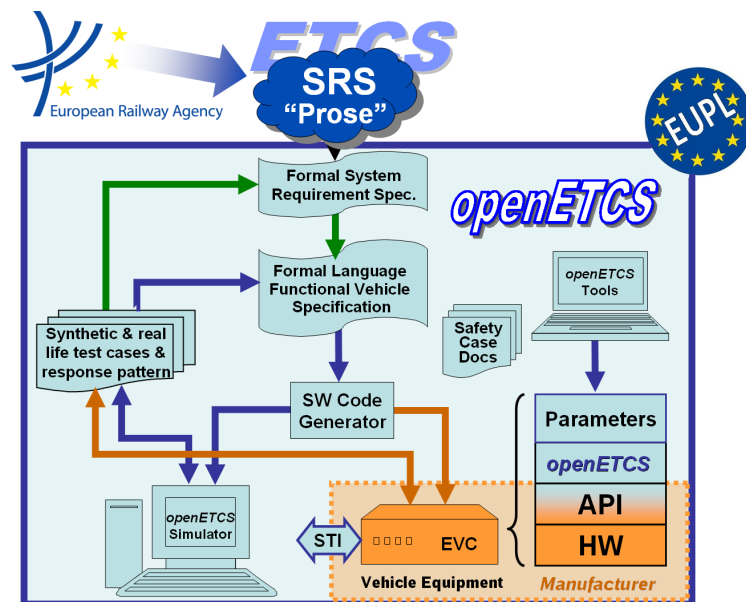


Work Package 3: "Modeling"

openETCS System Architecture and Design Specification

Baseliyos Jacob, Peter Mahlmann

May 2015



Funded by:


 Federal Ministry
 of Education
 and Research

 Région de
 Bruxelles-
 Capitale

 GOBIERNO
 DE ESPAÑA

 MINISTERIO
 DE INDUSTRIA, ENERGÍA
 Y TURISMO


This page is intentionally left blank

Work Package 3: “Modeling”**OETCS/WP3/D3.5.0
May 2015**

openETCS System Architecture and Design Specification

Document approbation

Lead author:	Technical assessor:	Quality assessor:	Project lead:
location / date	location / date	location / date	location / date
signature	signature	signature	signature
Baseliyos Jacob (DB Netz AG)	Jan Welte (Technische Universität Braunschweig)	Izaskun de la Torre (SQS)	Klaus-Rüdiger Hase (DB Netz)

Baseliyos Jacob, Peter Mahlmann
DB Netz AG

Architecture and Design Specification

Prepared for openETCS@ITEA2 Project

Abstract: This document gives an introduction to the architecture of openETCS. The functional scope is tailored to cover the functionality required for the openETCS demonstration as an objective of the ITEA2 project. The goal is to develop a formal model and to demonstrate the functionality during a proof of concept on the ETCS Level 2 Utrecht Amsterdam track with real scenarios. It has to be read as a complement to the models in SysML and Scade languages.

Disclaimer: This work is licensed under the "openETCS Open License Terms" (oOLT) dual Licensing: European Union Public Licence (EURL v.1.1+) AND Creative Commons Attribution-ShareAlike 3.0 – (cc by-sa 3.0)

THE WORK IS PROVIDED UNDER openETCS OPEN LICENSE TERMS (oOLT) WHICH IS A DUAL LICENSE AGREEMENT INCLUDING THE TERMS OF THE EUROPEAN UNION PUBLIC LICENSE (VERSION 1.1 OR ANY LATER VERSION) AND THE TERMS OF THE CREATIVE COMMONS PUBLIC LICENSE ("CCPL"). THE WORK IS PROTECTED BY COPYRIGHT AND/OR OTHER APPLICABLE LAW. ANY USE OF THE WORK OTHER THAN AS AUTHORIZED UNDER THIS OLT LICENSE OR COPYRIGHT LAW IS PROHIBITED.

BY EXERCISING ANY RIGHTS TO THE WORK PROVIDED HERE, YOU ACCEPT AND AGREE TO BE BOUND BY THE TERMS OF THIS LICENSE. TO THE EXTENT THIS LICENSE MAY BE CONSIDERED TO BE A CONTRACT, THE LICENSOR GRANTS YOU THE RIGHTS CONTAINED HERE IN CONSIDERATION OF YOUR ACCEPTANCE OF SUCH TERMS AND CONDITIONS.

<http://creativecommons.org/licenses/by-sa/3.0/>
<http://joinup.ec.europa.eu/software/page/eupl/licence-eupl>

Modification History

Version	Section	Modification / Description	Author	Date
0.1	Document	Initial document providing structure	Peter Mahlmann	27.05.2015

Table of Contents

Modification History	iii
Figures and Tables.....	vi
1 Introduction.....	1
1.1 Motivation.....	1
1.2 Objectives	1
1.2.1 Scope of deliverables	2
1.2.2 Scope of the ADD document.....	3
1.2.3 Scope of the User Validation Scenarios	4
1.2.4 Scope of the RFC process.....	5
1.2.5 Objectives and scope related to formal executable model	6
1.3 Roles, responsibilities and tasks	6
1.4 openETCS Design Process	7
1.4.1 SRS- Driven Approach vs. Functional Approach.....	7
1.4.2 Top- down vs. Bottom- Up	8
1.4.3 Working groups	8
1.4.4 Use- case driven setting of priorities	8
1.4.5 integration.....	8
1.4.6 Towards EN50128 SIL4 objectives, interface to WP4	9
1.5 openETCS history and iterations	9
2 Input documents	10
3 Use case description - proof of concept Utrecht - Amsterdam	12
3.1 Proof of concept on the Track Utrecht Amsterdam User Stories 1 - 4	12
3.1.1 Use Case and Scenario 1: Start of Mission - Awakening of the Train	12
3.1.2 Use Case and Scenario 2: Start of Mission - Start in Level 2 Mode FS	12
3.1.3 Use Case and Scenario 3: Brake intervention - Revocation of a Movement Authority and Overrun Permitted Speed	13
3.1.4 Use Case and Scenario 4: ETCS Onboard Unit is reading and sending track information ..	13
3.2 Environment model for the use case demonstrations.....	14
3.3 Dynamic track model of the ETCS Level 2 line Amsterdam- Utrecht	15
3.3.1 Model concept	15
4 Architecture Description	20
4.1 System Architecture view in ERA TSI Subset 25 Chapter 2 "Basic System Description".....	20
4.1.1 System Structure from the subchapter 2.4. of ERA TSI Subset 26 chapter 2	20
4.1.2 Sub System from the subchapter 2.5. of ERA TSI Subset 26 chapter 2	20
4.1.2.1 2.5.1 Trackside subsystem.....	20
4.1.3 System Structure from the subchapter 2.4. of ERA TSI Subset 26 chapter 2	22
4.2 System Architecture SysML View	22
4.2.1 1st Level System Architecture View	22
4.2.2 2nd Level System Architecture View	22
4.2.3 3rd Level System Architecture View	26
4.3 Interfaces	26
4.3.1 External Interfaces	26

