openETCS/WP2/D2.2

openETCS

ITEA2

INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

ITEA2 Project
2012 – 2015

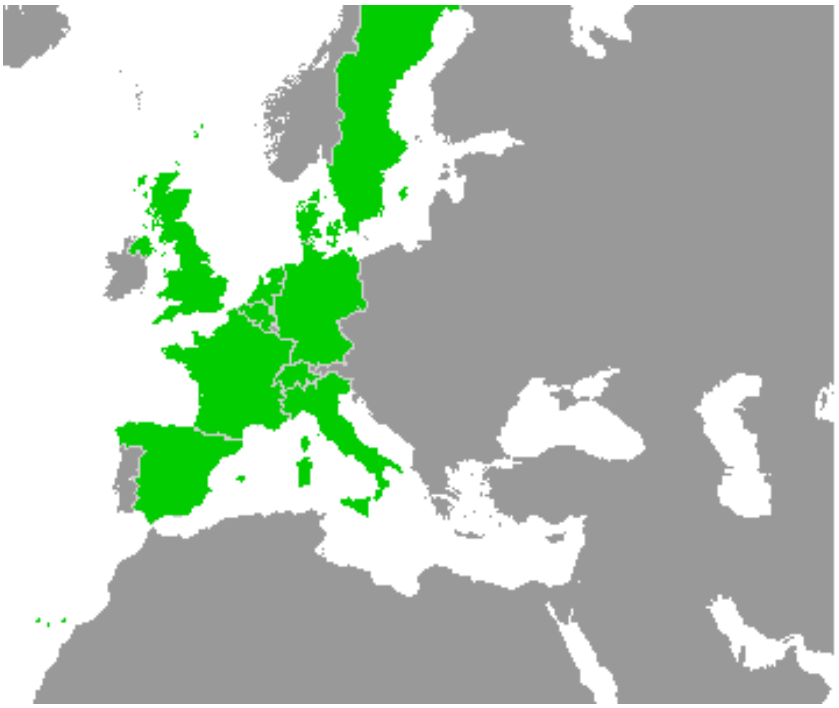
Work-Package 2: "Requirements for open proofs"

openETCS D2.2: Report on CENELEC standards

Set of requirements to be fulfilled according to the CENELEC standards

Merlin Pokam and Norbert Schäfer

April 2013



This page is intentionally left blank

openETCS D2.2: Report on CENELEC standards

Set of requirements to be fulfilled according to the CENELEC standards

Merlin Pokam and Norbert Schäfer

AEbt Angewandte Eisenbahntechnik GmbH
Adam-Klein-Straße 26
90429 Nürnberg
Germany

Intermediate Report

Prepared for ITEA2 openETCS consortium
Europa

Abstract: This report gathers requirements coming from the CENELEC standards for the railway sector particularly the CENELEC standard EN 50128:2011, that have to be fulfilled within the project openETCS, in order to prove that the openETCS application software and tools are fit for theirs intended purposes and respond correctly to safety issues that have been derived from the ERTMS System safety requirements specification and the risk analysis performed at the system level.

Document information	
Work Package	WP2
Deliverable ID or doc. ref.	D2.2
Document title	Report on CENELEC standards
Document version	00.01.00
Document authors (org.)	Merlin Pokam (AEbt)

Review information	
Last version reviewed	00.00.00
Main reviewers	Merlin Pokam, Jan Welte

Approbation			
	Name	**Role**	**Date**
Written by	Merlin Pokam	WP2-D2.2 Sub-Task Leader	See Document evolution
Approved by			

Document evolution			
Version	**Date**	**Author(s)**	**Justification**
00.00.00	10/02/13	Merlin Pokam	Document creation
00.01.00	20/03/13	Merlin Pokam	Updated, version for review
00.01.01	04/04/13	Merlin Pokam	The version of the report has been updated from "Preliminary" to "Intermediate" , version for review
00.01.02	08/04/13	Merlin Pokam	Updated, version for review

Table of Contents

Figures and Tables

Figures

Tables

Abbreviations

Terms	Definition
ATC	Automatic Train Control
FM	Formal Methods
CCS	Control command and signalling subsystems
CENELEC	(english) European Committee for Electrotechnical Standardization
COTS	Commercial off-the-shelf software, software defined by market-driven need, commercially available and whose fitness for purpose has been demonstrated by a broad spectrum of commercial users
CV	Curriculum Vitae
ETCS	European Train Control System
ERTMS	European Rail Traffic Management System
EUC	Equipment under control
FLOSS	free-libre / open source software
EN	European Norm
ERA	European Railway Agency
EVC	European Vital Computer
HW	Hardware
ISO	International Organization for Standardization
OBU	On Board Unit
OS	Operating System
PCA	Project Co-operation Agreement
PO FPP	Project Outline Full Project Proposal
RTOS	Real Time Operation Systems
SIL	Safety Integrity Level
SSIL	Software Safety Integrity Level
SRS	System Requirement Specification
STI	Standard Train Interface
TSI	Technical Specification for Interoperability
UNISIG	UNion Industry of SIGnalling

Table 1. Abbreviations

1 Terms and definitions

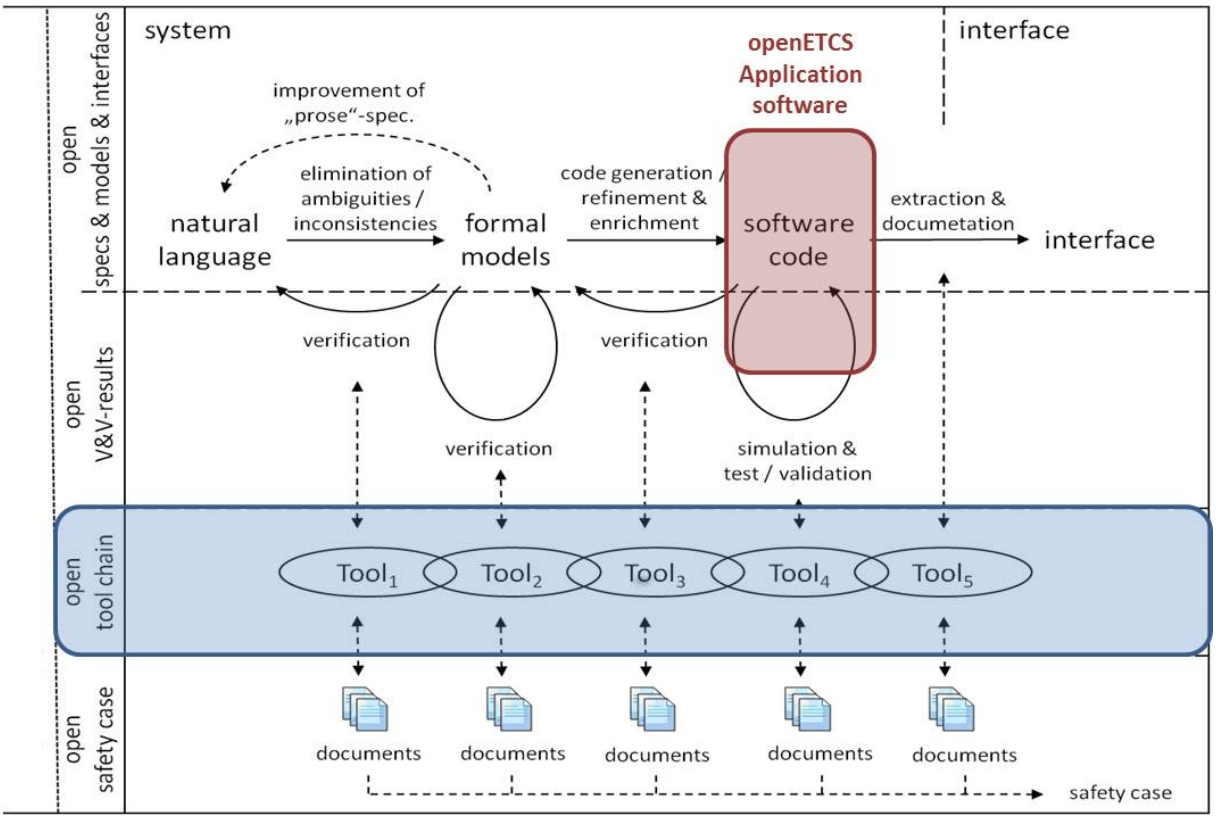


Figure 1. "open Proofs" the holistic approach

1.1 openETCS tools chain

Tools chain, that covers the entire software development process of the ETCS OBU, starting from a conventional natural language specification over a formalization of the ETCS OBU system description for the modeling with verification steps, through to code generation and automatic generation of documents, see Figure 1.

1.2 openETCS application software

Software code generated from formal models, see Figure 1 above.

1.3 Common terms and definitions

Terms	Definition	Source
formal specification	A formal specification is a concise description of the behavior and properties of a system written in a mathematically-based language, specifying what a system is supposed to do as abstractly as possible, thereby eliminating distracting detail and providing a general description resistant to future system modifications. The most formal specifications are written in a language with a well-defined semantics that supports formal deduction and allows the consequences of the specification to be calculated through proof of putative theorems.	[17]
formal proof	A formal proof is a complete and convincing argument for the validity of a statement about a system description. A proof proceeds in a series of steps, each of which draws conclusions from a set of assumptions. Justification for each step is derived from a small set of rules which state what conclusions can be reasonably drawn from assumptions. Such justification eliminates ambiguity and subjectivity from the argument. Formal proofs may be prepared manually or, preferably, with the assistance of an automated FM tool.	[17]
Open-Source-Software	Source code available to the general public with relaxed or non-existent copyright restrictions	EN 50128:2011
Pre-existing software	All software developed prior to the application currently in question is classed as pre-existing software including: • COTS (commercial off-the-shelf) and open source software, • Software previously developed.	EN 50128:2011
Open Proofs	An "open Proofs" is software or a system where all of the following are free-libre / open source software (FLOSS): • the entire implementation, • automatically-verifiable proof(s) of at least one key property, and, • required tools (for use and modification).	www.openproofs.org
Software	Intellectual creation comprising the programs, procedures, rules, data and any associated documentation pertaining to the operation of a system	EN 50128:2011
Application software	Part of the software of a programmable electronic system that specifies the functions that perform a task related to the EUC rather than the functioning of, and services provided by the programmable device itself	IEC 61508-4:2010

Table 2. Terms and definitions

2 Introduction and motivation

In all over Europe there are about 30 different, mostly not compatible signaling and train protection systems in use, see Figure 2. For a unified european rail system it is very costly to maintain this diversity of signaling systems and therefore the european commission has set new rules by so called Technical Specifications for Interoperability (TSI, see the latest decision [15]) with the goal to implement a unified "European Train Control System (ETCS)".

ETCS is intended to replace national legacy signaling and train control systems and consists of facilities in the infrastructure and on-board units (OBU). ETCS is a so called cab-signaling system, which means that in principle all commands for the driver are shown on screens inside the driver's cabin, making conventional track-side signals obsolete, resulting in considerable savings for the infrastructure operators. The shift of functionality and safety responsibilities from the infrastructure into the vehicle has caused an increase of complexity for the on-board equipment. In terms of technology, this migration is mostly done by software. While electronic hardware is getting continuously cheaper, the high complexity of the safety critical software has caused significant cost increases for development, homologation and maintenance of the ETCS. Despite the fact that several major European suppliers with substantial knowledge in signaling technology have worked on a common System Requirement Specification (SRS, see the latest baseline [16]) for over a decade, the main goal of interoperability has not yet been accomplished. Up to now, not a single ETCS onboard unit has been approved to operate on all existing European ETCS lines. One of the reasons is given by the fact that a plain English specification text "prose" of some complexity cannot be so precise and free of potential divergent interpretation that the resulting software products would behave identical, see Figure 3. Therefore the development of ETCS has to be considered as "work in progress", resulting in many software upgrades to be expected in the near and distant future.

