OETCS/WP2/D2.3.0 – 00/00

openETCS

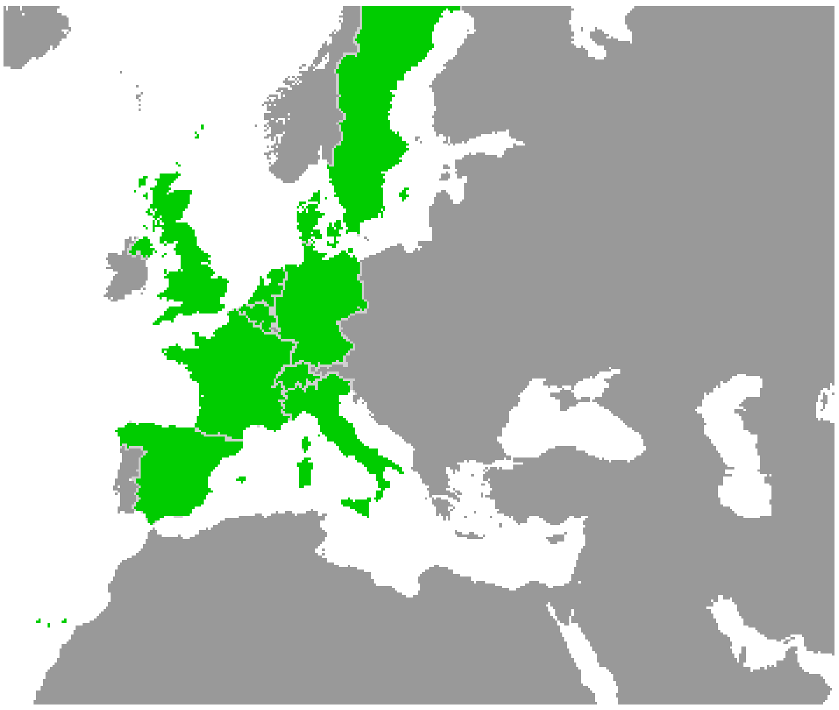
ITEA2 Project
2012 – 2015

Work-Package 2: “Requirements”

Preliminary Requirements for openETCS

Sylvain Baro and Stanislas Pinte

January 2013



This page is intentionally left blank

Preliminary Requirements for openETCS

Sylvain Baro
SNCF

Stanislas Pinte
ERTMS Solutions

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.

Table of Contents

Figures and Tables **Figures**

Tables

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. 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 standard, or to AeBT and TUBS presentations at WP2 Workshop.

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	Responsible	Major Participants	Participants
D2.3.1	Set of requirements on modeling	TUBS	TUBS; SNCF; ALSTOM; Siemens	DLR; CEDEX?; Multitel?; Mitsubishi E.; Innovalia?; SQS?
D2.3.2	Set of requirements on API	ALSTOM	SIEMENS; ALSTOM	ERSA?; SNCF; Fraunhofer;Innovalia?; SQS?
D2.3.3	Set of requirements on tools	ERTMS Solutions	SYSTEREL; CEA	%%STAN CONFIRM ACCORDING TO EFFORT- SN ERSA?; EclipseSource; ERTMS Solution; DLR; Mitsubishi E.; Innovalia?; SQS?
D2.3.4	Set of requirements on V&V	SNCF	SNCF; AEbT; ALSTOM; SIEMENS; DB; DLR	CEDEX?; Multitel?; Innovalia?; SQS?

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*

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 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-26.

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-026;
- to point and document any contradiction or inconsistency 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);
- to be transformed in a later phase to lower-level source code, for the purpose of being executed or interpreted in an EVC subsystem;

- ...

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

The output of this goal is a formal specification 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.

This formal specifications must be certifiable, but will not be certified as part of the project. %%STAN CONFIRM WITH SYLVAIN/PROJECT%%

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

These tools/methodologies must be certifiable, but will not be certified as part of the project.

By the combination of Goal 1 and 2, 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.

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.

5.3 Goal 3: 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.

5.4 Goal 4: 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.

Open Issue. Is it realistic to expect such properties on the whole model?

5.5 Goal 5: 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).

Open Issue. This point has to be discussed. The purpose here is of course to demonstrate the feasability of the safety process, but I am not convinced that it proves anything if the whole process is not applied.

5.6 Goal 6: 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. 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-26), then outlines the boundaries and the applicable requirements from the SUBSET-26 that will be used in the formal model.