4.3.2 Internal Interfaces	27
5 Runtime API	28
5.1 Introduction to the Architecture	28
5.1.1 Abstract Hardware Architecture	28
5.1.2 Definition of the reference abstract hardware architecture	28
5.1.3 Reference abstract software architecture	28
5.2 Functional breakdown	31
5.2.1 F1: openETCS Runtime System and Input to the EVC)	31
5.2.1.1 Principles for Interfaces (openETCS)	32
5.2.1.2 openETCS Model Runtime System	32
5.2.1.3 Input Interfaces of the openETCS API From other Units of the OBU.....	32
5.2.1.4 Message based interface (BTM, RTM)	33
5.2.1.5 Interfaces to the Time System	34
5.2.1.6 Interfaces to the Odometry System.....	35
5.2.1.7 Interfaces to the Train Interfaces (TIU).....	36
5.2.1.8 Output Interfaces of the openETCS API TO other Units of the OBU	36
Index	36

Figures and Tables

Figures

Figure 1. Document relationships in WP3.	3
Figure 2. Analysis of input documents.	11
Figure 3. Environment for use case demonstration.	14
Figure 4. High-level view of dynamic model with balises and radio block center.	16
Figure 5. Example of balise position data.	17
Figure 6. Example of balise message header data.	17
Figure 7. Graphical representation of five balise groups.	18
Figure 8. Example of radio message and packet data.	19
Figure 9. Scope of system according to ERA TSI Chapter 2.	23
Figure 10. 1st level system architecture view.	24
Figure 11. 2nd level system architecture view.	25
Figure 12. Eurobalise	26
Figure 13. DMI Interfaces.	27
Figure 14. Reference abstract hardware architecture.	29
Figure 15. Reference abstract software architecture.	30
Figure 16. openETCS API Highlevel View.	31

Tables

1 Introduction

1.1 Motivation

The openETCS work package 3 (WP3) aims to provide – amongst others - the software architecture for the openETCS kernel in order to eventually build the software itself.

The WP3 partners had by the end of 2014 put great effort in the openETCS software design, thus far without making definite choices on the software architecture itself respective of functional breakdown and data structures of the openETCS kernel. Since the project planning foresees in the production of a reference software to be used as a demonstrator by June 2015, it was of paramount importance that a first design deliverable of the openETCS kernel architecture be finalized shortly but no later than November 2014.

This document shows a snapshot of the software and will also give an outlook to the scope of the next iteration and the final objectives.

1.2 Objectives

The prime objective of WP3 is to produce a fully formal prototype for the openETCS reference system that can function as a demonstrator in collaboration with WP 4 and WP 5 for the openETCS approach and will be used as such in the final phase of the project. That phase is the first half of 2015. This objective is defined as ...

High level Objectives of this work: «any further general statements on the ITEA2 objectives, like...»

- Work on a model bases approach and process for effective collaborative work within an international ETCS developer team as stated above, the project needs a definite architecture design by the end of 2014.

This document targets:

- Defining the general design and conditions of the openETCS architecture, functional breakdown, data structures, behaviour and interfaces.
- Providing the guidelines for discussion during the workshops that are planned in October and November 2014 that will result in the final and decisive version of this document
- Being the ‘platform’ for finalization i.e. whatever be the products or results of the workshops shall be integrated in this document.

Apart from these general objectives, the document means to provide for the materials that will enable WP3 partners to improve the efficiency of the Work Package activities:

- The comprehensive architecture design shall enable splitting the work load according to the building blocks defined by the architecture and allocate strictly compartmented work parcels or activities to WP3 partners.

- Doing so will enable WP3 to avoid any double work
- Compartmenting the work load according to the functional building blocks as defined by the architecture will enable efficient planning of activities, be it individually or the integrated WP3 planning for the coming period, aiming at a just in time delivery of all results and products
- Compartment the work load according to the functional building blocks as defined by the architecture will enable efficient planning of activities, be it individually or the integrated WP3 planning for the coming period, aiming at a just in time delivery of all results and products

1.2.1 Scope of deliverables

The primary objective of WP3 was to produce a rapid prototype for the openETCS reference system that can function as a demonstrator in collaboration with WP 4 and WP 5 for the openETCS approach and will be used as such in the final phase of the project. That phase is the first half of 2015.

We have to see this original objective in the context of the overall innovation promise of openETCS and the related exploitation potential. Taking into account the current situation of the project (as of Nov. 2014) we have to make choices that make sure that the remaining budget and resources are wisely used, which necessitates that the WP3 deliverables become more "multi- purpose":

- provide a functional formal specification of openETCS, based on functional analysis, and traceable to the SRS.
- Iteratively develop the software architecture and design description document (ADD) while using the formal specification as an executable functional model.
- Utilise real- world use-case scenarios as early as possible in the development process.

If we take this approach, we can use the iterative development method, while applying a pragmatic view of SCRUM (which has been defined as the high- level project management method for this project) in order to build the following deliverables:

- An ADD document, providing a functional description of software architecture and design.
- A RFC (Request For Comment) process that provides a cross- reference to the SRS, listing all requirements and design conflicts that have been identified during our work.
- A formal, executable model of the openETCS kernel, which can directly be used as entry point into a (future) EN5012x SIL4 compliant software design process, as well as functional reference during simulation and on the planned WP5 demonstrator.

Each main WP3 collateral has it's distinct role in the overall project structure. For a schematic view of the dependencies, see Figure 1.