Almost all products on the market are based on different proprietary software designs, which results in a life-long dependency from the original manufacturers causing high life-cycle costs for vehicle owners. The key element for improving that situation seems to be a greater degree of standardization for: Hardware, software, methods and tools. An approach following the open source idea, called "openETCS" utilizing concepts from the automotive and aviation industry, has been suggested, not only covering the embedded application software of the ETCS onboard

Figure 2. Substitution of approximately 30 different signaling and ATP systems by just one single system: the European Train Control System (ETCS)

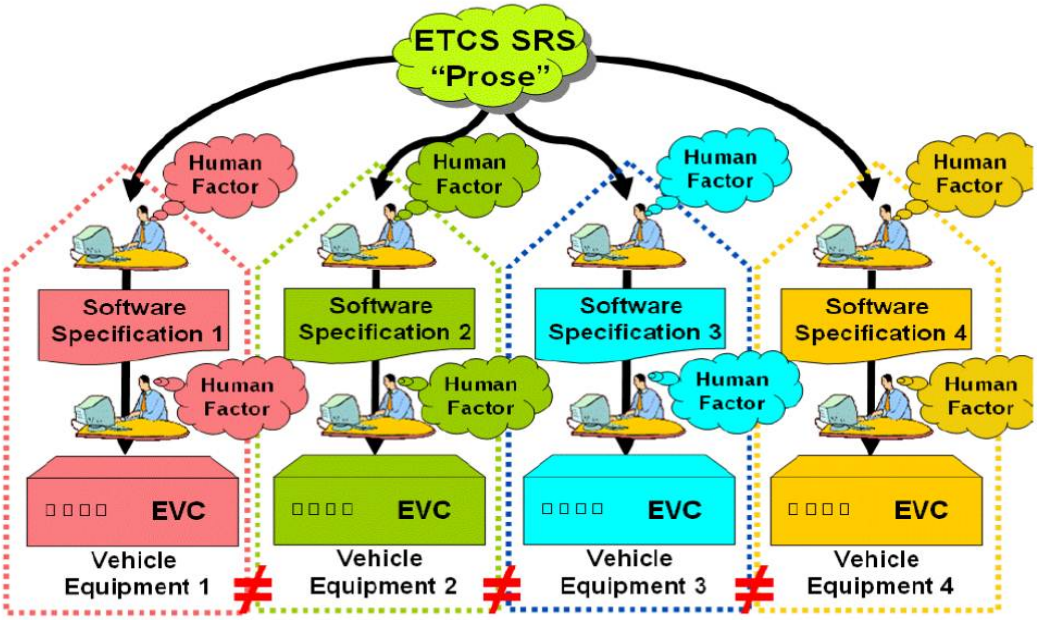


Figure 3. Divergent interpretation of a common and mandatory public domain ETCS SRS document, due to the "human factor" by parallel working, not closely cooperating manufacturers, causing different software solutions with deviant reaction patterns, which results in interoperability deficiencies and costly subsequent retrofit works

unit itself, but including all tools and documents in order to make the entire product life cycle as transparent as possible and make it comprehensible for third parties. Making the proof of safety open to the public has been called "open proof" and is new to the railway sector, see Figure 4.

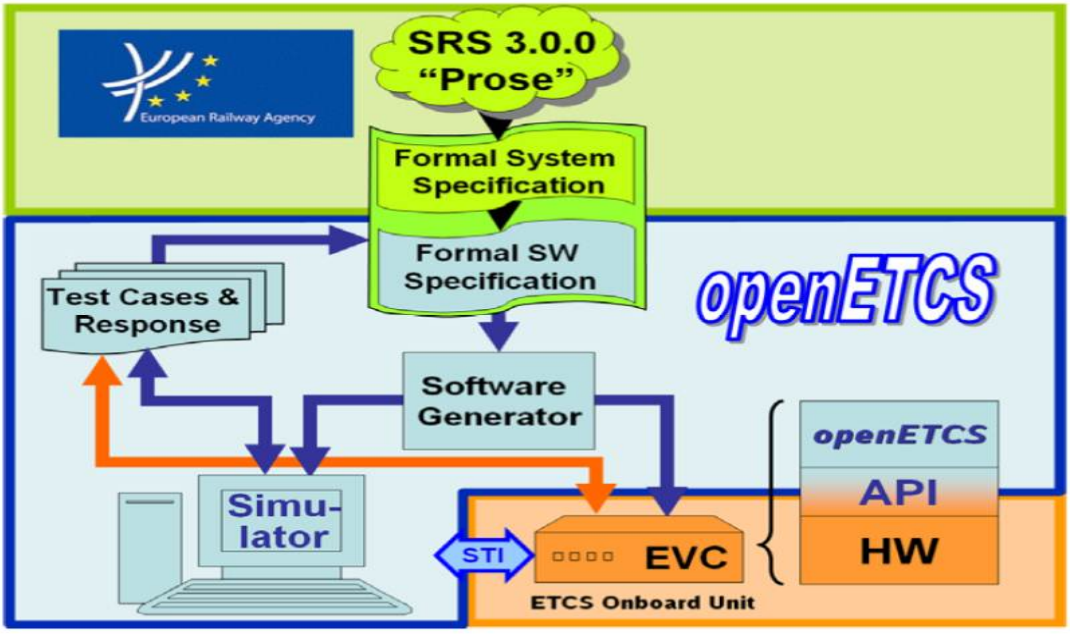


Figure 4. openETCS approach

3 What is openETCS?

3.1 Description

A detailed description of the project openETCS is given in the Project Outline Full Project Proposal, see [9].

3.2 Main goals and deliverables

The objective of the openETCS project is to provide a usable open source application software; including tools, documentations and the safety case for all ETCS OBU. That means, are made available as FLOSS under a "General Public License" (e.g. EUPL: European Union Public License).
According to the project co-operation agreement [10], the main goals and deliverables of the openETCS project are:

1. Create a formal specification of the ETCS OBU functionality according to UNISIG Subset 026,
2. Generate an executable software package from the formal specification and integrate the generated software package in the target hardware (non vital integration) for laboratory test, simulation and reference purposes,
3. Develop a tools chain supporting both previous bullet points including: code, test case and document generation, meeting CENELEC EN 50128:2011 requirements and certifiable for SIL4 software applications for signalling equipment.

4 Purpose and creation of this report

4.1 Purpose of this report

Due to the particularity of the openETCS project and taking into account its goals, it should be specify in advance which requirements shall be fulfilled in order to be compliant with the CENELEC standards, particularly the CENELEC standard EN 50128:2011. This work was carried out and coordinated by AEbt and the result is this report. Therefore, this report lists requirements that must be fulfilled in order to prove, that the openETCS project deliverables (as described in section 3.2) are fit for theirs intended purpose and responds correctly to safety issues that was derived from the ERTMS system safety requirements specification.

4.2 Creation of this report

Basis for the realisation of this report were:

- Articles and other technical literature
- Existing processes in the signalling industry
- CENELEC standards

This work was also carried out through interviews with experts. The experts interviewed were industrial experts, signalling experts and standardisation experts.

5 Results

This section describes the result of the performed work.

5.1 Safety requirements to be fulfilled according to the CENELEC standards

5.1.1 Safety requirements to be fulfilled according to the CENELEC standard EN 50128:2011

Safety requirements according to the CENELEC standard EN 50128:2011 that shall be fulfilled by the openETCS tools chain and the openETCS application software, are recorded in annex A and C.

5.1.2 Safety requirements to be fulfilled according to related CENELEC standards

When developing software, it is mandatory to prove that the actual behavior of the software, when executed, will be consistent with the behavioral semantics of the software. Behavior is observable in:

- The data domain,
- The time domain,
- The causal domain.

All of these behavioral aspects depend on both the OS and the underlying HW. For example:

- Data transformations fail if, the word length of the HW registers is inappropriate for the calculations involved,
- Expected reactions miss their deadlines if, the OS scheduler does not allocate appropriate amounts of CPU time to the task processing the input,
- Events occur in the wrong order if the OS does not provide adequate mechanisms for critical section management.

As a consequence the openETCS application software can only be validated and certified in relation to certain OS and HW capabilities.

Concerning the HW, it shall fulfill the requirements for safety-related hardware as defined in the CENELEC standard EN 50129 (see [4]) according to the required safety integrity levels resulting from the risk analysis. In other words, that means, before applying the EN 50129, a hazard analysis and risk assessment processes as defined in EN 50126 (see [2])shall be performed, in order to specify the System Safety Requirements. It is expected that the System in which the openETCS application software will be implemented, will exchange safety-related data with its environment, therefore, additional requirements for safety-related data communication as defined in EN 50159 (see [5]) shall also be fulfilled

Concerning the OS, We propose to elaborate specification documents about OS capabilities required. we expect that the EVC code generated from the openETCS models will reference the

interface of some OS application program interface (API) providing the services necessary to ensure the proper behavior of the software to be executed. Therefore, The API to be developed within the openETCS project shall be compliant with the requirements of EN 50128 (see [3]) according to the required software safety integrity levels resulting from the risk analysis. Instead of specifying OS-API properties from scratch we suggest to adopt the well known OS-API speciiňĄcation from the ARINC 653 standard (see [8]) for which OS-API have been developed and applied successfully for safety-critical tasks in the avionic domain.

5.2 Security requirements to be fulfilled

Security Requirements are not clearly described in the CENELEC standards, nevertheless due to the nature of the openETCS project and based on interviews performed with software and signalling experts, we recommend to take into account the recommendations described in sections 5.2.4, 5.2.4 and 5.2.4.
Sections 5.2.1, 5.2.2 and 5.2.3 give a brief description on familiar security patterns. A detailed description on all existing security patterns is given in [21].

5.2.1 Secure Design Patterns

During the design of the openETCS application software and tools, security can be enhanced through the implementation of secure design patterns. Three of the most used techniques are listed below:

1. Distrustful decomposition
2. Privilege separation
3. Clear sensitive information

Distrustful decomposition

The intent of the Distrustful Decomposition secure design pattern is to move separate functions into mutually untrusting programs, thereby reducing:

- the attack surface of the individual programs that make up the system
- the functionality and data exposed to an attacker if one of the mutually untrusting programs is compromised

Privilege separation

The intent of the Privilege separation is to reduce the amount of code that runs with special privilege without affecting or limiting the functionality of the program. The Privilege separation is a more specific instance of the distrustful decomposition.

Clear sensitive information

The use of this pattern ensures that sensitive information is cleared from reusable resources before the resource may be reused. It is possible that sensitive information stored in a reusable

resource may be accessed by an unauthorized user or adversary if the sensitive information is not cleared before freeing the reusable resource.

5.2.2 Secure coding Patterns

During the implementation of the openETCS application software and tools, software security can be augmented by avoiding a number of common software security vulnerabilities. The following software security vulnerabilities shall be avoided:

1. Buffer overflow
2. Pointer shenanigans
3. Dynamic memory allocation flaws
4. Tainted data

Buffer overflow

Buffer overflow can lead to more serious consequences, such as stack smashing, code injection, or even arc injection by which an attacker changes the control flow of the program by modifying the return address on stack. In arc injection, an attacker doesn't even have to inject any code, and he can jump to an arbitrary function in existing code, or bypass validity checks or assertions [13].

Pointer shenanigans

If an attacker can modify a data pointer, then the attacker can point to wherever he likes and write whatever he likes. If an attacker can overwrite a function pointer, the attacker is well on his way to executing his code on the processor [13].

Dynamic memory allocation flaws

The use of dynamic memory allocation shall be forbidden within the openETCS project. It is so easy to write defective code for dynamic memory allocation, so that attackers are eager to search out these defects [13].

Tainted data

Data entering an embedded system from the outside world must not be trusted. Instead, it must be "sanitized" before use [13]. A useful technique for data sanitization is called "white listing". It involves describing all possible valid values for a given piece of data and then writing code that only accepts those values. All unexpected values are viewed as "tainted" and are not used.

5.2.3 Backdoor

A backdoor can be created by each person who has access to the software source code, but it is also possible to create a backdoor without modifying the software source code, or even modifying

it after compilation. This can be done by rewriting the compiler so that it recognizes code during compilation that triggers inclusion of a backdoor in the compiled output. When the compromised compiler finds such code, it compiles it as normal, but also inserts a backdoor. This attack was first outlined by Ken Thompson in his paper "Reflections on Trusting Trust" see [12].

5.2.4 Recommendations for security requirements

Regarding Secure design Patterns

The architecture and design specification of the openETCS application software and tools shall implement the requirements described in section 5.2.1.

Regarding Secure coding Patterns

Since the software code of the openETCS application software will be automatically generated using tools, tools used or developed shall avoid the software security vulnerabilities described in section 5.2.2.

Regarding Backdoors

A useful technique against back doors is the so called "Diverse Double-Compiling"-technic see [11]. But, experiences from the past have shown that open source software are better positioned against backdoors than proprietary software. For this reason, it is clear from the report of the European Parliament, that: "Calls on the Commission and Member States to promote software projects whose source text is made public (open-source software), as this is the only way of guaranteeing that no backdoors are built into programmes" see [14]. Since openETCS is an open source software project and according to [14], no requirement regarding backdoor issues need to be considered.

5.3 Is "open proofs" suitable for safety railway applications?

As mentioned in section 1; a software or system is an "open proof" if all of the following are FLOSS:

1. the entire implementation (requirements, design, code, required documentation for use/maintenance, etc.),
2. automatically-verifiable proof(s) of at least one key property, and
3. all required tools needed for use and modification of the software or system.

According to (http://www.openproofs.org), something is FLOSS if it gives anyone the freedom to use, study, modify, and redistribute modified and unmodified versions of it, meeting the free software definition and the open source definition.

In a globally meaning, "open Proofs" embraces two approaches: formal methods and the freely availability of the software or system to others (in this report referred as open source).

5.3.1 Suitability of open source process for safety relevant railway applications

By order of the Deutsche Bahn AG (DB AG), AEbt has created an assessment report on the suitability of open source software for safety relevant railway applications. In this Report, the assessor came to the conclusion that open source software regarding their safety integrity should be treated just as proprietary software. In certain areas, the use of open source software for safety relevant applications is even recommended; because software bugs or intentionally programmed backdoor will be early detected by independent programmers or the "community", see [18].