The next step is the creation of the formal model itself. 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 inside 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.

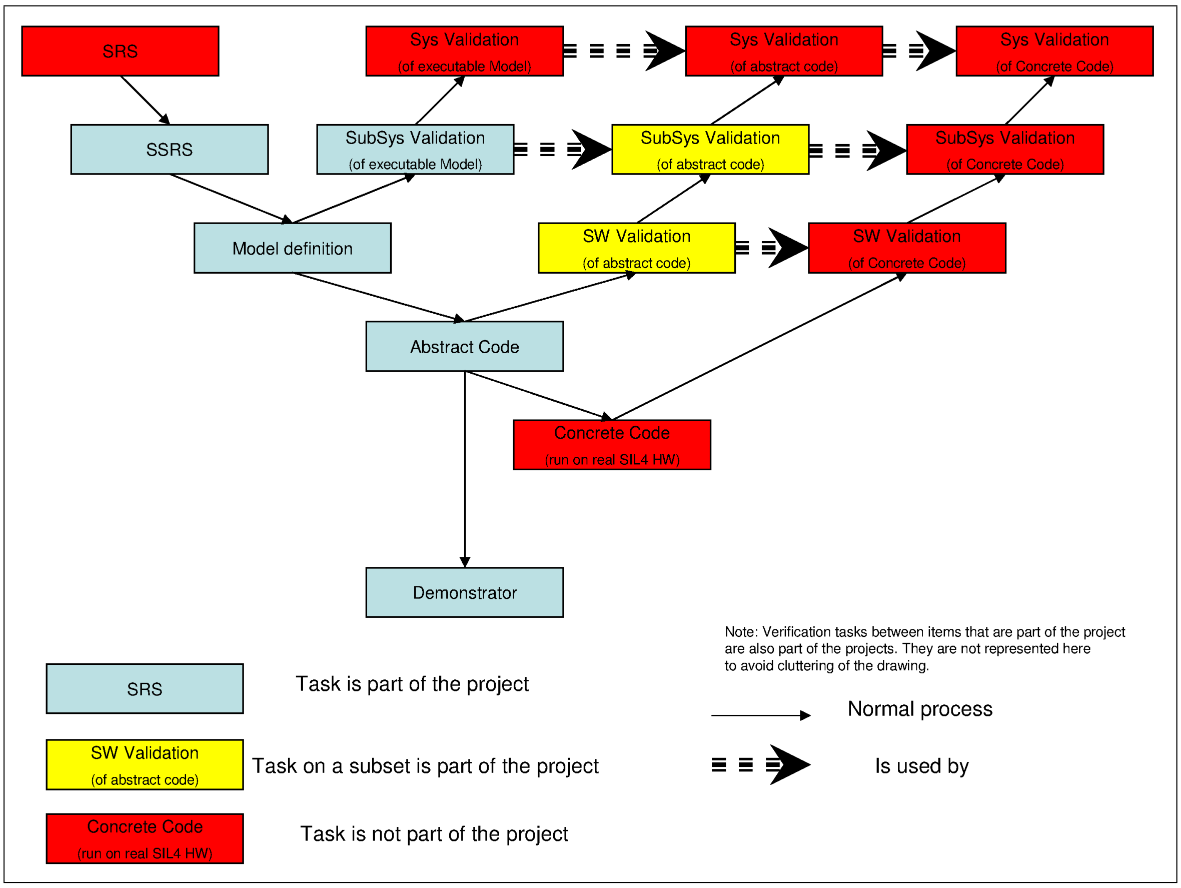

SRS
Sys Validation (of executable Model)
Sys Validation (of abstract code)
Sys Validation (of Concrete Code)
SSRS
SubSys Validation (of executable Model)
SubSys Validation (of abstract code)
SubSys Validation (of Concrete Code)
Model definition
SW Validation (of abstract code)
SW Validation (of Concrete Code)
Abstract Code
Concrete Code (run on real SIL4 HW)
Demonstrator
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.
SRS
Task is part of the project
Normal process
SW Validation (of abstract code)
Task on a subset is part of the project
Is used by
Concrete Code (run on real SIL4 HW)
Task is not part of the project

Figure 1. Main process

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.

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/D2.3.0-X-1 The project shall comply the CENELEC EN 50126 standard.

R-WP2/D2.3.0-X-2 The project shall comply the CENELEC EN 50129 standard.