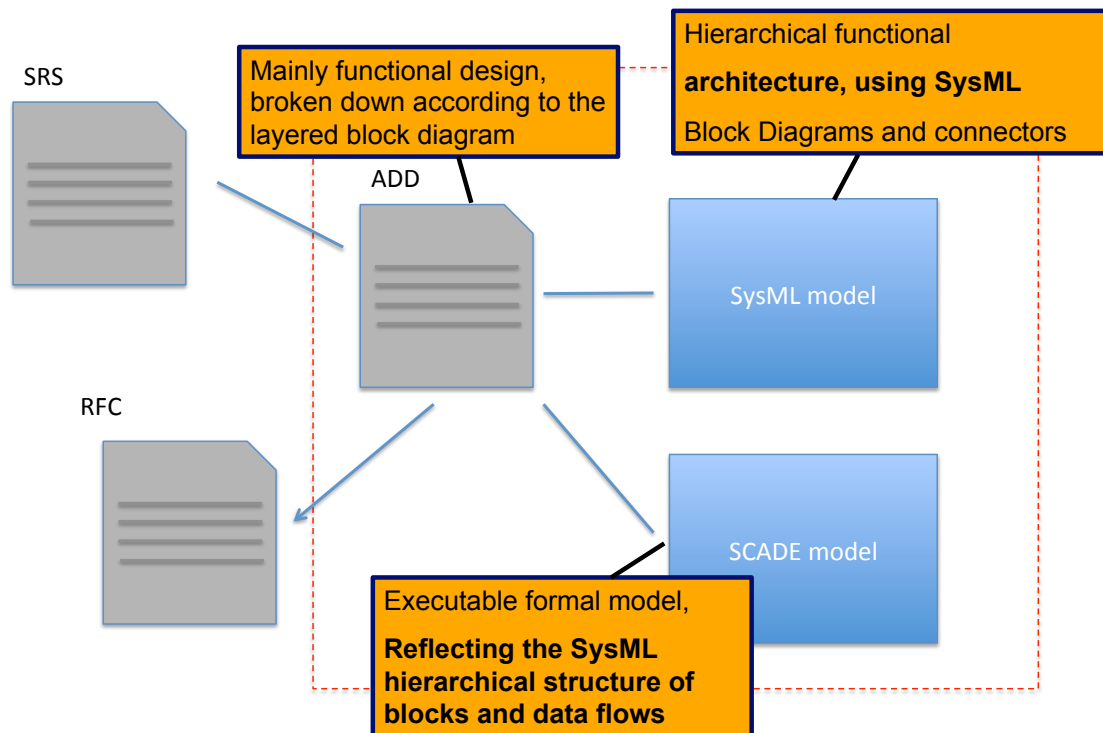


Figure 1. Document relationships in WP3.

1.2.2 Scope of the ADD document

The ADD document is intended to provide the following main information:

- Internal guidance for the project: Provide authoritative guidance on process, responsibilities, process and workflow.
- Architectural design description: Provide complete information on the overall architecture of the openETCS kernel functions.
- Functional design description: Provide comprehensive information on the design of the system, which includes QoS aspects, interfaces, data types, algorithms and finally the full software design.

The nature of the development process implies that this document will be a "living document", meaning that it will be constantly updated, with new iterations being planned as follows:

- Intermediate work iterations (0.x): as required until the ITEA2 intermediate review (March 2015)
- First main release (1.0): as a deliverable for the ITEA2 intermediate review (March 2015).
- Intermediate work iterations (1.x): as required until the demonstrator milestone (June 2015).
- Second main release (2.0): for the demonstrator milestone (June 2015).
- Intermediate work iterations (2.x) for the rest of the project duration (until Oct 2015).
- Final release (3.0) for the final review.

Intermediate work iteration shall be managed using the normal day-to-day pull/ review process, while main releases require a formal document sign-off.

1.2.3 Scope of the User Validation Scenarios

The work of WP3 has now been aligned to the milestones that are relevant for the overall project:

1. ITEA2 intermediate review (March 2015)
2. Demonstrator milestone (June 2015)
3. Final review (End of 2015)

In order to support the functional approach, we use operational scenarios that are described in the User Validation Scenario Document.

These are based on the following data:

- Rules from NS for the operation of trains
- Infrastructure data from the Utrecht- Amsterdam line

In order to support the functional analysis, which is aligned with the project's milestones, this document will define use cases along the following principles:

1. Selected operational scenarios on selected sections of the Utrecht- Amsterdam track
 Based on nominal scenarios, our simulated train will drive across actual (simulated) infrastructure, selecting only a few, typical balise groups. This will allow WP3 to implement the relevant mode/level/message (balise and RBC) functionality first.
 The openETCS kernel model has been complemented by a simple DMI representation, in order to validate actual behaviour.
 In march 2015, only basic scenarios will be demonstrated, which have been organised as use cases, based on a plant/ controller co-simulation concept.
 The future
2. Selected operational scenarios on the full Utrecht- Amsterdam track
 Based on nominal scenarios, our simulated train will drive across actual (simulated) infrastructure, covering the entire Utrecht- Amsterdam track. This will allow WP3 to implement the relevant mode/level/message (balise and RBC) functionality first, for full functional coverage of the test track.
 Compared to the first step in March, we will also cover more operational scenarios according to the NS operational rules.
 A generic, interactive simulation concept will be created for this purpose.
 The entire track with its 488 balise groups and known (real) RBC messages as well as specific validation scenarios will be modelled.
 A presentation of this concept can be found in the section "Dynamic Simulation".

3. Full coverage of Utrecht- Amsterdam

For the final review, we will prepare a openETCS kernel software that can run defined scenarios, according to the NS operation regulation, across the entire Utrecht- Amsterdam line, either replaying actual train rides or simulating various events, nominal and non-nominal.

In cooperation with WP4 and WP5 this will provide additional input for demonstration and validation.

1.2.4 Scope of the RFC process

The RFC document is a deliverable that was added to the WP3 set of documents during the Nov 3-7, 2014 Munich workshop.

Most (industrial) ETCS OBU systems have been built based on pure SRS analysis. at least in theory. In practise, each implementation is derived from the SRS, but with a mindset that is partially driven by the company culture and local operational regulation prior to ERTMS introduction.

This has lead to a plethora of ETCS systems, which exhibit subtle differences which lead to incomplete compatibility between OBUs and track equipment of different suppliers. Each OBU works best on a track which was built by the same supplier in the same country.

One of the objectives of openETCS is to provide a reference design for an ETCS UBU that reduces these variations.

The experts of WP3 and WP4 agree that a functional approach will lead to a better understanding of the system. As the SRS remains the formal reference, we have to provide formal feedback which is traceable to the SRS.

This is the objective of the RFC document.

The RFC document shall be structured following the paragraphs in the SRS and shall provide full traceability to the openETCS ADD. Each design decision that can be traced to a deviation from the SRS or that highlights inconsistencies inside the SRS shall be documented in the RFC.

The RFC is hence a deliverable that provides formal feedback to the relevant ERTMS stakeholders.