5.3.2 Suitability of formal methods for safety relevant railway applications

This section provided general information of the impact of introducing and integrating formal methods (FM) into the development process.

Formal methods for developing software embrace two techniques: formal specification and formal verification. Both are established based on elementary mathematics, such as set of theory, logic and algebraic theory [19].

When establishing formal methods on a project, there are basically two types of considerations, one of which is largely administrative, the other largely technical, [17].

A summary of the each appears below.

Administrative Factors:

- **Project Staffing:** The team responsible for planning the role of FM on a project should include at least one person knowledgeable in FM and one person knowledgeable about the application domain. The team responsible for applying FM must have FM expertise or be provided with hands-on training.
- **Project Scale:** The scale of the project should be taken into consideration. If project staff has little or no previous FM experience, an initial study may be advisable either as a final objective or as a leading to the full-scale project.
- **FM Training:** The training available to those project staff responsible for applying FM should be rigorous and include hands-on experience with the tool(s) and type of application that will be encountered on the project.
- **Process Integration:** The strategy for integrating FM into a new or existing process should be thoroughly planned and documented, preferably early in the project.
- **Project Guidelines:** Project guidelines, standards, and conventions, both for documentation and specification, should be developed early and adhered to.

Technical Factors:

- **Type of Application:** FM are not equally appropriate for all applications; they are best suited to analyzing complex problems, taken singly and in combination, and less suited for numerical algorithms or highly computational applications..
- **Size and Structure of Application:** The size and structure of an application determine the difficulty of using FM; ideally, applications should be of moderate size (guidance on how to assess size will be addressed in this item's section below), decomposable into subsystems or components, and based on a coherent underlying structure.

- **Type of Analysis/Formal Method:** The type of analysis, i.e., the reasons for applying FM, determine the most appropriate level of formalization and the most suitable FM and FM tools. Objectives in using FM range from producing clear, unambiguous documentation to mechanically verifying the correctness of crucial algorithms or components.
- **Levels of Rigor in FM:** FM may be applied at varying levels of rigor. The rigor, or extent to which a method is "truly formal" and "really calculates," can range from the occasional appearance of mathematical notation in an otherwise informal document, through "rigorous" methods that employ a standardized specification language, to "fully formal" methods that make use of mechanically-checked theorem proving.
- **Scope of Formal Method Use:** There are at least three dimensions to the scope of formal method use: (1) all/selected stages of development life cycle, (2) all/selected system components, (3) full/selected (system) functionality.
- **Type of Formal Method Tool:** The choice of FM tool, if any, should be directly determined by the application profile generated by evaluating the five preceding factors. Primary considerations include the type of specification language and the need for mechanical proof support.

Administrative and technical considerations are closely coupled, each having implications for the other. This is because the process of determining whether a given application is a good candidate for FM is not cut and dried and because the use of FM entails a serious technical commitment by project staff and a corresponding commitment to support and invest in the FM activity on the part of management.

Detailed description of the technical considerations

Formal methods cover a wide range of techniques that have different characteristics and utility. This section describes the scope and implications of these differences with respect to five technical factors that should be evaluated when considering the use of FM for a given application. The factors are introduced in the suggested order of consideration (according to [17]); e.g., before choosing a formal method tool, it is important, first, to define the type and scope of application, second, to specify the type of analysis to be performed and third, to determine the rigor and scope of the analysis.

Type of Application FM are not equally suitable for all types of applications. Although, in principle, the methods can be applied to nearly any application, in practice, the benefits that can be realized and the difficulty of achieving them will differ significantly from one application to another and from one subsystem to another within a single application. Suitability should be evaluated with respect to the characteristics of the problem domain and their implications for the modeling domain. Higher complexity applications stand to gain from FM much more than lower complexity ones simply because less complex problems can be solved dependably using less rigorous methods. Of particular interest are problem domains whose complexity stems not so much from the size and structure of the design, but from inherently difficult algorithms such as those for fault tolerance and parallel or distributed processes. A further consideration is the mathematical domain of discourse. Applications that are heavily based on numerical processing, especially those using floating point arithmetic, pose some difficulties for FM, while those that can be modeled using the domains of logic and discrete mathematics benefit from easier formalization, more tractable reasoning, and better FM tool support.

Size and Structure of Application The size of an application is a major factor in the cost and difficulty of its formalization. Usually, FM are most effectively applied to systems or subsystems

of moderate size; currently, FM cannot be applied in full to the largest systems implementable using conventional programming techniques. An alternative is to limit the scope of the formal method activity to critical properties or components of a very large system, assuming, of course, that the system is decomposable into small or medium-sized subsystems or components with well-defined interfaces. This clean structuring property is vital in any medium- or large-scale application to ensure that the results of separate FM analyses can be combined and valid inferences drawn about the composite behavior of cooperating subsystems. A second structural property, loosely referred to as structural entropy, is also important. If an application has intrinsically high entropy, i.e., is primarily a random collection of special cases with weak cohesion or few unifying principles, little can be expected from a formalization activity. Conversely, if an application exhibits strong underlying structural principles, well understood and easily expressed in a logically meaningful way, FM can effectively capture and exploit this structure.

Type of Analysis/Formal Method The type of analysis or formal method to be employed is determined largely by project objectives; the purpose for which FM are to be applied should be clearly defined and explicitly documented. For example, one application may use FM primarily to develop specifications for documentation, another may exploit the precision inherent in formally specified requirements to catch errors early in the life cycle, a third may use FM to analyze and assure the correctness of critical properties or algorithms. These equally legitimate objectives have very different implications for the rigor of the formal method analysis and the type of formal method tool appropriate for the project.

Levels of Rigor in Formal Methods FM techniques may be applied at varying levels of rigor. Here, rigor is used in a technical sense to mean the degree of formality of a method, i.e., the extent to which a method formulates specifications in an axiomatic style, explicitly enumerates all assumptions, and reduces proofs to explicit applications of elementary rules of inference. Increasing formality allows the products of FM (i.e., specifications and proofs) to be less dependent on subjective reviews and consensus and more amenable to systematic analysis and replication. Usually, increasing formality is associated with increasing dependence on mechanical support. Suggested levels of rigor according to [17]

- **Level 1:** Use of manual review and inspection, relying on documents written in a natural language, pseudo code, or programming language, possibly augmented with diagrams and equations, and validated with conventional testing techniques. Activities at this level are not "formal" in a strict sense, but represent current recommended practice, and serve as a baseline of discipline and structure necessary to support the additional activities at higher levels of formality.
- **Level 2:** Use of notations and concepts derived from logic and discrete math to develop more precise requirements statements and specifications. Proof, if any, is informal. This level of FM typically augments existing processes without imposing wholesale revisions.
- **Level 3:** Use of formalized specification languages with mechanized support tools ranging from syntax checkers and pretty printers to type checkers. This level of formality usually includes support for modern software engineering constructs, e.g., modules, abstract data types, and objects, all with explicit interfaces, but has not historically offered mechanized theorem proving.
- **Level 4:** Use of fully formal specification languages with rigorous semantics and correspondingly formal proof methods that support mechanization. State exploration, model checking, and language inclusion technologies also exemplify this level, although these technologies are highly specialized, automatic theorem provers that are limited to checking properties of finite-state systems.

Higher levels of rigor are not necessarily superior to lower levels; factors that determine the appropriate level of rigor include: project objectives, criticality of the application, and available resources. For example, if FM are used simply as documentation, Level 2 may be appropriate; if they are used to justify the design of a new and critical component, Level 4 may be the best choice. On the other hand, routine applications adequately handled by conventional processes are probably most appropriately left to Level 1. Finally, it is possible to use a formal method at a level of rigor lower than its ultimate capability, e.g., by using the specification language, but not the theorem-proving capability of a Level 4 formal method.

Scope of Formal Method Use The extent to which FM are applied can also vary. There are at least the following three dimensions to the notion of extent.

1. **All or selected stages of the development life cycle:** It is generally felt that the biggest payoff from the use of FM occurs in early life cycle stages, given that errors become more expensive to correct as they proceed undetected through later development stages; early detection leads to lower life cycle costs. Moreover, the use of FM in the early stages provides additional precision where it is currently most needed in the conventional development process.
2. **All or selected system components:** Criticality assessments, assurance considerations, and architectural characteristics are among the key factors used to determine which subsystems or components to analyze with FM. Since large systems are typically composed of components with widely differing criticalities, the extent of formal method use should be dictated by project-specific criteria. For example, a system architecture that provides fault containment for a critical component through physical or logical partitioning provides an obvious focus for FM activity and enhances its ability to assure key system properties.
3. **Full or selected system functionality:** Although FM have traditionally been associated with "proof of correctness," i.e., ensuring that a system component meets its functional specification, they can equally well be applied to only the most important system properties. Moreover, in some cases it is more important to ensure that a component does not exhibit certain negative properties or failures, rather than to prove that it has certain positive properties, including full functionality.

These are the three most commonly used variations on the extent of FM application, although others are certainly possible. Varying the degree of rigor along each of these three dimensions yields a wide range of options and provides maximal benefit from a limited investment in FM.

Type of Formal Method Tool The choice of tool is dictated by the application profile defined by consideration of all of the preceding factors, although the issue of tools is clearly moot if the most appropriate level of rigor falls below Level 3. For example, Level 3 documentation of sequential components is consistent either with a typical Level 3 notation supported by a type checker, or, if more powerful mechanization and stronger guarantees of consistency are desired, with a system normally used to support Level 4. Similarly, when choosing a Level 4 tool, the capability of the tool, the constraints of the problem domain, and the objectives of the analysis must be well matched. For example, verifying the correctness of fault-tolerant algorithms is probably best pursued with a general-purpose theorem prover, while exploring the properties of mode-switching or other complex control logic is probably more effectively pursued with a state-exploration system. The process of selecting a formal method tool is in many ways similar to selecting any other software system; the usual considerations of documentation, tutorials, history of use, ease of use, etc. apply. In this case, effective support for the selected formal method(s) is also important. A suggestive, but by no means exhaustive, list of the additional considerations necessary for judicious tool selection appears below.

1. **Specification Language:** Is the language adequately expressive for the given application and which of the following features important for the application does the language offer: well-defined semantics, modern programming language constructs (including support for abstraction, modularity, and encapsulation), familiar and convenient syntax, strong typing, encapsulation, parameterization, built-in model of computation, executable subset or other provision for animating specifications, support for state exploration, model checking, and related methods?
2. **Theorem Prover:** Does the FM tool offer a theorem prover or proof checker? If so, how is the theorem prover controlled and guided; is there automated support for arithmetic reasoning, efficient handling of large propositional expressions, and rewriting; what support is there for developing and viewing the proof; how is the proof presented to the user (e.g., user input or canonical expressions, with or without quantifiers); are the foundations (i.e., all axioms, definitions, assumptions, lemmas) of the proof identified; are there facilities for editing proofs; is it reasonably easy to re-verify a theorem after slight changes to the specification?
3. **Utilities:** Does the formal method offer a reasonably comprehensive library of standard types, functions, and other constructions and is the library validated; what, if any, editing and document preparation tools does the system provide; are there facilities for cross-referencing, browsing, and requirements tracing; is there support for incremental development across multiple sessions and for change control and version management?

5.3.3 Position of standards regarding formal methods in the railway sector

In EN 50128:2011, Formal Methods/Proofs are explicitly identified as relevant technique/measure for software requirements specification, software architecture, software design, implementation, verification and testing and data preparation techniques. More precisely they are "recommended" for SIL levels 1 and 2 and "Highly Recommended" for SIL levels 3 and 4. The standard puts additional constraints on tools, especially code/data generation tools with respect to the need of a specification and evidence that the implementations complies with the specification. Unfortunately, formal methods have not spread in the whole railway signalling industries, where much software is still written and tested in traditional ways. This lack of adoption is due to the investments needed to build up a formal methods culture, and to the high costs of commercial support tools. Moreover, equipment can conform to CENELEC without applying formal methods [20]. Another barrier is that, the certification bodies might not be familiar with FM as most of the systems they certify follow a test-based approach. For systems whose justification relies on such formal argument, the certification body might require additional information to be convinced. Typically, one should foresee some specific training-level information in the certification process. Once acquired, new system can easily follow the same path. For example siemens has gone through this process with the B-method, and it is now accepted by the certification bodies, so that the next projects have become easier to certify.

In summary: The EN 50128:2011 highly recommend formal methods but do not really describe how to manage a formal development. It is therefore difficult to prove that the development process comply with the standards. Certification authorities have to be convinced about this issue. As formal methods are less widespread and certification authorities are less familiar with them as compared to the classical development methods, there is more work to be done in comparison with other methods, especially the first time, and despite the fact that better argument could be provided. However the investment might be worth the effort as shown by the Siemens case for B-method.