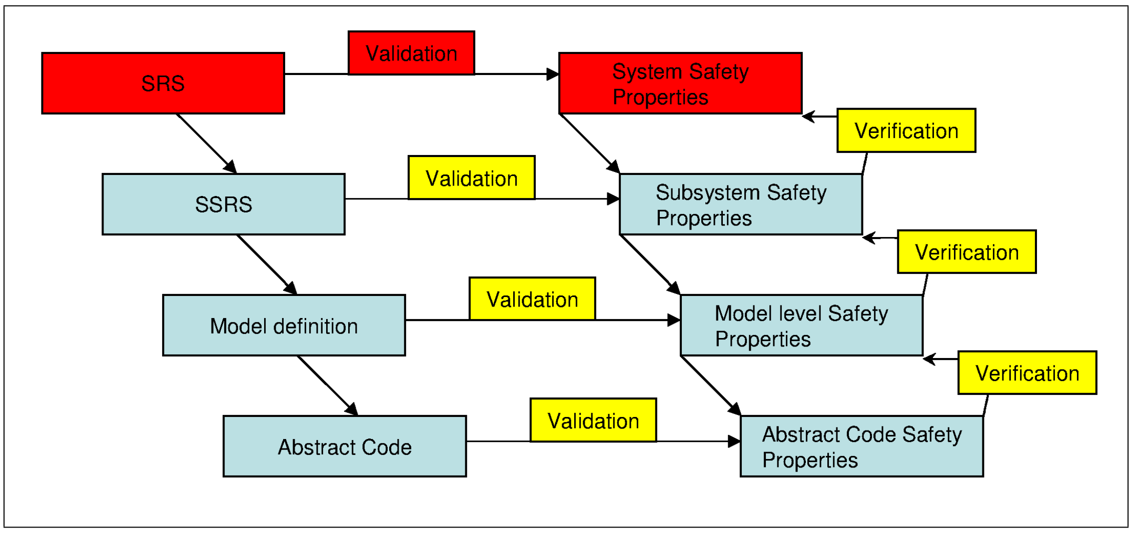


Figure 2. Safety analyses

Comment. These two requirements pulls 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.*

R-WP2/D2.3.0-X-3 All software in the scope of the project shall comply the EN 50128 standard.

R-WP2/D2.3.0-X-3.1 A QA Plan shall be issued and complied with.

R-WP2/D2.3.0-X-3.2 The QA Plan shall describe the management of versions, OpenETCS baselines, connexion with ERTMS baselines, requirements…

R-WP2/D2.3.0-X-3.3 A Verification plan shall be issued and complied with.

R-WP2/D2.3.0-X-3.4 A Validation Plan shall be issued and complied with.

R-WP2/D2.3.0-X-3.5 The techniques applied to the software will be compliant regarding the SIL.

R-WP2/D2.3.0-X-3.6 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. A documentation plan shall be issued, and include the full list of documents.

R-WP2/D2.3.0-X-3.7 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…*

[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.

R-WP2/D2.3.0-X-4 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.