(we still have to discuss the precise interaction between the SRS Analysis effort, WP4 and this RFC document)

The nature of the development process implies that this document will be a "living document", meaning that it will be constantly updated, with new iterations being planned as follows:

- Intermediate work iterations (0.x): as required until the ITEA2 intermediate review (March 2015)
- First main release (1.0): as a deliverable for the ITEA2 intermediate review (March 2015)
- Intermediate work iterations (1.x): as required until the demonstrator milestone (June 2015)

- Second main release (2.0): for the demonstrator milestone (June 2015)
- Intermediate work iterations (2.x) for the rest of the project duration (until Oct 2015)
- Final release (3.0) for the final review.

Intermediate work iteration shall be managed using the normal day-to-day pull/ review process, while main releases require a formal document sign-off.

1.2.5 Objectives and scope related to formal executable model

During the previous iterations of this document, it was proposed to use a traditional waterfall model in order to define the architecture and subsequently the functional and software design of the openETCS kernel software. We have now shifted the design paradigm to a more agile process, where we will use iterations (functional analysis, implementation of prototype formal executable model, refinement) in order to design the software modules. (bottom- up design of modules)

These will then be merged into a top- down architecture by the chief architect, using best practises that are well established for data- flow oriented software designs.

The result will be a functional, formal model, from which code can be generated that implements the functionality in an hardware- independent way.

This code is hardware- and platform agnostic and can be integrated with the openETCS runtime environment, which is adaptable to different system architectures and APIs.

1.3 Roles, responsibilities and tasks

In this section, the roles and responsibilities of the WP3 partners are confirmed, especially where they divert from what has been agreed upon at the start of WP3:

- **Responsibilities** First of all, in the last WP 3 meeting in Brussels on 10.09.2014 DB proposed to take over the lead of the architecture design and functional breakdown. At the subsequent weekly scrum meeting on 12.09.2014, it was agreed upon by all participants that DB will take over the lead (see Appendix ...);
- **Planning:** Alstom as WP 3 leader will remain to be responsible for the planning and the allocation of the defined tasks to the different partners
- **Roles:** Alstom will also coordinate the work and safeguard that the defined results will be delivered according to the quality requirements that are agreed within the ITEA2 project and the schedule and the milestones that will be agreed upon during the coming workshops;
- All WP3 partners will deliver the results or products according to planning as will be agreed upon during the said workshops.

In the interest of a swift production of the critical documentation of which this version is a draft, specific tasks will be defined in terms of concrete results to be delivered, the timeframes

in which these results must be produced and the partner who shall be responsible for that specific result and the planning. This is to safeguard the timely delivery. The process will be described in the next sections.

1.4 openETCS Design Process

1.4.1 SRS- Driven Approach vs. Functional Approach

Most attempts to formalise the ETCS onboard software have been focusing on creating models that were directly mirroring the SRS specification. While this gives a full picture of the status quo of the SRS, it is not sufficient in order to fully understand the main issues that stem from the approach by which the SRS was conceived.

The SRS is aiming in what could be called the quadrature of the circle:

- Define a common specification of all aspects of the requirements of the ETCS system (Basic System Description, Principles, Functionality, Procedures and Application Level Communication Protocols, amongst others)
- Create a framework that is compatible with all the local ways of driving trains
- Ensure a framework for simple interoperability
- Give a full understanding of the wayside and onboard views of the system

Instead of building yet another copy-paste direct formalisation of the ETCS SRS, we are proposing a different approach:

- Functional analysis of the SRS
 - onboard point-of view
 - SRS driven
 - Focused on the reference track (ETCS L2 Utrecht- Amsterdam)
- Formalisation
 - Executable formal model (using SCADE tool suite)
 - Dynamic simulation
- Validation
 - interface to WP4

Together with the document structure described above, this will for the first time lead to a OBU-centric formalisation of the SRS, with systematic Change Request process (through our RFC approach) and a functional reference that can be used to validate ETCS OBU software as well as wayside implementations once it is completed.

1.4.2 Top- down vs. Bottom- Up

The functional approach, together with other considerations mandated a combined top-down and bottom up design approach. While the basic architecture was derived from existing architectures, with basic concepts and functional breakdown derived from the existing implementations of the WP3 leader, the actual functional analysis and the related software design was built bottom-up.

Thanks to the model- based formal design approach, the components could then be seamlessly merged with the top-down architecture.

Using this approach, the system and software architecture was at a relatively immature approach when we split the work in order to separately analyse and design the different functional blocks of the system.

The openETCS agile approach allows to iteratively integrate and align the top-down architectural view with the separately developed main functional blocks.

1.4.3 Working groups

While the basic architectural concept has been driven jointly by Alstom, DB and NS, the analysis and design of the functional blocks was distributed into working groups. Each working group consists of a combination of application/ analysis specialists and software/SCADE specialists.

1.4.4 Use- case driven setting of priorities

In order to focalise the work and to ensure our functional approach, DB has taken the role of product owner. As a reminder: in an agile/ SCRUM project management approach, the product owner has the responsibility to represent the interests of the customer. The goal being to prove the openETCS concept and model on the ETCS L2 Utrecht- Amsterdam line, DB is formulating a set of at first very basic, then advanced and towards project completion comprehensive and complete scenarios and requirements for demonstration.

In- line with agile project management methodology, these requirements are presented to the WP3 team in the form of "user stories" which help to drive the project priorities.

1.4.5 integration

Integration on openETCS context is focusing on several aspects:

- Integration of SCADE functional blocks with overall SCADE architectural model
In a SCADE context, model-to-model integration is straightforward. As each functional block has a clear interface and all blocks have identical semantics we could follow an iterative, constructive and agile approach. Integration was mainly driven by alignment of interfaces and validated by interactive simulation.
- Integration of openETCS model with WP3 simulation environment
For the March 2015 ITEA review, a simple dynamic simulation environment was set up using SCADE tools (SCADE Display to build a first version of an openETCS DMI software, SCADE Rapid Prototyper to interact with the simulator). Integration of the openETCS OBU software model followed the same principles as model-to-model integration. The same

concept will be extended for a full dynamic simulation of the Utrecht-Amsterdam line in interaction with the openETCS reference model.

- Integration of openETCS model with openETCS runtime

Currently, the demonstration is purely on model/ simulation level. However, since the openETCS toolchain allows for automatic code generation, we will integrate the generated code with the openETCS runtime. As the generated code is strictly target agnostic, integration can be adapted very flexibly to different hosts and execution models. Details of this integration phase will be discussed in the final version of this document, a first overview is given in the section discussing the APIs.

