibly with an intermediate SW model between). There are two possibilities for the generation of code:

- a SIL0 *demonstrator* or *simulator*,
- a SIL4 implementation of the model.

[1]Or *models* if several models are developed. In order to enhance readability, I will use the singular even if it could cover a semi-formal model and a formal model.

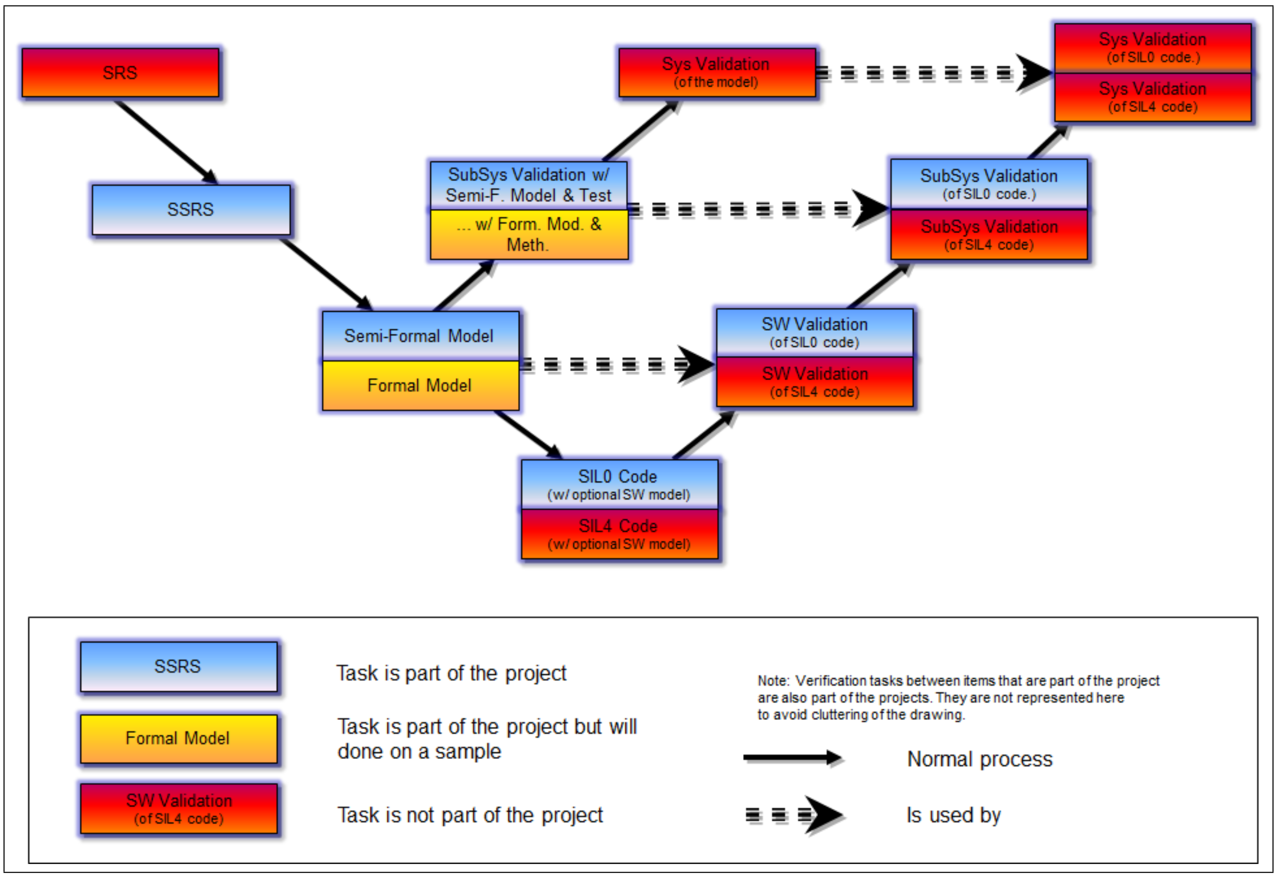

SRS
SSRS
Sys Validation (of the model)
Sys Validation (of SIL0 code.)
Sys Validation (of SIL4 code)
SubSys Validation w/ Semi-F. Model & Test
... w/ Form. Mod. & Meth.
SubSys Validation (of SIL0 code.)
SubSys Validation (of SIL4 code)
Semi-Formal Model
Formal Model
SW Validation (of SIL0 code)
SW Validation (of SIL4 code)
SIL0 Code (w/ optional SW model)
SIL4 Code (w/ optional SW model)
SSRS
Task is part of the project
Formal Model
Task is part of the project but will done on a sample
SW Validation (of SIL4 code)
Task is not part of the project
Note: Verification tasks between items that are part of the project are also part of the projects. They are not represented here to avoid cluttering of the drawing.