6 Summary

As task leader of the task D2.2 (according to [9]), AEbt has coordinated all the work. Basis, were the procedures listed in the section 4.2 of this report. The result is a set of requirements, which shall be fulfilled in order to assess tools of the openETCS tools chain as T1, T2 or T3 support tools according to EN 50128:2011. Regarding the openETCS application software, it was assumed during the creation of this document, that the whole or part of the software will be SIL4 compliant according to EN 50128:2011.

Requirements in appendix A and C with suggested implementations are general requirements that any SIL4 Software for railway applications and tools shall fulfill. For specific projects, there can still be minor deviations, which may have to be agreed with a software assessor.

Based on the performed assessment and interviews, we are of the opinion that "open proofs" is suitable for safety railway applications, when the process applied to develop, test, deploy and maintain the software and tools fulfills all the requirements of the CENELEC standard EN 50128:2011 as well as some security requirements.

In this report reference was made to other CENELEC standards (see EN 50126 [2], EN 50129 [4], EN 50159 [5] and ARINC [8]); which however have not been detailed. Related to the standard ARINC, an analysis shall be performed, in order to show how it can be helpful for the openETCS project. Regarding the other CENELEC standards Since these standards (particularly EN 50126 and EN 50129) define the process of specifying the safety functions allocated to software, we recommend to include them in an earlier phase of the project already.

Appendix A: Tools development: state of the art regarding CENELEC EN 50128:2011

A.1 Introduction

This section gathers requirements, which shall be fulfilled by tools within the openETCS tools chain. Since some of the tools of the openETCS tools chain not really need to be T3 compliant, the requirements to tools were therefore subdivided in 3 subsections:

- Section A.2, gathers requirements for tool in class T1,
- Section A.3, gathers requirements for tool in class T2 and,
- Section A.4, gathers requirements for tool in class T3.

A.1.1 How to read the tables used in this appendix

Tables used in this appendix consist of three columns.

- **Column 1** lists the requirements comming from the CENELEC standard EN 50128:2011. But since the standard is protected by copyright, only the requirements identification numbers are recorded there. The same requirements identification numbers (subclause) as defined in the EN 50128:2011 have been used. The textual requirement should therefore be read in the standard using the requirements identification number recorded in column 1 as reference. In order to simplify the readability, some of the requirements have been grouped, especially when they address the same subject; only few were left out, because they were not relevant for the purpose.
- **column 2** describes how the various requirements can be fulfilled. Everywhere where possible, recommendations were formulated.
- **column 3** gives a suggestion of documents that shall be created or provided, in order to provide the evidence.

A.2 Requirements for tools in class T1

Some examples of T1 tools:

- a text editor tool with no automatic code generation capabilities,
- a requirement support tool with no automatic code generation capabilities,
- a design support tool with no automatic code generation capabilities,
- configuration control tools.

Table A1. Requirements for tools in class T1

Requirements	How the evidence shall be provided	Documents to be created
6.7.4.1	Provide the evidence that tools in class T1 cooperate. NOTE Tools cooperate if the outputs from one tool have suitable content and format for automatic input to a subsequent tool, thus minimizing the possibility of introducing human error in the reworking of intermediate results.	Evidence report for T1 tools.

A.3 Requirements for tools in class T2

Tools in class T2 are in general verification tools.
Some examples of T2 tools:

- static code analysers,
- test coverage monitors,
- theorem proving assistants,
- simulators and,
- model checkers.

Table A2. Requirements for tools in class T2

Requirements	How the evidence shall be provided	Documents to be created
6.7.4.1	Provide the evidence that tools in class T2 cooperate. NOTE Tools cooperate if the outputs from one tool have suitable content and format for automatic input to a subsequent tool, thus minimizing the possibility of introducing human error in the reworking of intermediate results.	Evidence report for T2 tools.
6.7.4.2	Provide a justification report for the selection of tools in class T2. NOTE The justification report shall include the identification of potential failures which can be injected into the tools output and the measures to avoid or handle such failures.	Justification report for T2 tools.
6.7.4.3	Provide a tool manual for each tool in class T2. NOTE A manual which clearly defines the behaviour of the tool and any instructions or constraints on its use.	Tool Manual for each T2 tools.
6.7.4.10	Each tool in class T2 must be subject to configuration management.	A configuration management process for tools in class T2 shall be defined.
6.7.4.11	For each new version of tools in class T2 that is used, provide the evidence that the new version contains no significant new unknown faults.	Evidence report, when a new version of T2 tool class is used.

A.4 Requirements for tools in class T3

Some examples of T3 tools:

- Design refinement tools,
- Compilers,
- Assemblers,
- linkers,
- binders,
- loaders,
- Code generation tools,
- Diagnostic tools used to maintain and monitor the software under operating conditions,
- Application data/algorithm tools.

Table A3. Requirements for tools in class T3

Requirements	How the evidence shall be provided	Documents to be created
6.7.4.1	Provide the evidence that tools in class T3 cooperate. NOTE Tools cooperate if the outputs from one tool have suitable content and format for automatic input to a subsequent tool, thus minimizing the possibility of introducing human error in the reworking of intermediate results.	Evidence report for T3 tools.
6.7.4.2	Provide a justification report for the selection of tools in class T3. NOTE The justification report shall include the identification of potential failures which can be injected into the tools output and the measures to avoid or handle such failures.	Justification report for T3 tools.
6.7.4.3	Provide a tool manual for each tool in class T3. NOTE A manual which clearly defines the behaviour of the tool and any instructions or constraints on its use.	Tool Manual for each T3 tools.
6.7.4.4	Provide the evidence that the tools fulfill the requirements for tools in class T3. According to EN 50128:2011, the evidence may be based on one of the following topic: 1. a suitable combination of history of successful use in similar environments and for similar applications6 (within the organisation or other organisations). 2. a tool validation report as specified in 6.7.4.5. 3. diverse redundant code which allows the detection and control of failures resulting in faults introduced by a tool. 4. Compliance with the safety integrity levels derived from the risk analysis of the process and procedures including the tools. 5. other appropriate methods for avoiding or handling failures introduced by tools	**Recommendation:** **For Tool in class T3, we recommend to develop them according to the requirements described in annex B.**

Table A3. Requirements for tools in class T3

Requirements	How the evidence shall be provided	Documents to be created
6.7.4.5	Provide or create a tool validation report as evidence for the safety integrity of tool in class T3 (refer to 6.7.4.4) . NOTE The validation report as mentioned in 6.7.4.4 shall address the following topic: • a description of the validation activities; • the version of the tool manual being used; • the tool functions being validated; • tools and equipment used; • the results of the validation activity; • test cases and their results for subsequent analysis; • discrepancies between expected and actual results.	Tool validation report.

Appendix B: Development process for tools in class T3

B.1 Introduction

This appendix describes the development process for tools in class T3.
The process described in this appendix is only a recommendation, because the standard CENELEC EN 50128:2011 describes several ways to prove the safety integrity of a T3 tool. The process described in this appendix might be a complex one, but it offers the advantage that the assessment process is structured and less complex.
The process described in this appendix is a mapping of the development process for SIL4 software according to the CENELEC standard EN 50128:2011.

B.2 Boundary conditions for operating systems and hardware (Clause 4 according to EN 50128:2011)

Table B1. Boundary conditions for operating systems and hardware (Clause 4 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
4.1	specify the capabilities and boundary conditions to be fulfilled by the operating system and hardware (if not available).	Boundary conditions for the operating system and hardware.

B.3 Management and organisation (Clause 5 according to EN 50128:2011)

B.3.1 Management and organisation

The objective of this subclause is to ensure that all personnel who have responsibilities within the development of the tools are organised, empowered and capable of fulfilling their responsibilities.

Table B2. Management and organisation regarding the development tools in class T3 (Subclause 5.1 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
5.1.2.1	An EN ISO 9001 certification for each organisation involved in the tool development process is highly recommended	EN ISO 9001 certification or Evidence of a quality management process according to EN ISO 9001.
5.1.2.2, 5.1.2.3 and 5.1.2.9 to 5.1.2.14	Create a quality assurance plan for the tool development process. **NOTE** • The quality assurance plan shall be written according to subclause 6.5. • The quality assurance plan shall also implement the requirements of 5.1.2.2, 5.1.2.3, 5.1.2.9, 5.1.2.10, 5.1.2.13 and 5.1.2.14.	Quality assurance plan for the tool development process
5.1.2.4 to 5.1.2.8	An assessment plan for the tools shall be created.	Assessment Plan for tools

B.3.2 Personnel Competence

The objective of this subclause is to ensure that all personnel who have responsibilities for the development of tools are competent to discharge those responsibilities by demonstrating the ability to perform relevant tasks correctly, efficiently and consistently to a high quality and under varying conditions.

Table B3. Personnel competence regarding tools in class T3 (Subclause 5.2 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
5.2.2.1 and 5.2.2.2	After it has been named and recorded in the quality assurance plan who is: • Requirements Manager, • Designer, • Implementer, • Tester, • Verifier, • Integrator, • Validator, • Project Manager, • Configuration Manager. and what their responsibilities within the tool development process are, now it shall be proved, that these persons are competent to discharge these responsibilities. We propose to create a kind of "openETCS people competence matrix" at the management level. In this document the CV of all persons involved in the project must be recorded. Where key competencies to fulfill a role are miss-ing, trainings must be provided. Participation in trainings shall also be recorded in this document.	openETCS people competence matrix NOTE the created people competence matrix can also be used as evidence for the development process of the openETCS application software
5.2.2.3 and 5.2.2.4	See requirement 5.1.2.1	EN ISO 9001 certification or Evidence of a quality management process according to EN ISO 9001.

B.3.3 Lifecycle issues and documentation

The objective of this subclause is to structure the development of tools into defined phases and activities and to record all informations throughout the lifecycle.

Table B4. Lifecycle issues and documentation regarding tools in class T3 (Subclause 5.3 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
5.3.2.1 and 5.3.2.3	Select a lifecycle model for the development process of tools.	The selected lifecycle model shall be described in the **tool quality assurance plan**.
5.3.2.2	See requirement 5.3.2.14	See requirement 5.3.2.14

Table B4. Lifecycle issues and documentation regarding tools in class T3 (Subclause 5.3 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
5.3.2.4	**The Quality Assurance Plan**, **Verification Plan**, **Validation Plan** and **Configuration Management Plan** shall be drawn up at the start of the project and maintained throughout the development life cycle of the tools	Recommendation
5.3.2.5	• Define all the activities to be performed at each phase of the tool development lifecycle model, • Create planning documents at the project start	Planning documents
5.3.2.6 and 6.5.4.9	Define a document management process for the tool development process. • The process shall be described in a document management plan, • The created document management plan shall implement requirements of 5.3.2.6 and 6.5.4.9.	Document management plan for the tool development process
5.3.2.7	Create an overall notation for documents, create a documents relationship matrix	- Overall notation for documents - Documents relationship matrix for the tool development process. Both documents shall be referenced in the tool quality assurance plan
5.3.2.8 and 5.3.2.10	Create an overall definition of terms and abbreviations to be used for the tool development process. The created "overall definition of terms and abbreviations" shall implement requirements 5.3.2.8 and 5.3.2.10	Overall definition of terms and abbreviations. NOTE The created document shall be referenced in the tool quality assurance plan.
5.3.2.9	Create a checklist to verify the internal consistency of each created documents during the development of the tools. The created checklist shall implement the requirement 5.3.2.9.	Checklist to verify the internal consistency of documents. NOTE The created Checklist shall be referenced in the tool quality assurance plan.
5.3.2.11	Use well-established document file formats (html, ps, pdf, rtf, odf or latex).	The selected file formats shall be recorded in the tool quality assurance plan.
5.3.2.12 and 5.3.2.13	We do not recommend to combine documents created by independent roles.	-

Table B4. Lifecycle issues and documentation regarding tools in class T3 (Subclause 5.3 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
5.3.2.14	Where any alternative lifecycle or documentation structure is adopted, provide the evidence that it meets all the objectives and requirements of the CENELEC EN 50128:2011. NOTE The Evidence may be based on the following topics: • top-down design methods, • modularity, • verification of each phase of the development lifecycle, • verified components and component libraries, • clear documentation and traceability, • auditable documents, • validation, • configuration management and change control and • appropriate consideration of organisation and personnel competency issues.	Proof report, when any alternative lifecycle or documentation structure is adopted.

B.4 Quality assurance measures (Clause 6 according to EN 50128:2011)

B.4.1 Testing activities

The objective of this subclause is to ascertain the behaviour or performance of the developed tool against the corresponding test specification to the extent achievable by the test coverage.

Table B5. Testing activities during the development of tools in class T3 (Subclause 6.1 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
6.1.4.1 to 6.1.4.4 and 8.4.8.6	Created Test Specifications. Each created Test Specification shall implement the requirement of 6.1.4.4 and 8.4.8.6.	Test Specifications for tools
6.1.4.5	Created test report. Each created test report shall implement the requirement of 6.1.4.5.	Test Reports for tools

B.4.2 Verification activities

The objective of this subclause is to examine and arrive at a judgment based on evidence that output items (process, documentation, software) of a specific development phase fulfill the requirements and plans with respect to completeness, correctness and consistency.