7.2 Runtime Model & API

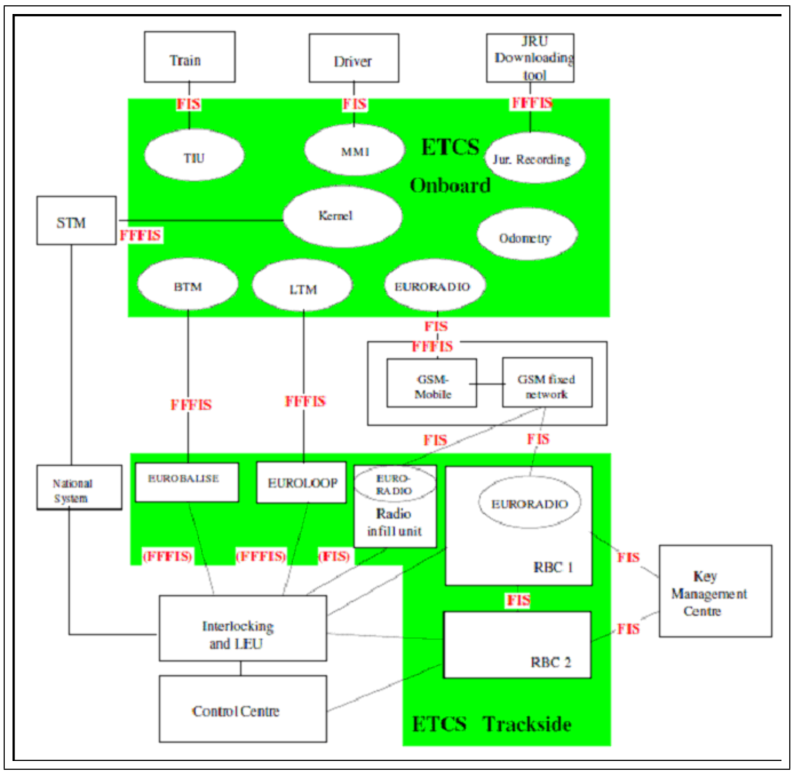


Figure 3. Architecture

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 “hardcoded” 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, 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.

7.2.1 RTM/API

This chapter has to be completed by the leader of D2.3.2

R-WP2/D2.3.0-X-5 The RTM/API model shall provide an abstraction layer of the hardware architecture.

R-WP2/D2.3.0-X-6 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/D2.3.0-X-7 The RTM/API shall allow discriminating Vital processing, data and I/O from Non Vital.

R-WP2/D2.3.0-X-8 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.

Open Issue. Should it be done for just SIL4 and SIL0, or should there be all levels from SIL0 to SIL4? It should be sufficient to distinguish only between safety and non safety, but an intermediate level could be useful (for SIL2 I/O for example). I have the feeling that it should not be more complicated to have one cluster of state machine per SIL, even if we only use SIL0 and SIL4.

R-WP2/D2.3.0-X-9 The RTM/API shall allow to introduce failures for test purpose.

R-WP2/D2.3.0-X-10 The RTM/API shall provide a way of reading configuration data (*e.g.* constants,. . .)

R-WP2/D2.3.0-X-11 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 D2.3.1 and D2.3.2

R-WP2/D2.3.0-X-12 The model shall comply all OBU ETCS mandatory requirements, in the functional perimeter provided by the Functional Architecture.

R-WP2/D2.3.0-X-12.1 The model shall comply the OBU part of SUBSET-026 version 3.3.0 or higher version.

R-WP2/D2.3.0-X-12.2 The reference ETCS baseline shall be modified only by project decision, according to the QA Plan.

R-WP2/D2.3.0-X-12.3 All divergences against the chosen baseline shall be documented and tracked, according to the QA Plan.

R-WP2/D2.3.0-X-13 The model shall be as close as possible (in terms of level of abstraction) to the Subset-026 and shall yield as few as possible "design choices".

R-WP2/D2.3.0-X-13.1 Complete bidirectional traceability between the model and the SRS shall be provided.

R-WP2/D2.3.0-X-13.2 Each interpretation, ambiguity and inconsistency of the SRS shall be indicated precisely.

R-WP2/D2.3.0-X-13.3 Each SRS requirement not formalized in the model (*e.g.* allocated to RBC) should be traced and justified.

R-WP2/D2.3.0-X-13.4 Each missing SRS requirement, needed in the model (*e.g.* any gap in the specification) should be clearly identified and justified.

R-WP2/D2.3.0-X-14 When the boundary of the formalized subsystem corresponds to a FIS or FFFIS, the Functional Architecture shall try to comply to it even when it is not mandatory.

R-WP2/D2.3.0-X-15 The Functional Architecture shall split the KERNEL into independent functions.

Open Issue. Should we stick to the SUBSET-026-4.5.2 function list? STAN: Is this really needed at the model level? Is this really possible? A lot of the functions are going to relate to each other. For instance the braking curves depend on the MA processing that depends on location principles, etc. According to our experience of modelling 33% of Subset-026, it is impossible to split the kernel in independent functions. The only possible separation is in different namespaces, where fonctions are grouped (like BTM, ODO, Radio session, Speed supervision, ...) However, one possible thing would be to track the dependencies between each namespace.

R-WP2/D2.3.0-X-16 The Functional Architecture shall allow a universal method of adding functions, removing functions and overwriting functions (modularity and model extensibility). For instance, the model should allow the disabling of the SH mode.

%%Design, Evolution, Validation, Transformation%%

8 Verification, Validation and Safety issues

This chapter has to be completed by the leader of D2.3.3

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 WP2 (equivalent to a preliminary hazard analysis) from the SUBSET-91 document.