Normal process
Is used by

Figure 1. Main process

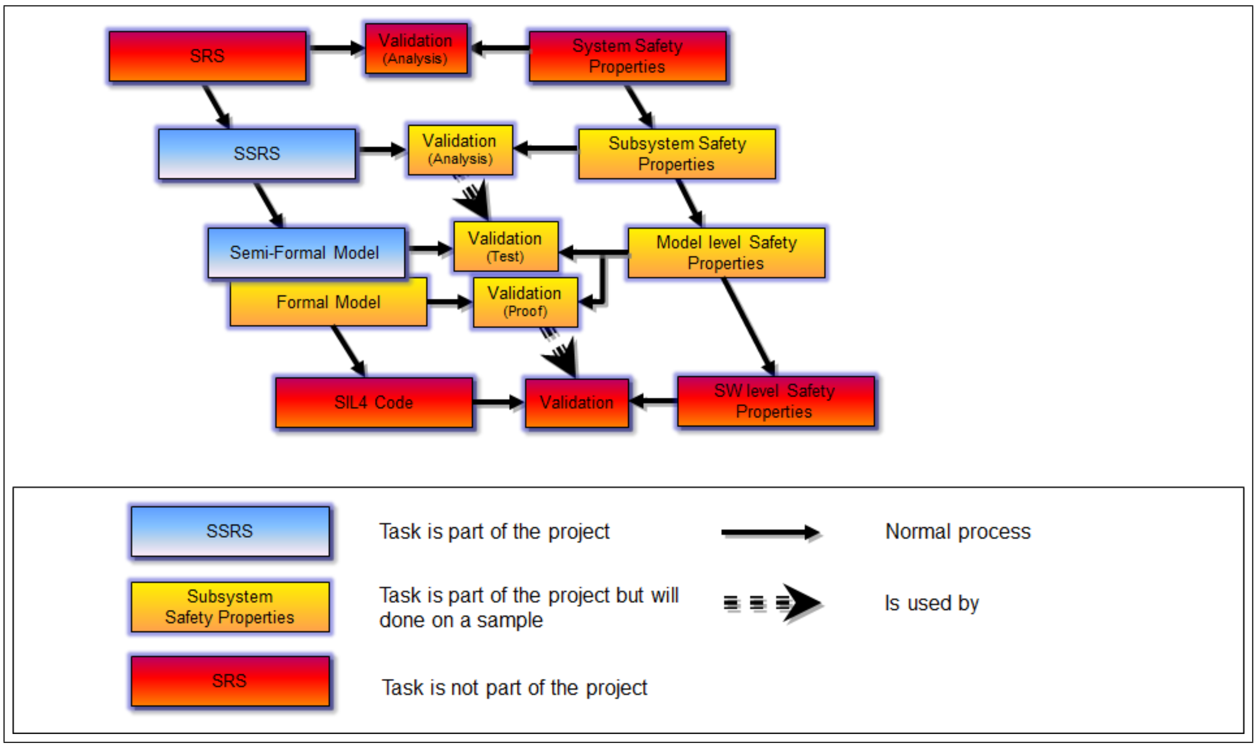

SRS
Validation (Analysis)
System Safety Properties
SSRS
Validation (Analysis)
Subsystem Safety Properties
Semi-Formal Model
Validation (Test)
Model level Safety Properties
Formal Model
Validation (Proof)
SIL4 Code
Validation
SW level Safety Properties
SSRS
Task is part of the project
Subsystem Safety Properties
Task is part of the project but will done on a sample
SRS
Task is not part of the project
Normal process
Is used by

Figure 2. Safety analyses

The SIL0 demonstrator is a deliverable of the project openETCS. On the other hand, the project toolchain and methodology shall be compatible with the generation of a SIL4 software, but the generation of this SW and the generator used for this generation are *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. These activities will be conducted on a subset of the model. The validation tasks corresponds to the safety validation of the model w.r.t. the safety properties.