1.4.6 Towards EN50128 SIL4 objectives, interface to WP4

One of the objectives of openETCS is to demonstrate feasibility of the approach and concept in a CENELEC context. While we are focusing on analysis and design for now, we will also formulate a certification strategy.

An actual EN50128 certification of the openETCS software is beyond the scope of this project.

1.5 openETCS history and iterations

The openETCS Architecture and Design is being implemented in iterations. The current step (third iteration) is implementing essential kernel functions of the ETCS system. For a better understanding of the scope the Iteration is described in the following.

Third Iteration Functional Scope: The OBU functions for Szenarios defined in chapter 3.

The openETCS third iteration model is defining the architecture and design of the openETCS OBU software. It is referencing [?] UNISIG Subset_026 version_3.3.0.

The appropriate functionality has been divided into a number of functional blocks that are being analysed and designed by work groups (as described in the previous section of this document).

The third iteration of the openETCS modelling effort is focusing on demonstrating some simple scenarios on the actual (for now simulated) Utrecht- Amsterdam infrastructure.

2 Input documents

This section gives an overview about the input documents used for the

- analysis of the OBU functions,
- functional decomposition and allocation of functional blocks, functions and libraries,
- design of the OBU functions, and
- determination of "use cases" and scenarios for the different iterations of the Architecture and Design Document

List of main documents that are being used as reference or input for analysis and design:

- ERA TSI CCS Documents
- openETCS API Spec
- openETCS Requirements WP 2
- Railway Operator Documents
- Industry Documents
- ERSA Simulator Documents
- Other Project Partner Data information and documents
- Utrecht - Amsterdam Track Documents

Furthermore, relevant ETCS know how from industry and operators is used for the design and the analysis.

While the list above serves as a high-level reference, detailed information, links to the actual documents and additional remarks are being maintained at <https://github.com/openETCS/modeling/wiki/Input-Documents-Repository> while the documents describing the standard are referenced at <https://github.com/openETCS/SSRS/wiki/SSRS-Documents>

The figure below illustrates the relationships among the input used documents:

For a detailed discussion of the actual work process, please refer to the previous chapters.

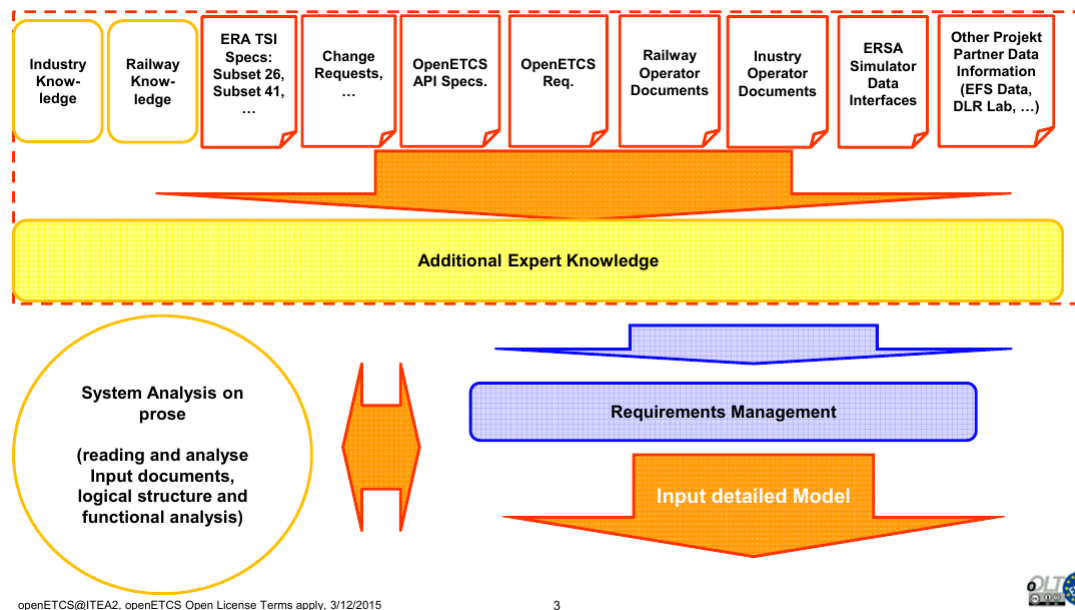


Figure 2. Analysis of input documents.

3 Use case description - proof of concept Utrecht - Amsterdam

3.1 Proof of concept on the Track Utrecht Amsterdam User Stories 1 - 4

The goal of the openETCS@ITEA2 project is to deliver at the end a proof of concept in a lab on a real ETCS track. Since the Level 2 Utrecht - Amsterdam track was evaluated as the most appropriate reference track for this concept due to the maturity and representativeness of the track, it will be used for the mentioned simulation.

To start with the realisation of the concept in an iterative way in the same pattern the industry is proceeding and regarding the "classical" state of the art of system analysis we started in this third iteration with the following Use Cases and Scenarios:

3.1.1 Use Case and Scenario 1: Start of Mission - Awakening of the Train

This use case according to the procedure in chapter 5 will demonstrate the start of a train from a no power mode to the state that the train will be ready for level and mode change according to the chapter 5. Link on GitHub: <https://github.com/openETCS/modeling/issues/66>. The following Subsystems need to be realised for Scenario 1:

- Procedure
- TIU management
- DMI management and controller
- Position report
- Management of radio communication
- Manage track data
- Manage mode and level
- Train supervision

3.1.2 Use Case and Scenario 2: Start of Mission - Start in Level 2 Mode FS

This use case according to the procedure in chapter 5 will demonstrate the start of a train from the awakening of the train in mode stand-by to the state that the train will receive a movement authority in level 2 and change into the mode full supervision to start running under real supervision according to the chapter 5. Link on GitHub: <https://github.com/openETCS/modeling/issues/67>. The following Subsystems need to be realised for Scenario 2:

- Procedure
- TIU management

- DMI management and controller
- Position report
- Management of radio communication
- Manage track data
- Manage mode and level
- Train supervision

3.1.3 Use Case and Scenario 3: Brake intervention - Revocation of a Movement Authority and Overrun Permitted Speed

This use case according to the subset 26 chapter 3 principles will demonstrate the brake intervention that will cause by a revocation of a movement authority due to a occupied section or track an due simple overrun of a permitted speed according to the chapter 3. Link on GitHub <https://github.com/openETCS/modeling/issues/68>. The following Subsystems needs to be realised for Scenario 3:

- Procedure
- TIU management
- DMI management and controller
- Position report
- Management of radio communication
- Manage track data
- Manage mode and level
- Train supervision

3.1.4 Use Case and Scenario 4: ETCS Onboard Unit is reading and sending track information

This use case according to the subset 26 chapter 3 principles will demonstrate the full completeness and checking the reading and sending of track information in interaction with the ETCS Onboard Unit and the track that will be separated in radio and balise messages. Messages and packages are defined in chapter 7 and 8 of the subset 26. Link on GitHub <https://github.com/openETCS/modeling/issues/69>. The following Subsystems needs to be realised for Scenario 4:

- Procedure
- TIU management
- DMI management and controller
- Position report



Figure 3. Environment for use case demonstration.