They will be refined by the safety analysis process (WP4) into properties of the same level than the model. The process of doing so shall be described in the Safety Plan.

This will provide Safety Properties 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/D2.3.0-X-17 A safety plan shall be provided and complied with.

R-WP2/D2.3.0-X-18 The Functional Architecture shall identify the Vital and Non Vital functions.

R-WP2/D2.3.0-X-19 The subsystem shall be compatible with the THR required in the SUBSET-091.

R-WP2/D2.3.0-X-20 The safety analysis shall consider the Dread Event of the SUBSET-091, restricted to the scope of the subsystem.

R-WP2/D2.3.0-X-21 The model-level safety properties shall be written in a formal language.

%%To Be Confirmed%%

8.2 Verification and Validation

R-WP2/D2.3.0-X-22 A V&V plan shall be written and complied with.

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.

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 at the model level. Unit testing shall still be necessary at the code generator level.*

%%Verification that the refinements of system design to software code is coherent?%%

%%Verification that the proofs defined at system level design, are valids?%%

%%To Be Confirmed%%

9 Language and formalism

This chapter has to be completed by the leader of D2.3.1 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/D2.3.0-X-23 The model formalism shall be easily understandable by the domain experts.

R-WP2/D2.3.0-X-24 The safety properties should be provided in a declarative, simple and formal language.

R-WP2/D2.3.0-X-25 The formal model shall be transformable to others tools (SCADE, Simulink, B tools, OpenETCS tool chain %%STAN is this a requirement on the language?%% . . .

R-WP2/D2.3.0-X-26 Formal specifications should be able to formalize:

R-WP2/D2.3.0-X-26.1 State machines,

R-WP2/D2.3.0-X-26.2 Time-outs,

R-WP2/D2.3.0-X-26.3 Truth tables,

R-WP2/D2.3.0-X-26.4 Arithmetics,

R-WP2/D2.3.0-X-26.5 Braking curves,

R-WP2/D2.3.0-X-26.6 Logical statements.

R-WP2/D2.3.0-X-26.7 ERTMS 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/D2.3.0-X-27 The formal model shall be executable.

R-WP2/D2.3.0-X-27.1 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-27.2 The formal model shall support going back in time during a test, i.e. after having reached a certain step, to go backwards.

R-WP2/D2.3.0-X-27.3 The formal model shall support what-if analysis, i.e. to change the value of any model variable during the execution of a test in order to measure its impact.

R-WP2/D2.3.0-X-27.4 The formal model shall be self-explainable, i.e. shall be able to output its complete calculation tree for complex functions, calling each other

R-WP2/D2.3.0-X-27.5 The formal model shall support vizualization of the model elements activated during each step of a test

R-WP2/D2.3.0-X-27.6 The formal model shall support importing and executing the Subset-076 test sequences

R-WP2/D2.3.0-X-27.7 The environment shall be emulated by high level construction of the inputs.

R-WP2/D2.3.0-X-28 The formal model should have a meta-model.

R-WP2/D2.3.0-X-29 The formal model should have a meta-meta-model.

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/D2.3.0-X-30 It shall be possible to assert logical properties on the model (*i.e. invariants*).

R-WP2/D2.3.0-X-30.1 It shall be able to check the conformance of these properties at runtime.

R-WP2/D2.3.0-X-30.2 It shall be able to prove the conformance of the model to these properties.

10 Tool chain

This chapter has to be completed by the leader of D2.3.2

R-WP2/D2.3.0-X-31 The tool chain shall be composed only of Open Source components licensed under a license compatible with the EUPL license.

R-WP2/D2.3.0-X-31.1 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/D2.3.0-X-32 The tool chain shall be sufficiently robust to the management of the complete model, covering 100% of the Subset-026.[3]

R-WP2/D2.3.0-X-32.1 It shall allow modularity at any level (proof, model, software).

R-WP2/D2.3.0-X-32.2 It shall support meta-model evolution.

R-WP2/D2.3.0-X-32.3 It shall allow model-level refactoring, for instance rename model elements and move model elements.

R-WP2/D2.3.0-X-32.4 It shall allow the management of documentation within the same tool.

R-WP2/D2.3.0-X-32.5 It shall allow distributed software development.

R-WP2/D2.3.0-X-32.6 It shall include an *issue-tracking system*, in order to allow change management and errors/bugs management.

R-WP2/D2.3.0-X-32.7 It shall offer suitable code-generation interface and services.