A subset of 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 subset.

7 Requirements

7.1 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 which 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,
- *real time* clock.

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, Euroradio. . .). 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.

R-WP2/D2.6-X-1 The RTM/API model shall provide an abstraction layer of the hardware architecture.

R-WP2/D2.6-X-1.1 The RTM/API shall abstract memory management.

R-WP2/D2.6-X-1.2 The RTM/API shall abstact the execution of states machine.

R-WP2/D2.6-X-1.3 The RTM/API shall allow communication and concurrence (if requested by the model formalism).

R-WP2/D2.6-X-1.4 The RTM/API shall allow read/write to a persistent data store.

R-WP2/D2.6-X-1.5 The RTM/API shall allow to state *real time* constraints.

R-WP2/D2.6-X-1.6 The RTM/API shall provide emulation for a *real time* clock.

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 processing, data and I/O.

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 System and Architecture

R-WP2/D2.6-X-9 The reference ETCS baseline shall be modified only by project decision.

R-WP2/D2.6-X-10 The SRS (SUBSET-026 for the reference baseline) shall be refined into a SSRS.

R-WP2/D2.6-X-10.1 The SSRS shall identify all functions, interfaces and parameters in addition to the ones covered by the SRS but which are requested at functional level in order to be compliant to the SRS.

R-WP2/D2.6-X-10.2 The SSRS shall provide a functional architecture of the OBU.

R-WP2/D2.6-X-10.2.1 The SSRS shall split the KERNEL into independent functions.

R-WP2/D2.6-X-10.2.2 The description of the architecture shall be semi-formal.

R-WP2/D2.6-X-10.2.3 This architecture shall provide the functions and the data streams between them.

R-WP2/D2.6-X-10.2.4 The SSRS shall describe which part of this architecture will be modelled.

R-WP2/D2.6-X-10.2.5 The SSRS shall provide the interfaces between the considered subsystem and its environment.

R-WP2/D2.6-X-10.2.6 The SSRS shall identify all functions, interfaces and parameters in addition to the ones covered by the SRS but which are requested at functional level in order to be compliant to the SRS.

R-WP2/D2.6-X-10.2.7 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-X-10.3 The SSRS shall allocate the requirements of the SRS to the functions and their I/O.

R-WP2/D2.6-X-10.4 The SSRS shall be compliant or shall allow compliance to the mandatory requirements from the other subsets of the TSI.

R-WP2/D2.6-X-10.5 Full traceability between the SRS and SSRS shall be provided.

R-WP2/D2.6-X-10.5.1 All interpretations, additions, omissions and design choices during the allocation have to be documented in one log and justified.

R-WP2/D2.6-X-10.5.2 The requirements allocated to other subsystemes (*e.g.* RBC) shall be tracked.

R-WP2/D2.6-X-10.5.3 Lower level requirements refined from higher level requirements shall be tracked.

R-WP2/D2.6-X-10.6 In case of divergence between an identified national behavior and the SRS behavior, it shall be discussed and decided whether or not the national behaviors are required in the SSRS.

R-WP2/D2.6-X-10.6.1 If it is the case, it shall be traced explicitly.

R-WP2/D2.6-X-10.6.2 If it is the case, it shall be possible by configuration to enforce pure compliance to the SRS.

R-WP2/D2.6-X-11 The SSRS shall identify the Vital and Non Vital functions, requirements, and data streams.

7.3 Model(s)

R-WP2/D2.6-X-12 The subsystem described in the SSRS shall be modelled into a Semi-Formal Model (SFM).

R-WP2/D2.6-X-12.1 The SFM shall be modelled in a semi-formal means of decription.

R-WP2/D2.6-X-12.2 The SFM shall be as consistent as possible with the SSRS level of abstraction.

R-WP2/D2.6-X-12.2.1 All the requirements of the SSRS shall be covered by the SFM.