- Management of radio communication
- Manage track data
- Manage mode and level
- Train supervision
- Building of coordinate system

3.2 Environment model for the use case demonstrations

In order to dynamically explore and demonstrate the openETCS OBU kernel software, a dynamic simulation and demonstration environmental model is being created. During Iteration 3, this comprises the following features and functionalities (c.f. Figure 3.):

- openETCS OBU formal model
- openETCS DMI formal model with Display specification model
- Simplified version, not all features are implemented yet; following our use- case driven approach we are only implementing features that are essential to show the use cases selected by our "internal customer".
- Environment model with
 - simplified track model (balise locations, speed profile)
 - simplified model of Movement Authority
 - interactive widgets to manipulate the simulation environment

3.3 Dynamic track model of the ETCS Level 2 line Amsterdam- Utrecht

This environment model will be fourthly enhanced during the last project period, in order to:

- Allow full dynamic simulation of the Utrecht- Amsterdam line.
- Form a basis for a (future) dynamic track simulator, which models the balise locations, balise messages and RBC messages for any given line, and which can automatically be generated from engineering datasets.
- Provide a full track model for the purposes of openETCS.

The principle of this model is shown in Figure ???. The idea is to split the model in two parts:

- Generic functions, that represent the behaviour of the trackside installations (for example balise groups, RBC reception, RBC sending).
- Data that instantiate these generic functions (for example balise locations, balise message/ packet data, radio message/ packet data).

This simulation approach is designed to be adaptable to very different kinds of scenarios:

Proof of concept in openETCS: Amsterdam-Utrecht line. In that case we have a partially known line (balise locations and track topology are known, messages and packets are known "as-is" from JRU data owned by the partners).

We have been using SCADE graphical notation for the development of the concept in order to disseminate it more easily. The models will however be instantiated by automatic conversion of engineering and JRU data to simulation scenarios for "nominal" operations validation.

Specific test cases will either be transformed by script from test data bases (in WP4) or can also be graphically modelled (for user validation scenarios or demonstration purposes).

OBU validation for existing tracks. Once the concept and simulation environment have been validated in openETCS context, the tooling can be used to import track data from other tracks, such as the Corridor A, in order to develop interoperable ETCS specifications.

Track validation for existing OBUs (or against the TSI). With a validated OBU software model, the system can also be used to validate new track layouts. Dynamic track simulation opens new dimensions of attacking the interoperability issues that Europe is facing.

3.3.1 Model concept

Traditionally, most ETCS simulators use predefined simulation scenarios, which define the inputs to the model at predefined steps. They can either be time triggered (update every n time units), distance- oriented (update every n distance units) or cycle- oriented (update every simulation cycle)

The disadvantage of such an approach is clear: all parameters have to be predefined and pre-validated, for example to simply make sure that the time/ distance ratio is consistent. In such a

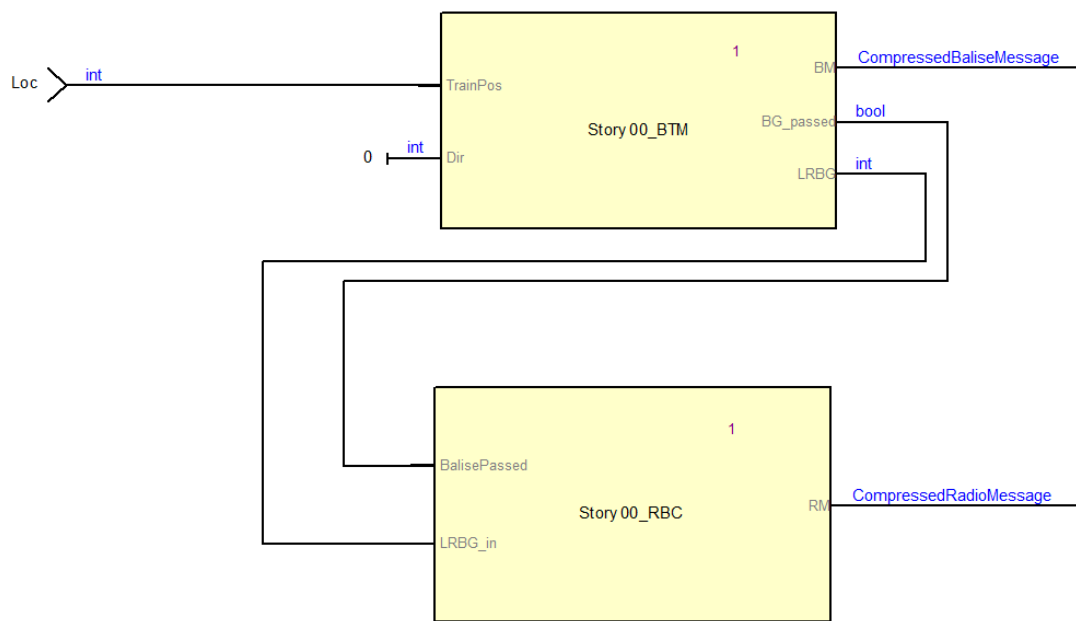


Figure 4. High-level view of dynamic model with balises and radio block center.

setting it is very difficult to simulate events in just one dimension (independently of the other parameters).

By creating functional models of the track elements and combining them with data we are now able to explore much more complex scenarios with for example interactive driver interaction, thus not only exploring the implementation of the OBU or the engineering / layout of the track, but also the impact on operational procedures, human factors.

As all events can be recorded and replayed and are based on a formal model (also for the track model and engineering data), the entire environment can be validated and potentially even be certified. The same EN50128 certifiable tool chain is used that serves the consortium for modelling of the OBU itself.

The basic concept is very simple, but can be developed into a full dynamic track and RBC model. One functional block represents the balises, one functional block represents the RBC (see Figure 4.) When train movement is simulated, the functional blocks representing the balises and the RBC, respectively, receive the steadily updated train position (via the input "Loc") and react accordingly. When a balise group detects a passing train, it reacts accordingly by emitting the defined messages and packets.