Table B6. Verification activities during the development of tools in class T3 (Subclause 6.2 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
6.2.4.1 to 6.2.4.9 and 8.4.8.6	Create a verification plan. The created verification plan shall implement the requirement of 6.2.4.9 and 8.4.8.6. NOTE: For verification activities, appropriate combinations of techniques for software SIL4 as described in annex A, table A.5, A,6, A.7 and A.8 of EN 50128:2011 shall be selected and recorded in the tool verification Plan.	verification plan for tools
6.2.4.10 to 6.2.4.11	verify the created verification plan for tools. • The results of the verification shall be recorded in the quality assurance verification report, • The created quality assurance verification report shall implement the requirements of 6.2.4.11.	Quality assurance verification report for tools.
6.2.4.12 to 6.2.4.13	Create verification reports. Each created verification report shall implement the requirements of 6.2.4.13.	Verification Reports for tools.

B.4.3 Validation activities

The objective of this subclause is to determine whether the developed tool fits the user needs, in particular with respect to safety and quality and with emphasis on the suitability of its operation in accordance to its purpose in its intended environment.

Table B7. Validation activities during the development of tools in class T3 (Subclause 6.3 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
6.3.4.1 to 6.3.4.6 and 8.4.8.6	Create a validation plan. The created validation plan shall implement the requirements of 6.3.4.4 and 8.4.8.6.	Validation Plan for tools
6.3.4.7 to 6.3.4.11	Create a validation report. The created validation report shall implement the requirements from 6.3.4.8 to 6.3.4.11	Validation report for tools.
6.3.4.13	verify the created validation plan for tools. • The results of the verification shall be recorded in the validation verification report, • The created validation verification report shall implement the requirements of 6.3.4.13.	Validation plan verification report for tools.
6.3.4.14	verify the created validation report for tools. • The results of the verification shall also be recorded in the validation verification report, • The created validation verification report shall also implement the requirements of 6.3.4.14.	Validation verification report for tools.

B.4.4 Assessment activities

These activities are not part of the openETCS project and will be carried out by an independent safety assessor.

B.4.5 Quality assurance activities

The objective of this subclause is to identify, monitor and control all those activities, both technical and managerial, which are necessary to ensure that the developed tool achieves the quality required.

Table B8. Quality assurance activities during the development of tools in class T3 (Subclause 6.5 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
6.5.4.1	See requirement 5.3.2.4	See requirement 5.3.2.4
6.5.4.2	See requirement 5.3.2.1	See requirement 5.3.2.1
6.5.4.3 to 6.5.4.6 and 6.5.4.13	Create a quality assurance plan. The created quality assurance plan shall implement the requirements of 6.5.4.5, 6.5.4.6 and 6.5.4.13	Quality assurance plan for tools.
6.5.4.7 to 6.5.4.8	Verify the created quality assurance plan for tools. • The results of the verification shall also be recorded in the quality assurance verification report, • The created quality assurance verification report shall implement the requirements of 6.5.4.8.	Quality assurance verification report for tools.
6.5.4.9	See requirement 5.3.2.6	See requirement 5.3.2.6
6.5.4.10 to 6.5.4.12 and 9.1.4.14	Create a configuration management plan for tools. • The IEEE Standard for Software Configuration Management Plans (IEEE 828) can be used as guideline, • The created configuration management plan shall also implement the requirements 6.5.4.10 to 6.5.4.12 and 9.1.4.14.	Configuration management plan for tools.
6.5.4.14 to 6.5.4.17 and 9.1.4.19	Define a traceability process at documents level and at requirements level. The process shall implement requirements from 6.5.4.14 to 6.5.4.16 and 9.1.4.19.	Traceability plan for tools

B.4.6 Modification and change control activities

The objective of this subclause is to ensure that the tools perform as required, preserving the safety integrity and dependability when modifying the tools.

Table B9. Modification and change control activities during the development of tools in class T3 (Subclause 6.6 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
6.6.4.1 to 6.6.4.2 and 8.4.8.5	Define a change management process. The change management process shall be described in a change management plan and shall implement the requirements from 6.6.4.1 to 6.6.4.2 and 8.4.8.5.	Change management plan for tools

B.4.7 Support tools and languages

This subclause refers to tools that will be used for the development of T3 tools.

Table B10. Support tools and languages used for the development tools in class T3 (Subclause 6.7 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
6.7.4.1	See section A.2	See section A.2
6.7.4.2	See section A.3	See section A.3
6.7.4.3	See section A.3	See section A.3
6.7.4.4 and 6.7.4.5	For tools used to generate the source code of T3 tools, we suggest to provide a validation report as evidence for the safety integrity of these tools. NOTE 1 The validation report shall address the following topic: • a description of the validation activities; • the version of the tool manual being used; • the tool functions being validated; • tools and equipment used; • the results of the validation activity; • test cases and their results for subsequent analysis; • discrepancies between expected and actual results. NOTE 2 The evidence may also be based on one of the following topic: 1. a suitable combination of history of successful use in similar environments and for similar applications6 (within the organisation or other organisations). 2. diverse redundant code which allows the detection and control of failures resulting in faults introduced by a tool. 3. Compliance with the safety integrity levels derived from the risk analysis of the process and procedures including the tools. 4. other appropriate methods for avoiding or handling failures introduced by tools	Tool validation report.
6.7.4.6 to 6.7.4.11	We suggest to implement only requirements 6.7.4.1 to 6.7.4.5.	-

B.5 Tools development phases (Clause 7 according to EN 50128:2011)

B.5.1 Lifecycle and documentation

See Subsection B.3.3.

B.5.2 Requirements phase

The objective of this subclause is to describe a complete set of requirements for tools to be developed and to describe the overall test specification.

Table B11. Requirements for tools in class T3 (Subclause 7.2 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
7.2.4.1 to 7.2.4.14 and 8.4.8.2	Create a requirements specification for the tools to be developed. • The requirements specification shall implement requirements from 7.2.4.2 to 7.2.4.15 and 8.4.8.2, • Requirements 7.2.4.10 and 7.2.4.11 are not relevant for tools. NOTE: The traceability of requirements at this stage back to all input documents is mandatory. • System Requirements Specification, • System Safety Requirements Specification, • System Architecture Description, • External Interface Specifications (e.g. Software/Software Interface Specification, Software/Hardware Interface Specification).	- Requirements specification for tools. - Traceability proof
7.2.4.15	Select techniques from Table A.2 of EN 50128:2011 to specify requirements. The selected techniques shall be suitable for software SIL4.	Record the selected techniques in the tool quality assurance plan.
7.2.4.16, 7.2.4.17 and 7.2.4.19	Create an overall Test Specification for testing the tools to be developed, (refer to 6.1.4.4). The created overall test specification shall also implement the requirements of 7.2.4.19.	Overall Test Specification for the tools
7.2.4.18	Select techniques from Table A.7 of EN 50128:201 to specify the overall test specification. The selected techniques shall be suitable for software SIL4.	The selected techniques shall be recorded in the quality assurance plan of tools.
7.2.4.21 and 7.2.4.22	Verify the created requirement specification. • The results of the verification shall be recorded in the requirements verification report, • The created requirements verification report shall implement the requirements of 7.2.4.22.	Requirements verification report for tools

B.5.3 Architecture and design phase

The objective of this subclause is to develop an abstract architecture of the tools to be developed, that achieves the requirements described in B.5.2 and to identify and evaluate the significance of the interactions of the tools with the hardware.

Table B12. Architecture and design for tools in class T3 (Subclause 7.3 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
7.3.4.1 to 7.3.4.15	Create an architecture specification for the tools to be developed. • The architecture specification shall implement requirements from 7.3.4.1 to 7.3.4.15, • Requirement 7.3.4.4 is not relevant for tools. NOTE: The traceability of requirements at this stage back to B.5.2 is mandatory.	- Architecture specification for tools. - Traceability proof
7.3.4.14	Select techniques from Table A.3 of EN 50128:2011 to specify the architecture The selected techniques shall be suitable for software SIL4	Record the selected techniques in the tool quality assurance plan. **NOTE** The technic "Error Detecting Codes" is mandatory.
7.3.4.16 and 7.3.4.17	Prototyping can be used at this stage.	Provide the evidence that the development process of the prototype and its documentation fulfill SIL4 according to EN 50128:2011
7.3.4.18 and 7.3.4.19 and 8.4.8.3	Create an interface specification for the tools to be developed. The Interface specification shall implement requirements from 7.3.4.18 and 7.3.4.19 and 8.4.8.3.	Interface specification for tools.
7.3.4.20 to 7.3.4.23, 7.3.4.28, 9.1.4.10, 9.1.4.13, 9.1.4.16 and 9.1.4.20	Create a design specification for the tools to be developed. The design specification shall implement requirements from 7.3.4.21 to 7.3.4.24, 7.3.4.28, 9.1.4.10, 9.1.4.13, 9.1.4.16 and 9.1.4.20.	Design specification for tools.
7.3.4.24	Select techniques from Table A.4 of EN 50128:2011 to specify the design of tools to be developed. The selected techniques shall be suitable for the software SIL4	Record the selected techniques in the tool quality assurance plan.
7.3.4.25 to 7.3.4.27	Specify a Coding standard. The specified Coding standard should include: • language justification, • Scope and base standard when available, (NOTE For domain specific languages base standards may not be available.), • procedure for changing the coding standard, • analysis of the potential faults and recommended treatment, • restrictions to avoid the faults, • portability. **NOTE:** A Justification shall also be provided that the coding standards used satisfy the software SIL4.	- Coding standard, - Justification for the coding standards used. **NOTE** The coding standards shall be referenced in the tool quality assurance plan.
7.3.4.29 and 7.3.4.31	Create an integration Test Specification for tools(refer to 6.1.4.4). The integration test specification shall also implement the requirements of 7.3.4.31	Integration test specification for tools.
7.3.4.32	Select techniques from Tables A.5 and A.6 of EN 50128:2011 to specify integration tests. The selected techniques shall be suitable for the software SIL4	Record the selected techniques in the tool quality assurance plan.

Table B12. Architecture and design for tools in class T3 (Subclause 7.3 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
7.3.4.33 to 7.3.4.38	Create a hardware integration test specification for the tools (refer to 6.1.4.4) The created Hardware integration test specification shall also implement the requirements from 7.3.4.34 to 7.3.4.39	Hardware integration test specification for tools
7.3.4.39	See requirement 7.3.4.32	See requirement 7.3.4.32.
7.3.4.40 to 7.3.4.43	Verify the created architecture Specification for tools, the Design Specification for tools and all the test specifications. • The results of the verification shall be recorded in a architecture and design verification report for tools, • The created architecture and design verification report for tools shall implement requirements from 7.3.4.40 to 7.3.4.43.	Architecture and design verification report for tools

B.5.4 Components design phase

The objective of this subclause is to develop an detailed architecture of the tools to be developed, that achieves the requirements described in B.5.3.

Table B13. Components design for tools in class T3 (Subclause 7.4 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
7.4.4.1 to 7.4.4.5	Create a component design Specification for the tools to be developed. The created component design specification shall implement requirements from 7.4.4.1 to 7.4.4.6	Components design specification for tools
7.4.4.6	See requirement 7.3.4.24	See requirement 7.3.4.24
7.4.4.7 to 7.4.4.9	Create a component test specification for the tools (refer to 6.1.4.4). The created tools component test Specification shall also implement requirements from 7.4.4.8 to 7.4.4.10	Components test Specification for tools
7.4.4.10	See requirement 7.3.4.32	See requirement 7.3.4.32.
7.4.4.11 to 7.4.4.13	Verify the created components design specification and the components test specification for the tools. • The results of the verification shall be recorded in a component design verification report, • The created component design verification report shall implement requirements from 7.4.4.11 to 7.3.4.13.	Component design verification report for tools

B.5.5 Implementation and Testing phase

The objective of this subclause is to provide a source code of the developed tool, which is analysable, testable, verifiable and maintainable.

Table B14. Implementation and tests of tools in class T3 (Subclause 7.5 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
7.5.4.1 to 7.5.4.4	Provide the source code of tools. The source code shall compliant with the requirements of 7.5.4.1 and 7.5.4.2.	Software source code of tools.
7.5.4.5 to 7.5.4.7	Perform components tests. • The result of all component tests shall be recorded in a component test report for tools, • The component test report for tools shall be written according to requirement 6.1.4.5, • The created component test report shall also address requirements from 7.5.4.6 to 7.5.4.7.	Component test report for tools.
7.5.4.8 to 7.5.4.10	Verify the source code of tools. • The results of the verification shall be recorded in a component design verification report for tools, • The created component design verification report for tools shall implement requirements from 7.4.4.11 to 7.3.4.13.	Component design verification report for tools.

B.5.6 Integration phase

The objective of this subclause is to carry out the component- and hardware integration, in order to demonstrate that the developed tools perform their intended functions and also interact correctly with the hardware.