R-WP2/D2.6-X-12.2.2 All safety requirements derived from the Hazard Log and allocated to the sub-system functionality shall be covered by the SFM

R-WP2/D2.6-X-12.2.3 Needed interpretations, additions and omissions shall be tracked and justified.

R-WP2/D2.6-X-12.2.4 If a deviation is needed, it shall be tracked and justified, and it will also be insulated as much as possible using proper modularity.

R-WP2/D2.6-X-12.2.5 Traceability between the SSRS and the SFM shall be provided.

R-WP2/D2.6-X-12.2.6 Exported requirements shall be tracked.

R-WP2/D2.6-X-12.2.7 Lower level functional requirements refined from higher level functional requirements shall be tracked.

R-WP2/D2.6-X-12.2.8 Lower level safety requirements refined from higher level safety requirements shall be tracked.

R-WP2/D2.6-X-13 The SFM design and level of description shall be executable.

R-WP2/D2.6-X-14 Some parts of the subsystem described in the SSRS shall be modelled into a Fully Formal Model (FFM).

R-WP2/D2.6-X-14.1 The FFM shall be modelled in a strictly formal mean of description[2].

R-WP2/D2.6-X-14.2 The FFM shall be modelled based on the SFM.

[2]Nevertheless, we will use the wording "Fully Formal Model" in order to be allowed to use a different abbreviation than for "Semi-Formal Model".

R-WP2/D2.6-X-14.3 The SFM shall be designed in order to make possible the modelling and the validation of the FFM.

R-WP2/D2.6-X-14.4 The transformation of the SFM into the FFM shall be as straightforward and automated as possible[3].

R-WP2/D2.6-X-15 The SFM design shall be modular and extensible.

R-WP2/D2.6-X-16 The FFM design shall be modular and extensible.

R-WP2/D2.6-X-16.1 The modular design of the FFM should represent a refinement of the modular SFM design.

R-WP2/D2.6-X-17 Safety relevant function should be as much as possible insulated from Non Safety relevant functions.

7.4 Safety

Side to side with the models (SFM and FFM), should lay a set of static safety properties on the model. The higher level properties will be provided by the SUBSET-088 and SUBSET-091 documents, that we will consider here as a preliminary hazard analysis, because it provides us the higher level dread events.

These dread events will be refined during a Sub System Hazard Analysis (SSHA) into events of the abstraction level of the sub system, which will be collected in the Hazard Log. For all hazardous events named in the Hazard Log a risk analysis will be performed. Based on the analyses suitable risk control activities a determined and collected in the Hazard Log. All hazards and their control activities are then refined into safety requirements/ properties[4] which are transferred to the respective backlogs for the different models and their verification and validation. 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). The exact process shall be described in the Safety Plan.

R-WP2/D2.6-X-18 A Safety Case Concept and Safety Plan shall be issued describing the safety activities for the model according to CENELEC EN 50126, EN 50128 and EN 50129.

R-WP2/D2.6-X-18.1 All the output documents required by the EN 50126, EN 50128 and EN 50129 for each step of the lifecycle shall be described, or their lack of shall be justified.

[3]Please note that the requirements on the FFM do not forbid a complete automation of the transformation, nor the identity of the FFM and SFM.

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

R-WP2/D2.6-X-18.2 The safety activities described in this documents shall cover all the steps from the SRS to the SSRS, then to the SFM, the FFM and the source code.

R-WP2/D2.6-X-19 The safety activities shall at least be applied on a sample of the on-board functionalities and the respective safety requirements to demonstrated the suitability of the safety plan.

R-WP2/D2.6-X-20 The safety activities shall aim at a THR compatible with the SUBSET-091.

R-WP2/D2.6-X-21 The safety analysis shall consider as higher level Dread Events the events named in the SUBSET-091 in respect to the scope of the subsystem.

R-WP2/D2.6-X-21.1 Those Dread Events shall be refined to SRS and SSRS levels, then to the model level, and allocated to the functions to determine hazardous events.

R-WP2/D2.6-X-21.2 The refinement shall only be done for a sample part of the system.

R-WP2/D2.6-X-22 The model-level safety requirements shall be written in the same degree of formality as the corresponding model or higher.