In an analog way, the RBC block reacts as defined when it receives a train to track message from the simulated OBU by emitting the appropriate set of messages and packets. While the functionality of the balise groups is modelled in order to reflect their behaviour, they are instantiated using parameter sets that are defined as SCADE constants. One dataset each defines:

- Balise positions (from engineering data), (c.f. Figure 5).

Field	Type	Value
NID_C	int	426
NID_BG	int	139
Pos	int	20155
Or_BG	OrBG	Utrecht
Or_Line	OrLine	N

Figure 5. Example of balise position data.

Constant	Type	Value
B139_H1	BaliseTelegramHeader_int_T	
q_updown	int	1
m_version	int	16
q_media	int	0
n_pig	int	0
n_total	int	1
m_dup	int	1
m_mcount	int	255
nid_c	int	426
nid_bg	int	139
q_link	int	1
B139_H2	BaliseTelegramHeader_int_T	
q_updown	int	1
m_version	int	16
q_media	int	0
n_pig	int	1
n_total	int	1
m_dup	int	2
m_mcount	int	255
nid_c	int	426
nid_bg	int	139
q_link	int	1

Figure 6. Example of balise message header data.

In this example, the balise's engineering data have a local coordinate system (km on track, nominal orientation in direction of Amsterdam or Utrecht, respectively, and orientation of the line (Z= zuid, Dutch for southbound, N= northbound))

- Balise messages and packets (from JRU data), c.f. Figure 6).
Actual data from the Utrecht- Amsterdam line can be seen in this example. Typical for a Level 2 line is that most balise groups are only used as positional reference for the train. The balise groups are then put together to a "track" which emits telegrams as the train "passes" over it. This concept is scaleable and the balises can equally instantiated using textual Scade models which can automatically from (for example) JRU data (c.f. Figure 7).
- RBC messages and packets (from JRU data). The modelling of the RBC follows a heavily simplified concept at the moment: The simulated RBC emits telegrams/messages/packets in predefined situations. For example, the reception of a specific train-to-track message such as a specific position report triggers emission of a set of messages and packets, such as a complete Movement Authority message with packets for international static speed profile, national values, gradient profile, MA, etc. The functional block may seamlessly be replaced with a full RBC simulation. Figure 8 shows a dataset representing RBC packets in SCADe constant format.
- Potentially, any required data set can be defined on both the input side (static data or scenarios) and on the output side (expected outcome for validation).

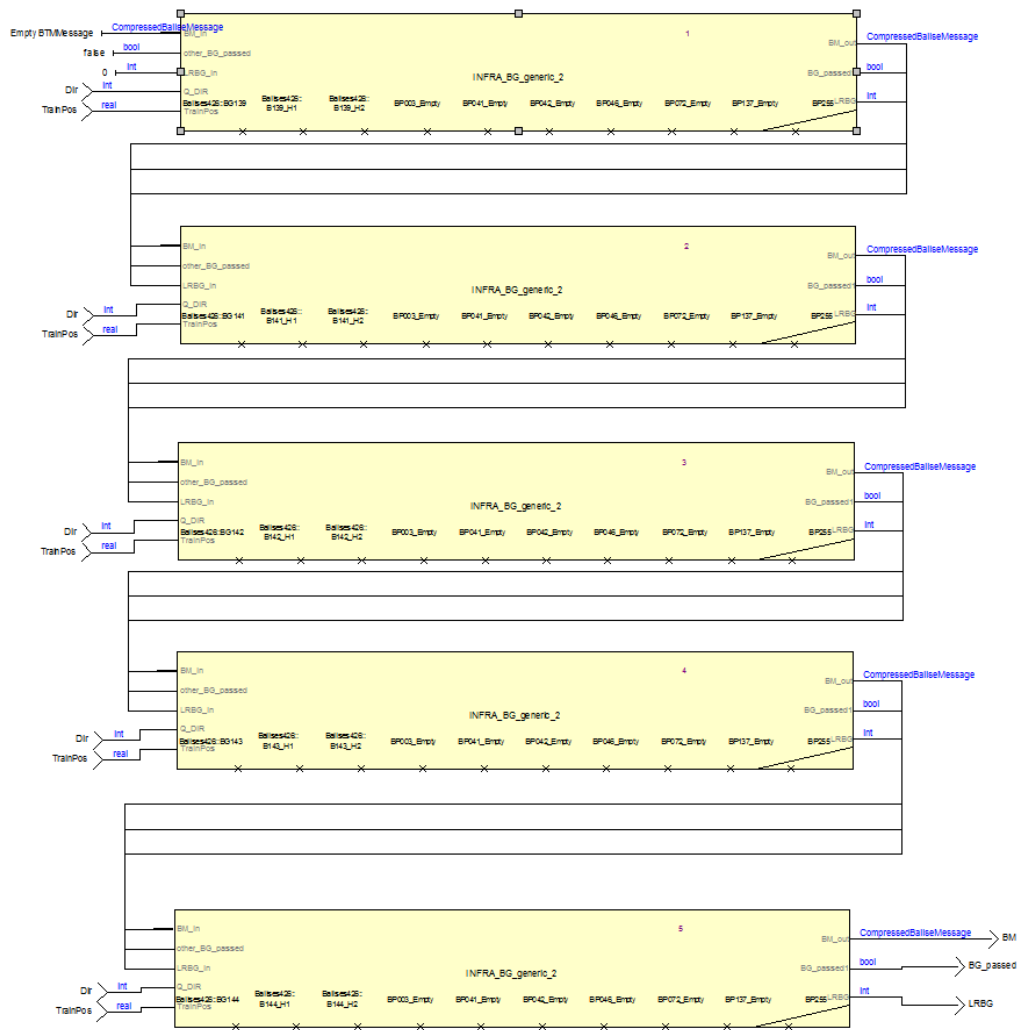


Figure 7. Graphical representation of five balise groups.