Table B15. Integration of the source code of the developed tools (Subclause 7.6 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
7.6.4.3 to 7.6.4.5	Perform component integration tests of the developed tools. • The result of the component integration tests shall be recorded in a integration test report for tools, • The integration test report for tools shall be written according to 6.1.4.5. • The created integration test report for tools shall also implement the requirements of 7.6.4.5	Integration test report for tools.
7.6.4.6	See requirement 7.3.4.32	See requirement 7.3.4.32.
7.6.4.7 to 7.6.4.9	Perform hardware integration tests. • The result of the hardware integration test shall be recorded in a hardware integration test report for tools, • The hardware integration test report for tools shall be written according to 6.1.4.5, • The created hardware integration test report for tools shall also implement the requirements of 7.6.4.9.	Hardware integration test report for tools.
7.6.4.10	See requirement 7.3.4.32	See requirement 7.3.4.32.

Table B15. Integration of the source code of the developed tools (Subclause 7.6 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
7.6.4.11 to 7.6.4.13	Verify the created Integration test report for tools and Hardware integration test report for tools. • The results of the verification shall be recorded in a integration verification report for tools, • The created integration verification report for tools shall implement requirements from 7.6.4.11 to 7.6.4.13.	Integration verification report for tools.

B.5.7 Validation phase

The objective of this subclause is to analyse and test the integrated source code of the developed tools and hardware to ensure compliance with the requirements specification B.5.2 with particular emphasis on the functional and safety aspects according to software SIL4 and to check whether it is fit for its intended application.

Table B16. Validation of the source code of the developed tools (Subclause 7.7 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
7.7.4.1 to 7.7.4.4	Perform the overall test of the source code of the developed tool. • The result of the overall test shall be recorded in the overall test report of the tool, • The overall test of the tool shall be written according to requirement 6.1.4.5.	Overall test report of the tool.
7.7.4.6 to 7.7.4.11	Perform a validation of the developed tool. • The result of the validation shall be recorded in the tool validation report, • The tool validation report shall be written according to requirements 7.7.4.6 to 7.7.4.11.	Tool validation report.
7.7.4.12, 8.4.8.8, 9.1.4.4 and 9.1.4.5	A Release Note which accompanies the delivered tool shall be created. The release note shall be written according to requirements 7.7.4.12, 8.4.8.8, 9.1.4.4 and 9.1.4.5	Tool release note.

B.6 Development of application data or algorithms (Clause 8 according to EN 50128:2011)

Tools to configurea system are T3 tools and shall be developed according to the requirements described in clauses B.3, B.4, B.5 and B.7.

B.7 Deployement and maintenance (Clause 9 according to EN 50128:2011)

B.7.1 Deployment of the developed tools

The objective of this subclause is to ensure that the tool performs as required, preserving the required safety integrity and dependability when it is deployed in the final environment of application.

Table B17. Deployment of the developed tools (Subclause 9.1 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
9.1.4.1 and 9.1.4.5	Define a tool deployment process. The tool deployment process shall implement the requirements 9.1.4.4, 9.1.4.5 and 9.1.4.12 to 9.1.4.20.	Tool deployment manual.
9.1.4.6 to 9.1.4.9	Verify the create tool deployment manual • The results of the verification shall be recorded in the tool deployment verification report, • The created tool deployment verification report shall implement requirements from 9.1.4.6 to 9.1.4.9.	Tool deployment verification report.

B.7.2 Maintenance of the developed tools

The objective of this subclause is to ensure that the tools performs as required, preserving the required safety integrity and dependability when making corrections, enhancements or adaptations to the tools itself.

Table B18. Maintenance of the developed tools (Subclause 9.2 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
9.2.4.1 to 9.2.4.6	Define a tool maintenance process The tool maintenance process shall implement the requirement 9.2.4.6, 9.2.4.12 to 9.2.4.19.	Tool maintenance Plan.
9.2.4.7 to 9.2.4.10	Verify the created Tool maintenance Plan. • The results of the verification shall be recorded in the tool maintenance verification report, • The created tool maintenance verification report shall implement requirements from 9.2.4.7 to 9.2.4.10.	Tool Maintenance verification report.
9.2.4.12	Select techniques from the Table A.10 of EN 50128:2011 for maintenance activities. The selected techniques shall be suitable for the software SIL4	Record the selected techniques in the tool maintenance plan.

Appendix C:
Software development: state of the art regarding CENELEC EN 50128:2011

C.1 Introduction

This section gathers requirements, which shall be fulfilled by the openETCS application software (in this annex also named openETCS software). Regarding the openETCS application software, it was assumed during the creation of this document, that the whole or part of the software will be SIL4 compliant according to EN 50128:2011. Therefore, the sections below address requirements for SIL 4 Software according to EN 50128:2011.

C.1.1 How to read the tables used in this appendix

Tables used in this appendix consist of three columns.

- **Column 1** lists the requirements comming from the CENELEC standard EN 50128:2011. But since the standard is protected by copyright, only the requirements identification numbers are recorded there. The same requirements identification numbers (subclause) as defined in the EN 50128:2011 have been used. The textual requirement should therefore be read in the standard using the requirements identification number recorded in column 1 as reference.
  In order to simplify the readability, some of the requirements have been grouped, especially when they address the same subject; only few were left out, because they were not relevant for the purpose.
- **column 2** describes how the various requirements can be fulfilled. Everywhere where possible, recommendations were formulated.
- **column 3** gives a suggestion of documents that shall be created or provided, in order to provide the evidence.

C.2 Software Safety Integrity Levels (Clause 4 according to EN 50128:2011)

Table C1. Safety Integrity Levels (Clause 4 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
4.1	Requirement (4.1) describes the allocation of software safety integrity levels to the openETCS application software in accordance with the CENELEC standards EN 50126 and EN 50129. NOTE: A software can not be assessed without specifying the capabilities and boundary conditions to be fulfilled by the operating system and hardware. Regarding the openETCS application software, we therefore recommend to perform a safety analysis, in order to specify the boundary conditions for both: operating system and hardware.	Boundary conditions for the operating system and hardware.

C.3 Management and organisation (Clause 5 according to EN 50128:2011)

C.3.1 Management and organisation

The objective of this subclause is to ensure that all personnel who have responsibilities within the development of the openETCS software are organised, empowered and capable of fulfilling their responsibilities.

Table C2. Management and organisation regarding the development process of the openETCS software

Requirements	How the evidence shall be provided	Documents to be created
5.1.2.1	EN ISO 9001 certification for each organisation involved in the development process of the openETCS software is highly recommended.	EN ISO 9001 certification or Evidence of a quality management process according to EN ISO 9001.
5.1.2.2,5.1.2.3, 5.1.2.9 to 5.1.2.14	Create a software quality assurance plan for the development process of the openETCS software. • The software quality assurance plan shall be written according to subclause 6.5, • The created software quality assurance plan shall also implement the requirements of 5.1.2.2, 5.1.2.3, 5.1.2.9, 5.1.2.10, 5.1.2.13 and 5.1.2.14.	software quality assurance plan for the openETCS software.
5.1.2.4 to 5.1.2.8	Create an assessment plan for the openETCS software. (only when required).	openETCS software assessment plan for the openETCS software

C.3.2 Personnel Competence

The objective of this subclause is to ensure that all personnel who have responsibilities within the development process of the openETCS software are competent to discharge those responsibilities by demonstrating the ability to perform relevant tasks correctly, efficiently and consistently to a high quality and under varying conditions.

Table C3. Personnel competence

Requirements	How the evidence shall be provided	Documents to be created
5.2.2.1 and 5.2.2.2	After it has been named and recorded in the software quality assurance plan who is: • Requirements Manager, • Designer, • Implementer, • Tester, • Verifier, • Integrator, • Validator, • Project Manager, • Configuration Manager. and what their responsibilities within the development process of the openETCS application software are, now it shall be proved, that these persons are competent to discharge these responsibilities. We propose to create a kind of "openETCS people competence matrix" at the management level. In this document the CV of all persons involved in the project must be recorded. Where key competencies to fulfill a role are missing, trainings must be provided. Participation in trainings shall also be recorded in this document.	openETCS people competence matrix. NOTE The openETCS people competence matrix created in annex A, can also be used here as evidence.
5.2.2.3 and 5.2.2.4	See requirement 5.1.2.1	EN ISO 9001 certification or Evidence of a quality management process according to EN ISO 9001

C.3.3 Lifecycle issues and documentation

The objective of this subclause is to structure the development process of the openETCS software into defined phases and activities and to record all information pertinent to the software throughout the lifecycle of the software.

Table C4. Lifecycle issues and documentation regarding the development of the openETCS software)

Requirements	How the evidence shall be provided	Documents to be created
5.3.2.1 and 5.3.2.3	A lifecycle model for the development of the openETCS software shall be selected.	The selected lifeycle model shall be described in the quality assurance plan of the openETCS software.
5.3.2.2	See requirement 5.3.2.14	See requirement 5.3.2.14
5.3.2.4	**The Quality Assurance Plan**, **Verification Plan**, **Validation Plan** and **Configuration Management Plan** of the openETCS software shall be drawn up at the start of the project and maintained throughout the development life cycle of the openETCS software	Recommendation
5.3.2.5	• Define all the activities to be performed at each phase of the openETCS software development lifeycle model, • Create planning documents at the project start	Planning documents

Table C4. Lifecycle issues and documentation regarding the development of the openETCS software)

Requirements	How the evidence shall be provided	Documents to be created
5.3.2.6 and 6.5.4.9	Define a document management for the openETCS software development process. • The process shall be described in a document management plan, • The created document management plan shall implement requirements of 5.3.2.6 and 6.5.4.9.	Document management plan for the openETCS software development process.
5.3.2.7	Create a overall notation for documents, Create a documents relationship matrix.	- Overall notation for documents - Documents relationship matrix, Both documents shall be referenced in the software quality assurance plan of the openETCS software
5.3.2.8 and 5.3.2.10	Create an overall definition of terms and abbreviations to be used within the openETCS software development process. The created "overall definition of terms and abbreviations" shall implement requirements 5.3.2.8 and 5.3.2.10	Overall definition of terms and abbreviations. NOTE The created document shall be referenced in the software quality assurance plan of the openETCS software.
5.3.2.9	Create a checklist to verify the internal consistency of each created documents during the development of the openETCS software. The created checklist shall implement the requirement of 5.3.2.9.	Checklist to verify the internal consistency of documents. NOTE The created Checklist shall be referenced in the software quality assurance plan of the openETCS software.
5.3.2.11	Use well-established document file formats (html, ps, pdf, rtf, odf or latex).	The selected file formats shall be recorded in the quality assurance plan of the openETCS software. **NOTE** the openETCS tools shall generate documents in the selected file formats
5.3.2.12 and 5.3.2.13	We do not recommend to combine documents created by independent roles.	-

Table C4. Lifecycle issues and documentation regarding the development of the openETCS software)

Requirements	How the evidence shall be provided	Documents to be created
5.3.2.14	Where any alternative lifecycle or documentation structure is adopted, provide the evidence that it meets all the objectives and requirements of the CENELEC EN 50128:2011. NOTE The Evidence may be based on the following topic: • top-down design methods, • modularity, • verification of each phase of the development lifecycle, • verified components and component libraries, • clear documentation and traceability, • auditable documents, • validation, • configuration management and change control and • appropriate consideration of organisation and personnel competency issues.	Proof report, when any alternative lifecycle or documentation structure is adopted.

C.4 Quality assurance measures (Clause 6 according to EN 50128:2011)

C.4.1 Testing activities

The objective of this subclause is to ascertain the behaviour or performance of the developed openETCS software against the corresponding test specification to the extent achievable by the test coverage.

Table C5. Testing activities during the development of the openETCS software (Subclause 6.1 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
6.1.4.1 to 6.1.4.4 and 8.4.8.6	Created Test Specifications. Each created Test Specification shall implement the requirement of 6.1.4.4 and 8.4.8.6.	Test Specifications for the openETCS software
6.1.4.5	Created test report. Each created test report shall implement the requirement of 6.1.4.5.	Test Reports for the openETCS softwar

C.4.2 Verification activities

The objective of this subclause is to examine and arrive at a judgment based on evidence that output items (process, documentation, software) of a specific development phase fulfill the requirements and plans with respect to completeness, correctness and consistency.

Table C6. Verification activities during the development of the openETCS software (Subclause 6.2 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
6.2.4.1 to 6.2.4.9 and 8.4.8.6	Create a verification plan. The created verification plan shall implement the requirement of 6.2.4.9 and 8.4.8.6. NOTE: For verification activities, appropriate combinations of techniques for software SIL4 as described in annex A, table A.5, A,6, A.7 and A.8 of EN 50128:2011 shall be selected and recorded in the tool verification Plan.	Software verification Plan for the openETCS software
6.2.4.10 to 6.2.4.11	verify the created verification plan for the openETCS software. • The results of the verification shall be recorded in the software quality assurance verification report, • The created software quality assurance verification report shall implement the requirements of 6.2.4.11.	Software quality assurance verification report for the openETCS software.
6.2.4.12 to 6.2.4.13	Create software verification reports. Each created software verification report shall implement the requirements of 6.2.4.13.	Software verification Reports for the openETCS software.