R-WP2/D2.6-X-23 It shall be verified that the SFM, FFM and source code complies with all respective safety requirements:

R-WP2/D2.6-X-23.1 either by a real-time constraint [5];

R-WP2/D2.6-X-23.2 using test for the SFM;

R-WP2/D2.6-X-23.3 using formal proof for the part modelled in the FFM and for the sorce code;

R-WP2/D2.6-X-23.4 if a safety requirement/property can not be proven, testing covering all reasonable possible events shall be used.

7.5 Verification and Validation

The already provided requirements requires a safety plan compliant to the CENELEC EN 50126, 50128 and 50129. This pulls a number of requirements on V&V, including Verification and Validation plans. On the topic of compliance to EN 50128, one shall also refer to the D2.2 document.

R-WP2/D2.6-X-24 A Verification plan shall be issued and complied with.

[5]This real-time constraint is not guaranteed in the model itself, but can be expressed in the model using the RTM/API.

R-WP2/D2.6-X-24.1 The verification plan shall provide a method to demonstrate the requirement covering over all development artifacts.

R-WP2/D2.6-X-24.2 The verification plan shall state all verification activities from the SUBSET 26 and all other relevant TSI documents to the source code.

R-WP2/D2.6-X-25 A Validation Plan shall be issued and complied with.

R-WP2/D2.6-X-25.1 The validation plan shall provide a method to validate all functional and safety requirement over all development artifacts.

R-WP2/D2.6-X-25.2 The validation plan shall state all validation activities from the SUBSET 26 and all other relevant TSI documents to the source code.

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.

Justification. It will possibly be difficult to model all the tests in the course of the project, but the test plan should at least be complete.

7.6 Language and formalism

R-WP2/D2.6-X-27 The description of the SFM shall be understandable by domain experts without specific training.

R-WP2/D2.6-X-27.1 The SFM shall provide graphical descriptions for components relations, process sequences and data flows.

R-WP2/D2.6-X-27.2 Process involving timing and scheduling aspects shall be visually demonstrated.

R-WP2/D2.6-X-28 The means of discription used to formalize safety properties shall be formal and declarative.

R-WP2/D2.6-X-28.1 The formalism shall be understandable by domain experts without specific training.

R-WP2/D2.6-X-28.2 The formalism shall be able to state:

R-WP2/D2.6-X-28.2.1 constraints,

R-WP2/D2.6-X-28.2.2 logical relations,

R-WP2/D2.6-X-28.2.3 timing relations.

R-WP2/D2.6-X-29 All means of description used shall be standardized or at least documented in detailed.

R-WP2/D2.6-X-30 The means of descriptions used for SSRS, SFM and FFM modeling shall be open for the use with different tool *via*:

R-WP2/D2.6-X-30.1 direct exchange of the model data file;

R-WP2/D2.6-X-30.2 or fully automated translation of the model data file.

R-WP2/D2.6-X-31 The means of description for the SFM shall be able to formalize:

R-WP2/D2.6-X-31.1 State machines,

R-WP2/D2.6-X-31.2 Time-outs,

R-WP2/D2.6-X-31.3 Truth tables,

R-WP2/D2.6-X-31.4 Arithmetics,

R-WP2/D2.6-X-31.5 Braking curves,

R-WP2/D2.6-X-31.6 Logical statements,

R-WP2/D2.6-X-31.7 Messages and fields.

R-WP2/D2.6-X-32 The language for the FFM shall be able to formalize as much as possible of:

R-WP2/D2.6-X-32.1 State machines,

R-WP2/D2.6-X-32.2 Time-outs,

R-WP2/D2.6-X-32.3 Truth tables,

R-WP2/D2.6-X-32.4 Arithmetics,

R-WP2/D2.6-X-32.5 Braking curves,

R-WP2/D2.6-X-32.6 Logical statements,

R-WP2/D2.6-X-32.7 Messages and fields,

R-WP2/D2.6-X-32.8 Constraints,

R-WP2/D2.6-X-32.9 Logical relations.

Comment. These requirements do not state that all these objects need to be first order objects *of the language. They only state that it should be possible 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-X-33 The mean of description of the SFM shall allow it to 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. 7.7.