R-WP2/D2.3.0-X-32.8 It shall allow to document/track the differences between the model and the ERTMS reference.

R-WP2/D2.3.0-X-32.9 It shall support management of subsequent Subset-026 versions, as well as differences tracking between Subset-026 versions.

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-32.10 It shall allow concurrent version development, or be compatible with tools allowing concurrent version development.

R-WP2/D2.3.0-X-32.11 The version management tools shall use model-based version control, instead of text-based version control.[4]

R-WP2/D2.3.0-X-32.12 In particular, it shall be made easy [5] to track the differences between two releases of a model and to manage conflicts.

[3]As a reference, the Subset-026 includes roughly 3200 requirements applicable to the OBU.

[4]As explained in Towards software configuration management for unified models, [1] "Traditional SCM systems operating on the abstraction of a filesystem and managing change at the granularity of textual lines are not adequate for these requirements. We propose a novel approach to SCM for unified models combining product versioning, operation-based deltas and change packages. To demonstrate feasibility we have implemented our approach in Sysiphus a suite of tools for collaborating over Software Engineering artifacts represented in a unified model."

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

R-WP2/D2.3.0-X-32.13 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-32.14 In particular, version management shall allow to track version of the safety properties together with the model.

R-WP2/D2.3.0-X-33 The tool chain shall include support for the following graphical user interface facilities:

R-WP2/D2.3.0-X-33.1 model syntax highlighting and auto-completion, for the parts of the model that are text-based

R-WP2/D2.3.0-X-33.2 graphical representation of Subset-026 braking curves

R-WP2/D2.3.0-X-33.3 tree-based display of model elements

R-WP2/D2.3.0-X-33.4 model animation and debugging

R-WP2/D2.3.0-X-33.5 perspective management: offer in a single graphical user interface several interconnected views for SRS, model and tests.

R-WP2/D2.3.0-X-34 The tool chain shall be platform independent, i.e. be portable accross windows, linux, mac, etc.

R-WP2/D2.3.0-X-35 The tool chain shall allow traceability between:

R-WP2/D2.3.0-X-35.1 the Subset-026 and the model, and vice-versa

R-WP2/D2.3.0-X-35.2 the Subset-026 and the tests, and vice-versa

R-WP2/D2.3.0-X-35.3 the model and the tests, and vice-versa

R-WP2/D2.3.0-X-35.4 the documentation and the model

R-WP2/D2.3.0-X-36 The tool chain shall allow traceability between the different layers of model and safety properties.

R-WP2/D2.3.0-X-37 The tool chain shall provide the following reporting facilities:

R-WP2/D2.3.0-X-37.1 detailed implementation metrics

R-WP2/D2.3.0-X-37.2 detailed traceability reports

R-WP2/D2.3.0-X-37.3 detailed model errors and warnings list

R-WP2/D2.3.0-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 the other tool.

R-WP2/D2.3.0-X-39 The tool chain shall conform to 50128 requirements, for the corresponding SIL and tool class.

R-WP2/D2.3.0-X-39.1 Each tool in the tool chain shall be classified among T1, T2 and T3

R-WP2/D2.3.0-X-39.2 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/D2.3.0-X-39.3 All T2 and T3 tools must be provided with their user manuals.

R-WP2/D2.3.0-X-39.4 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-39.4.1 . . . for data transformation,

R-WP2/D2.3.0-X-39.4.2 . . . for software transformation (*e.g.* translation, compilation. . .).

R-WP2/D2.3.0-X-40 The tool chain shall allow to write, store and execute*test cases* for the model.

R-WP2/D2.3.0-X-41 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. STAN: I confirm: if the test cases are generated based on the model, then any error in the model shall cause the generation of incorrect test cases, and shall therefore not be detected. It is a real added value to have two independent parts in the model: one model part, and one test part, with the test part including test cases for all functions in the model. As these tests make up a second implementation of the model, they enable a high-quality error detection in the model, also ensuring non-regression for the model.

Open Issue. TBD requirements on the prover. Should it verify De Bruijn's criterion[7]. *Should at least the proof tree be exportable and checkable in another tool? (if the proof tree itself is mandatory).*

References

[1] Maximilian Kögel. Towards software configuration management for unified models. In *Proceedings of the 2008 international workshop on Comparison and versioning of software models*, CVSM '08, pages 19–24, New York, NY, USA, 2008. ACM.

[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. . .)

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