C.4.3 Validation activities

The objective of this subclause is to determine whether the developed openETCS software fits the user needs, in particular with respect to safety and quality and with emphasis on the suitability of its operation in accordance to its purpose in its intended environment.

Table C7. Validation activities during the development of the openETCS software (Subclause 6.3 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
6.3.4.1 to 6.3.4.6 and 8.4.8.6	Create a Software validation plan. The created software validation plan shall implement the requirements of 6.3.4.4 and 8.4.8.6.	Software Validation Plan for the openETCS software
6.3.4.7 to 6.3.4.11	Create a software validation report. The created software validation report shall implement the requirements from 6.3.4.8 to 6.3.4.11	Software validation report for the openETCS software.
6.3.4.13	verify the created validation plan for the openETCS software. • The results of the verification shall be recorded in the software validation verification report, • The created software validation verification report shall implement the requirements of 6.3.4.13.	Software validation plan verification report for the openETCS software.
6.3.4.14	verify the created validation report for the openETCS software. • The results of the verification shall also be recorded in the software validation verification report, • The created software validation verification report shall also implement the requirements of 6.3.4.14.	Software Validation verification report for the openETCS software.

C.4.4 Assessment activities

These activities are not part of the openETCS project and will be carried out by an independent safety assessor.

C.4.5 Quality assurance activities

The objective of this subclause is to identify, monitor and control all those activities, both technical and managerial, which are necessary to ensure that the developed openETCS software achieves the quality required.

Table C8. Quality assurance activities during the development of the openETCS software (Subclause 6.5 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
6.5.4.1	See requirement 5.3.2.4	See requirement 5.3.2.4
6.5.4.2	See requirement 5.3.2.1	See requirement 5.3.2.1
6.5.4.3 to 6.5.4.6 and 6.5.4.13	Create a software quality assurance plan. The created quality assurance plan shall implement the requirements of 6.5.4.5, 6.5.4.6 and 6.5.4.13	Software quality assurance plan for the openETCS software.
6.5.4.7 to 6.5.4.8	Verify the created software quality assurance plan for the openETCS software. • The results of the verification shall also be recorded in the software quality assurance verification report, • The created software quality assurance verification report shall implement the requirements of 6.5.4.8.	Software quality assurance verification report for the openETCS software.
6.5.4.9	See requirement 5.3.2.6	See requirement 5.3.2.6
6.5.4.10 to 6.5.4.12 and 9.1.4.14	Create a software configuration management plan for the openETCS software. • The IEEE Standard for Software Configuration Management Plans (IEEE 828) can be used as guideline, • The created software configuration management plan shall also implement the requirements of 6.5.4.10, 6.5.4.11, 6.5.4.12 and 9.1.4.14.	Software configuration management plan for the openETCS software.
6.5.4.14 to 6.5.4.17 and 9.1.4.19	Define a traceability process at documents level and at requirements level. The process shall implement requirements from 6.5.4.14 to 6.5.4.16 and 9.1.4.19.	Traceability plan for the openETCS software.

C.4.6 Modification and change control activities

The objective of this subclause is to ensure that the openETCS software performs as required, preserving the software safety integrity and dependability when modifying the software.

Table C9. Modification and change control activities during the development of the openETCS software (Subclause 6.6 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
6.6.4.1, 6.6.4.2 and 8.4.8.5	Define a change management process. The change management process shall be described in a software change management plan and shall implement the requirements from 6.6.4.1 to 6.6.4.2 and 8.4.8.5.	Software change Management Plan for the openETCS software

C.4.7 Support tools and languages for the development of the openETCS software

The objective of this subclause is to provide evidence that potential failures of tools do not adversely affect the integrated toolset output in a safety related manner that is undetected by technical and/or organisational measures outside the tool.

Table C10. Support tools and languages used for the development of the openETCS software (Subclause 6.7 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
6.7.4.1 to 6.7.4.11	Use the openETCS tools chain	openETCS tools chain documentation

C.5 Generic openETCS Software development (Clause 7 according to EN 50128:2011)

C.5.1 Lifecycle and documentation regarding the development of the openETCS software

See Subsection C.3.

C.5.2 openETCS Software requirements

The objective of this subclause is to describe a complete set of requirements for the openETCS software meeting all safety requirements and provides a comprehensive set of documents for each subsequent phase and to describe the overall test specification of the openETCS software.

Table C11. Requirements for the openETCS (Subclause 7.2 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
7.2.4.1 to 7.2.4.14 and 8.4.8.2	Create a requirements specification for the openETCS software. • The created requirements specification shall implement requirements from 7.2.4.2 to 7.2.4.15 and 8.4.8.2, NOTE: The traceability of requirements at this stage back to all input documents is mandatory. (refer to 6.5.4.15)	- Requirements specification of the openETCS software. - Traceability proof
7.2.4.15	Select techniques from Table A.2 of EN 50128:2011 to specify requirements. The selected techniques shall be suitable for software SIL4.	Record the selected techniques in the software quality assurance plan of the openETCS software.

Table C11. Requirements for the openETCS (Subclause 7.2 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
7.2.4.16, 7.2.4.17 and 7.2.4.19	Create an overall Test Specification for testing the openETCS software (refer to 6.1.4.4) The created overall test specification shall also implement the requirements of 7.2.4.19.	Overall Test Specification of the openETCS software
7.2.4.18	Select techniques from Table A.7 of EN 50128:201 to specify the overall test specification. The selected techniques shall be suitable for software SIL4.	Record the selected techniques in the software quality assurance plan of the openETCS software.
7.2.4.21 and 7.2.4.22	Verify the created Requirement Specification for the openETCS software. • The results of the verification shall be recorded in the software requirements verification report, • The created software requirements verification report shall implement the requirements of 7.2.4.22.	software requirements verification report of the openETCS software.

C.5.3 openETCS Software Architecture and Design

The objective of this subclause is to develop an architecture of the openETCS software that achieves the requirements of the software and to identify and evaluate the significance of the interactions of the openETCS software with the hardware.

Table C12. Architecture and design for the openETCS software (Subclause 7.3 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
7.3.4.1 to 7.3.4.15	Create an architecture specification for the openETCS software. • The created architecture specification shall imple-ment requirements from 7.3.4.2 to 7.3.4.15. NOTE: The traceability of requirements at this stage back to C.5.2 is mandatory. (refer to 6.5.4.15)	- Architecture specification of the openETCS software. - Traceability proof
7.3.4.14	Select techniques from Table A.3 of EN 50128:2011 to specify the architecture. The selected techniques shall be suitable for software SIL4.	Record the selected techniques in the software quality assurance plan of the openETCS software. NOTE The technic "Error Detecting Codes" is mandatory.
7.3.4.16 and 7.3.4.17	Prototyping can be used at this stage.	Provide the evidence that the development process of the prototype and its documentation fulfill SIL4 according to EN 50128:2011
7.3.4.18, 7.3.4.19 and 8.4.8.3	Create an interface specification for the openETCS software. The created software interface specification shall implement the requirements of 7.3.4.18, 7.3.4.19 and 8.4.8.3.	Software interface specification of the openETCS software.

Table C12. Architecture and design for the openETCS software (Subclause 7.3 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
7.3.4.20 to 7.3.4.23, 7.3.4.28, 8.4.8.7, 9.1.4.10, 9.1.4.13, 9.1.4.16 and 9.1.4.20	Create a design specification for the openETCS software. The created design specification shall implement requirements from 7.3.4.20 to 7.3.4.24, 7.3.4.28, 8.4.8.7, 9.1.4.10, 9.1.4.13, 9.1.4.16 and 9.1.4.20.	Software design specification of the openETCS software.
7.3.4.24	Select techniques from Table A.4 of EN 50128:2011 to specify the design of the openETCS software. The selected techniques shall be suitable for the software SIL4.	Record the selected techniques in the software quality assurance plan of the openETCS software.
7.3.4.25 to 7.3.4.27	Specify a Coding standard. The specified Coding standard should include: • language justification, • Scope and base standard when available, (NOTE For domain specific languages base standards may not be available.), • procedure for changing the coding standard, • analysis of the potential faults and recommended treatment, • restrictions to avoid the faults, • portability. **NOTE:** A Justification shall be provided that the coding standards used, satisfy the software SIL4.	- Coding standard specification, - Justification for the coding standards used. **NOTE:** The coding standards shall be referenced in the software quality assurance plan of the openETCS software.
7.3.4.29 and 7.3.4.31	Create an integration Test Specification for the openETCS software, according to requirement 6.1.4.4. The created openETCS software integration test specification shall also implement the requirements of 7.3.4.31	Integration test specification of the openETCS software.
7.3.4.32	Select techniques from Tables A.5 and A.6 of EN 50128:2011 to specify integration tests. The selected techniques shall be suitable for the software SIL4	Record the selected techniques in the software quality assurance plan of the openETCS software.
7.3.4.33 to 7.3.4.38	Create a software/hardware integration test specification for the openETCS software, according to requirement 6.1.4.4. The created openETCS software/Hardware integration test specification shall also implement the requirements from 7.3.4.34 to 7.3.4.39	Software/hardware integration test specification of the openETCS software.
7.3.4.39	See requirement 7.3.4.32	See requirement 7.3.4.32.
7.3.4.40 to 7.3.4.43	Verify the created the created architecture Specification-, Design Specification-, software- and software/hardware integration test specification of the openETCS software. • The results of the verification shall be recorded in the software architecture and design verification report, • The created openETCS software architecture and design verification report shall implement requirements from 7.3.4.40 to 7.3.4.43.	openETCS software architecture and design verification report of the openETCS software.

C.5.4 openETCS software Component design

The objective of this subclause is to develop and design software component that achieves the requirements of the openETCS software design specification to the extent required by the software safety integrity level.

Table C13. Components design for the openETCS software (Subclause 7.4 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
7.4.4.1 to 7.4.4.5	Create a component design Specification for the openETCS software. The created component design specification for the openETCS software shall implement requirements from 7.4.4.1 to 7.4.4.6	Components design specification of the openETCS software
7.4.4.6	See requirement 7.3.4.24	See requirement 7.3.4.24
7.4.4.7 to 7.4.4.9	Create a component test specification for the openETCS software, according to requirement 6.1.4.4. The created openETCS software component test Specification shall also implement requirements from 7.4.4.8 to 7.4.4.10	Components test Specification for the openETCS software
7.4.4.10	See requirement 7.3.4.32	See requirement 7.3.4.32.
7.4.4.11 to 7.4.4.13	Verify the created components design specification and the components test specification of the openETCS software. • The results of the verification shall be recorded in the openETCS software component design verification report, • The created openETCS software component design verification report shall implement requirements from 7.4.4.11 to 7.3.4.13.	openETCS software component design verification report of the openETCS software.

C.5.5 openETCS Software implementation and testing

The objective of this subclause is to achieve software which is analysable, testable, verifiable and maintainable.

Table C14. Implementation and tests of the openETCS software (Subclause 7.5 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
7.5.4.1 to 7.5.4.4	Provide the openETCS software source code. The source code shall be compliant with the requirements of 7.5.4.1 and 7.5.4.2.	Software source code of the openETCS software.
7.5.4.5 to 7.5.4.7	Perform component tests of the openETCS software. • The result of all openETCS software component tests shall be recorded in the openETCS software component test report, • The openETCS software component test report shall be written according to requirement 6.1.4.5, • The created openETCS software component test report shall also implement requirements from 7.5.4.6 to 7.5.4.7.	Software component test report of the openETCS software.
7.5.4.8 to 7.5.4.10	Verify the source code of the openETCS software. • The results of the verification shall be recorded in the openETCS software component design verification report, • The created openETCS software component design verification report shall implement requirements from 7.4.4.11 to 7.3.4.13.	Software component design verification report of the openETCS software.

C.5.6 openETCS Software Integration

The objective of the openETCS software integration is to carry out openETCS software and software/hardware integration and demonstrate that the openETCS software and the hardware interact correctly to perform their intended functions.

Table C15. Integration of the openETCS software source code (Subclause 7.6 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
7.6.4.3 to 7.6.4.5	Perform integration tests of component of the openETCS software. • The result of the openETCS software integration test shall be recorded in the openETCS software integration test report, • The openETCS software integration test report shall be written according to requirement 6.1.4.5, • The created openETCS software integration test report shall also implement the requirements of 7.6.4.5.	Software integration test report of the openETCS software.
7.6.4.6	See requirement 7.3.4.32	See requirement 7.3.4.32.
7.6.4.7 to 7.6.4.9	Perform software/hardware integration tests of component of the openETCS software. • The result of the openETCS software/hardware integration test shall be recorded in the openETCS software/hardware integration test report, • The openETCS software/hardware integration test report shall be written according to requirement 6.1.4.5, • The created openETCS software/hardware integration test report shall also implement the requirements of 7.6.4.9.	Software/hardware integration test report of the openETCS software.
7.6.4.10	See requirement 7.3.4.32	See requirement 7.3.4.32.
7.6.4.11 to 7.6.4.13	Verify the created Software integration test report and Software/hardware integration test report of the openTCS software. • The results of the verification shall be recorded in the openETCS software integration verification report, • The created openETCS software integration verification report shall implement requirements from 7.6.4.11 to 7.6.4.13.	Software integration verification report of the openETCS software.