R-WP2/D2.6-X-34 It shall be possible to assert logical properties on the SFM and FFM (*e.g.* invariants).

R-WP2/D2.6-X-34.1 It shall be possible to check the conformance of these properties at runtime on the SFM.

R-WP2/D2.6-X-34.2 It shall be possible to prove the conformance of the FFM to these properties.

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

7.7 Tools chain

7.7.1 Usage

R-WP2/D2.6-X-36 The tools chain shall be composed as far as possible of Open Source components licensed under a license compatible with the EUPL license.

R-WP2/D2.6-X-36.1 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-X-36.2 If a closed source component is used, it has to be displayed how an open source component has to be designed to replace the closed component later.

R-WP2/D2.6-X-37 The tools chain shall be portable to common operating systems.

R-WP2/D2.6-X-37.1 The tool chain shall run stable on all main operating systems.

R-WP2/D2.6-X-37.2 The tool chain shall run with a good performance on all main operating systems.

R-WP2/D2.6-X-38 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 another tool.

R-WP2/D2.6-X-38.1 All possible input and output formats of a tool have to be documented.

R-WP2/D2.6-X-38.2 Open data formats shall be used for the tool cooperation.

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

R-WP2/D2.6-X-40 The tools chain shall allow to generate executable code from the model(s).

7.7.2 Information management

R-WP2/D2.6-X-41 The tools chain shall be sufficiently robust to allow large software management (at least covering the onboard part of the SUBSET-026).

R-WP2/D2.6-X-41.1 It shall allow modularity at any level (proof, models, software).

R-WP2/D2.6-X-41.2 It shall allow the management of documentation.

R-WP2/D2.6-X-41.3 It shall allow distributed software development.

R-WP2/D2.6-X-41.4 It shall allow simultaneous multi user usage.

R-WP2/D2.6-X-41.5 It shall include an *issue-tracking system*, in order to allow change management and errors/bugs management.

R-WP2/D2.6-X-41.6 It shall allow to document/track the differences between the models and the ERTMS reference.

R-WP2/D2.6-X-41.7 It shall support management of subsequent Subset-026 versions, as well as differences tracking between Subset-026 versions.

R-WP2/D2.6-X-41.8 It shall allow concurrent version development, or be compatible with tools allowing concurrent version development.

R-WP2/D2.6-X-41.9 The version management tools shall use model-based version control instead of text-based version control, when appropriate.

R-WP2/D2.6-X-41.10 In particular it shall allow to track the roles and responsibilities of each participant on a configuration item, at each step of the project lifecycle.

R-WP2/D2.6-X-41.11 In particular, version management shall allow to track version of the safety properties together with the models.

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

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

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

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

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

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

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

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

R-WP2/D2.6-X-43 The tools chain shall be compliant to EN 50128 for the corresponding tool level.

7.7.3 Testing

R-WP2/D2.6-01-036 The SFM 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 SFM.

R-WP2/D2.6-X-47 Version management will allow to map test cases version to the SFM, the FFM and source code versions.

R-WP2/D2.6-X-48 The tool chain shall allow to generate test cases for the SFM, the FFM and source code from a test model.

R-WP2/D2.6-X-49 The tool chain shall allow to write, execute and store test sequences combining multiple test cases for the SFM, the FFM and source code.

7.7.4 Conformance to standards

R-WP2/D2.6-X-50 Each tool in the tool chain shall be classified among T1, T2 and T3 depending on its usage in the process.

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

R-WP2/D2.6-01-042.01 For T2 and T3 tools [7], 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.

[6] Refer in particular to D2.2.

[7] 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...)

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. . .).

7.8 Demonstrator

R-WP2/D2.6-X-52 A demonstrator software will be built from the SFM.

R-WP2/D2.6-X-53 The demonstrator shall be non vital.

R-WP2/D2.6-X-54 The demonstrator shall be able to run in real time.

R-WP2/D2.6-X-55 The demonstrator shall comply to the standardized interfaces in order to be able to interface with other subsystems.

R-WP2/D2.6-X-56 The demonstrator shall be able to run on a SIL0 onboard computer.

<End of Document>