C.5.7 openETCS Software Validation

The objective of the software validation is to analyse and test the integrated openETCS software and hardware to ensure compliance with the software requirements specification with particular emphasis on the functional and safety aspects according to software SIL4 and to check whether it is fit for its intended application.

Table C16. Validation of the openETCS software source code (Subclause 7.7 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
7.7.4.1 to 7.7.4.4	Perform the overall test of the openETCS software. • The result of the overall openETCS software test shall be recorded in the overall openETCS software test report, • The openETCS software test report. shall be written according to requirement 6.1.4.5.	Overall openETCS software test report of the openETCS software.
7.7.4.6 to 7.7.4.11	Perform a validation of the openETCS software. • The result of the openETCS software validation shall be recorded in the openETCS software validation report, • The openETCS software validation report shall be written according to requirements 7.7.4.6 to 7.7.4.11.	Software validation report of the openETCS software.
7.7.4.12, 8.4.8.8, 9.1.4.4 and 9.1.4.5	A Release Note which accompanies the delivered openETCS software shall be created. The release note shall be written according to the requirements of 7.7.4.12, 8.4.8.8, 9.1.4.4 and 9.1.4.5.	Release note of the openETCS software.

C.6 Development of application data or algorithms (Clause 8 according to EN 50128:2011)

C.6.1 Application Data Development Process

Table C17. Development process of application data of the generic openETCS software (Subclause 8.4.1 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
8.4.1.2 to 8.4.1.11 and 8.4.4.1	Created an application preparation plan for the application data of the openETCS application software. The created application preparation plan shall implement requirements from 8.4.1.2 to 8.4.1.11 and 8.4.4.1	openETCS Application data preparation plan.
8.4.1.12 to 8.4.1.13	Verify the created openETCS Application data preparation plan. • The results of the verification shall be recorded in the openETCS application data verification report, • The created application data verification report shall implement the requirement of 8.4.1.13.	openETCS application data verification report.

C.6.2 Application Data Requirements Specification

Table C18. Requirements Specification of application data of the generic openETCS software (Subclause 8.4.2 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
8.4.2.1 to 8.4.2.3	Create a requirement specification for the application data of the openETCS software. The created openETCS application data requirements specification shall implement requirements from 8.4.2.1 to 8.4.2.3.	openETCS application data requirements specification.

Table C18. Requirements Specification of application data of the generic openETCS software (Subclause 8.4.2 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
8.4.2.4 and 8.4.2.5	Verify the created openETCS application data requirements specification. • The results of the verification shall be recorded in the openETCS application data verification report, • The created openETCS application data verification report shall im-plement the requirements of 8.4.2.5.	openETCS application data verification report.

C.6.3 Application Data Architecture and Design

Table C19. Architecture and design of application data of the generic openETCS software (Subclause 8.4.3 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
8.4.3.1	Create an architecture and Design specification for the application data of the openETCS application software.	openETCS application data architecture and design specification.

C.6.4 Application Data Production

Table C20. Production of application data of the generic openETCS software (Subclause 8.4.4 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
8.4.4.3	Create an application test report. The test report can be written according to requirement 6.1.4.5.	openETCS application data test report.
8.4.4.4	Create an application Preparation Verification Report. The openETCS application preparation verification report shall be created according to the requirement of 8.4.4.4.	openETCS application data preparation verification report.
8.4.4.5 and 8.4.4.6	Create an application Test Specification. The openETCS Application Test Specification shall be created according to the requirement of 8.4.4.6.	openETCS Application data Test Specification.
8.4.4.7 and 8.4.4.8	Create an application Data Verification Report. The openETCS Application Data Verification Report shall be created according to the requirement of 8.4.4.8.	openETCS Application Data Verification Report.

C.6.5 Application Data Integration and Testing Acceptance

Table C21. Integration and Tests of application data of the generic openETCS software (Subclause 8.4.5 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
8.4.5.1 and 8.4.5.2	Created a test specification for the application data of the openETCS application software. The created application test specification shall implement the requirement of 8.4.5.2	openETCS application data test specification.

Table C21. Integration and Tests of application data of the generic openETCS software (Subclause 8.4.5 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
8.4.5.3 and 8.4.5.4	Verify the created openETCS application data test specification. • The results of the verification shall be recorded in the openETCS application data verification report, • The created openETCS application data verification report shall implement the requirements of 8.4.5.4.	openETCS application data verification report **NOTE** The test report can also be written according to requirement 6.2.4.12.

C.6.6 Application Data validation

The validation of the application data of the openETCS software shall be carried out according to subclause C.5.7.

C.6.7 Application Data preparation procedures and tools

Table C22. preparation procedures and tools of application data of the generic openETCS software (Subclause 8.4.7 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
8.4.7.1 and 8.4.7.2	Tools for the development of application data respectively to specify the generic openETCS application software, shall fulfill the requirements for tools in class T3 according to EN 50128:2011	See Annex B. **Recommendation** Use the openETCS tool chain.
8.4.7.3 and 8.4.7.4	See subclause C.7.	See subclause C.7.
8.4.7.5	A configuration management plan for application data shall be created. **NOTE** **The configuration management plan for application data shall be separated from the configuration management of the openETCS application software**	openETCS Configuration management plan for application data.

C.6.8 Application Data: Development of Generic Software

Table C23. Additional requirements for the development of the generic openETCS software (Subclause 8.4.8 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
8.4.8.2	Requirement 8.4.8.2 this requirement shall be implemented during the requirements specification of the openETCS software	See requirement 7.2.4.1.
8.4.8.3	Requirement 8.4.8.3 this requirement shall be implemented during the interface specification of the openETCS software.	See requirement 7.3.4.18.
8.4.8.4	**The openETCS software must be independent from the application data/algorithms**.	**highly recommended.**
8.4.8.5	Requirement 8.4.8.5 this requirement shall be implemented in the Change Management Process of the openETCS software.	See requirement 6.6.4.1.
8.4.8.6	Requirement 8.4.8.6 **this requirement shall be implemented in the software test plan, software verification plan and software validation plan of the openETCS software**.	See requirements 6.1.4.1, 6.2.4.1 and 6.3.4.1.

Table C23. Additional requirements for the development of the generic openETCS software (Subclause 8.4.8 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
8.4.8.7	Requirement 8.4.8.7 this requirement shall be implemented during the design specification of the openETCS software.	See requirement 7.3.4.20.
8.4.8.8	Requirement 8.4.8.8 this requirement shall be implemented in the Release Note of the openETCS software.	See requirement 7.7.4.12.

C.7 deployement and maintenance (Clause 9 according to EN 50128:2011)

C.7.1 openETCS Software deployment

The objective of this subclause is to ensure that the openETCS software performs as required, preserving the required software safety integrity level and dependability when it is deployed in the final environment of application (non vital OBU).

Table C24. Deployment of the openETCS Software (Subclause 9.1 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
9.1.4.1 and 9.1.4.5	Define a deployment process for the openETCS software. The deployment process of the openETCS software shall implement the requirements 9.1.4.4, 9.1.4.5 and 9.1.4.12 to 9.1.4.20.	Deployment manual of the openETCS software.
9.1.4.6 to 9.1.4.9	Verify the create Deployment manual of the openETCS software • The results of the verification shall be recorded in the deployment verification report of the openETCS software, • The created deployment verification report of the openETCS software shall implement requirements from 9.1.4.6 to 9.1.4.9.	Deployment verification report of the openETCS software.

C.7.2 openETCS Software maintenance

The objective of this subclause is to ensure that the openETCS software performs as required, preserving the required software safety integrity level and dependability when making corrections, enhancements or adaptations to the openETCS software itself.

Table C25. Maintenance of the openETCS Software (Subclause 9.2 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
9.2.4.1 to 9.2.4.6	Define a maintenance process for the openETCS software The maintenance process for the openETCS software shall implement the requirement 9.2.4.6, 9.2.4.12 to 9.2.4.19.	Maintenance Plan for the openETCS software.
9.2.4.7 to 9.2.4.10	Verify the created Maintenance Plan for the openETCS software. • The results of the verification shall be recorded in the maintenance verification report of the openETCS software, • The created maintenance verification report of the openETCS software shall implement requirements from 9.2.4.7 to 9.2.4.10.	Maintenance verification report of the openETCS software.

Table C25. Maintenance of the openETCS Software (Subclause 9.2 according to EN 50128:2011)

Requirements	How the evidence shall be provided	Documents to be created
9.2.4.12	Select techniques from the Table A.10 of EN 50128:2011 for maintenance activities. The selected techniques shall be suitable for the software SIL4	Record the selected techniques in the Maintenance Plan of the openETCS software.

Appendix: References

[1] Cecile Braunstein, Jan Peleska, Johannes Feuser
Position paper on openETCS development work flow and associated tools
University of Bremen, October 2012.

[2] European Committee for Electrotechnical Standardization
EN 50126
Railway applications - The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS)- Part 1: Basic requirements and generic process
1999.

[3] European Committee for Electrotechnical Standardization
EN 50128
Railway applications - Communication, signalling and processing systems - Software for railway control and protection systems
2011.

[4] European Committee for Electrotechnical Standardization
EN 50129
Railway applications - Communication, signalling and processing systems- Safety related electronic systems for signalling
2003.

[5] European Committee for Electrotechnical Standardization
EN 50159
Railway applications - Communication, signalling and processing systems - Safety-related communication in transmission systems
2010.

[6] European Railway Agency
TSI CCS
Technical specification for interoperability relating to the control-command and signalling subsystems of the trans-European rail system
2006.

[7] Union Industry of Signalling
UNISIG SUBSET-026 - System Requirements Specification
Version 3.3.0, 2012.

[8] Airlines Electronic Engineering Committee
ARINC 653
Avionics application software standard interface
Part 1 - Required services (December 1, 2005)
Part 2 - extended services (anuary 22, 2007).

[9] Klaus-Rüdiger Hase
Project Outline Full Project Proposal - openETCS: Open Proofs Methodology for the European Train Control System
Version 3.0, January 2013.

[10] Klaus-Rüdiger Hase
Project Co-operation Agreement
Version 3.0, December 2012.

[11] David Wheeler
Countering Trusting Trust through Diverse Double-Compiling (DDC)- Countering Trojan Horse attacks on Compilers
http://www.dwheeler.com/trusting-trust
2009.

[12] Ken Thompson
Reflections on Trusting Trust
August 1984.

[13] David Kalinsky
Security fundamentals for embedded software
http:// www.kalinskyassociates.com
2009.

[14] European Parliament
European Parliament resolution on the existence of a global system for the interception of private and commercial communications (ECHELON interception system)
http://cryptome.org/echelon-ep-fin.htm
2001.

[15] Control command and signalling subsystems - CCS TSI
2012/88/EU Commission Decision of 25 January 2012 on the technical specification for interoperability relating to the control-command and signalling subsystems of the trans-European rail system
http://www.era.europa.eu/Document-Register/Pages/CCS-TSI.aspx
2012.

[16] ERTMS/ETCS System Requirements Specification - SRS
New Annex A for ETCS Baseline 3 and GSM-R Baseline 0
http://www.era.europa.eu/Document-Register/Pages/New-Annex-A-for-ETCS-Baseline-3-and-GSM-R-Baseline-0.aspx
2012.

[17] NATIONAL AERONAUTICS AND SPACE ADMINISTRATION WASHINGTON, DC 20546
Formal methods specification and verification guidebook for software and computer systems Volume I: planning and technology insertion
Release 1.0, July 1995.

[18] Jürgen Weber, Michael Horst, Norbert Schäfer
Gutachterliche Stellungnahme Ãijber den Einsatz von Open Source Software im Bereich ETCS Fahrzeug AusrÃijstung
Berichtsnummer:1011G02-Gut
08.12.2010

[19] Shaoying Liu
Formal Engineering for Industrial Software Development Using the SOFL Method
Springer-Verlag
2004

[20] S. Bacherini, A. Fantechi, M. Tempestini, N. Zingoni
A Story about Formal Methods Adoption by a Railway Signaling Manufacturer
in Proc. FM 2006, Hamilton, Canada, August 2006, Lecture Notes in Computer Science, 4085, 1996.

[21] Chad Dougherty, Kirk Sayre, Robert C. Seacord, David Svoboda, Kazuya Togashi
TECHNICAL REPORT: Secure Design Patterns
CMU/SEI-2009-TR-010
ESC-TR-2009-010
March 2009; Updated October 2009