Constant	Type	Value
RM139_00Header	Radio_TrackTrain_Header...	
radioDevice	int	0
receivedSyste...	int	0
nid_message	int	3
t_train	int	4048087
m_ack	int	1
nid_lrbg	int	140
t_train_reference	int	0
nid_em	int	0
q_scale	int	0
d_sr	int	0
t_sh_rqst	int	0
d_ref	int	0
q_dir	int	0
d_emergencyst...	int	0
m_version	int	0
RM139_03	RMP03	
valid	bool	true
Q_DIR	int	2
L_PACKET	int	MaxElementsRPacket03
Q_SCALE	int	1
D_VALIDNV	int	0
N_ITER	int	1
NID_C	int	426
V_NVSHUNT	int	40
V_NVSTFF	int	40
V_NVONSIGHT	int	40
V_NVUNFIT	int	10
V_NVREL	int	15
D_NVROLL	int	5
Q_NVSRBKTRG	int	0
Q_NVEMRRLS	int	0
V_NVALLOWO...	int	0
V_NVSUPOVT...	int	40
D_NVOVTRP	int	200
T_NVOVTRP	int	60
D_NVOTRP	int	60
M_NVCONTACT	int	0
T_NVCONTACT	int	35
M_DVDERUN	int	1
D_NVSTFF	int	-1
Q_NVDRIVER	int	1
RM139_05	RMP05	
valid	bool	true
Q_DIR	int	1
L_PACKET	int	MaxElementsRPacket05
Q_SCALE	int	1
N_ITER	int	4
SECTIONS	RMP05Es_T	
0	RMP05E_T	{valid : true, D_LINK : 435, Q_NEWCOUNTRY : 0, NID_BG : 139, Q_LINKORIENTATION : 1, Q_LINKREACTION : 0, Q_LOCACC : 2}
1	RMP05E_T	{valid : true, D_LINK : 54, Q_NEWCOUNTRY : 0, NID_BG : 141, Q_LINKORIENTATION : 0, Q_LINKREACTION : 0, Q_LOCACC : 2}
2	RMP05E_T	{valid : true, D_LINK : 973, Q_NEWCOUNTRY : 0, NID_BG : 142, Q_LINKORIENTATION : 0, Q_LINKREACTION : 0, Q_LOCACC : 2}
3	RMP05E_T	{valid : true, D_LINK : 54, Q_NEWCOUNTRY : 0, NID_BG : 143, Q_LINKORIENTATION : 0, Q_LINKREACTION : 0, Q_LOCACC : 2}
4	RMP05E_T	{valid : true, D_LINK : 167, Q_NEWCOUNTRY : 0, NID_BG : 144, Q_LINKORIENTATION : 1, Q_LINKREACTION : 0, Q_LOCACC : 2}

Figure 8. Example of radio message and packet data.

4 Architecture Description

4.1 System Architecture view in ERA TSI Subset 25 Chapter 2 "Basic System Description"

The system architecture is an outcome on the functional decomposition according to the analysis of the document in § 2 Input documents. A starting point for the first analysis is the System Structure in ERA TSI Subset 26 chapter 2 the subchapter 2.5 and 2.4.

Since we decided in the openETCS project not to consider all subsystem due to our using scope for the proof of concept on the ETCS Level 2 Utrecht - Amsterdam line, we analysed the necessary subsystem as seen here:

4.1.1 System Structure from the subchapter 2.4. of ERA TSI Subset 26 chapter 2

- 2.4.1.1 Due to the nature of the required functions, the ERTMS/ETCS system will have to be partly on the trackside and partly on board the trains.
- 2.4.1.2 This defines two sub-systems, the on-board sub-system and the trackside sub-system.
- 2.4.1.3 The environment of ERTMS/ETCS system is composed of:
 - a) the train, which will then be considered in the train interface specification;
 - b) the driver, which will then be considered via the driver interface specification;
 - c) other onboard interfaces (see architecture drawing in 2.5.3),
 - d) external trackside systems (interlockings, control centres, etc.), for which no interoperability requirement will be established.

4.1.2 Sub System from the subchapter 2.5. of ERA TSI Subset 26 chapter 2

4.1.2.1 2.5.1 Trackside subsystem

§ 2.5.1.1 Depending of the application level (see further sections), the trackside sub-system can be composed of:

- a) balise
- b) lineside electronic unit - *not in the scope of this project*
- c) the radio communication network (GSM-R)
- d) the Radio Block Centre (RBC)
- e) Euroloop - *not in the scope of this project*
- f) Radio infill unit - *not in the scope of this project*
- g) Key Management Centre (KMC) - *not in the scope of this project*

2.5.1.2 Balise

- 2.5.1.2.1 The balise is a transmission device that can send telegrams to the on-board sub-system.
- 2.5.1.2.2 The balise is based on the existing Eurobalise specifications. These documents are included in the frame of the ERTMS/ETCS specifications.
- 2.5.1.2.3 The balises provides the up-link, i. e. the possibility to send messages from trackside to the on-board sub-system.
- 2.5.1.2.4 The balises can provide fixed messages or, when connected to a lineside electronic unit, messages that can be changed.
- 2.5.1.2.5 The balises will be organised in groups, each balise transmitting a telegram and the combination of all telegrams defining the message sent by the balise group.

2.5.1.3 Lineside electronic unit - *not in the scope of this project*

- 2.5.1.3.1 The lineside electronic units are electronic devices, that generate telegrams to be sent by balises, on basis of information received from external trackside systems.

2.5.1.4 Trackside radio communication network (GSM-R)

- 2.5.1.4.1 The GSM-R radio communication network is used for the bi-directional exchange of messages between on-board sub-systems and RBC or radio infill units.
- 2.5.1.4.2 Intentionally deleted

2.5.1.5 RBC

- 2.5.1.5.1 The RBC is a computer-based system that elaborates messages to be sent to the train on basis of information received from external trackside systems and on basis of information exchanged with the on-board sub-systems.
- 2.5.1.5.2 The main objective of these messages is to provide movement authorities to allow the safe movement of trains on the Railway infrastructure area under the responsibility of the RBC.
- 2.5.1.5.3 The interoperability requirements for the RBC are mainly related to the data exchange between the RBC and the on-board sub-system.

2.5.1.6 Euroloop - *not in the scope of this project*

- 2.5.1.6.1 The Euroloop subsystem operates on Level 1 lines, providing signalling information in advance as regard to the next main signal in the train running direction.
- 2.5.1.6.2 The Euroloop subsystem is composed of an on-board functionality and by one or more trackside parts.

2.5.1.7 Radio infill Unit - *not in the scope of this project*

- 2.5.1.7.1 The RADIO INFILL subsystem operates on Level 1 lines, providing signalling information in advance as regard to the next main signal in the train running direction.
- 2.5.1.7.2 The RADIO INFILL subsystem is composed of an on-board functionality and by one or more trackside parts (named RADIO INFILL Unit).

2.5.1.8 KMC - *not in the scope of this project*

- 2.5.1.8.1 The role of the KMC is to manage the cryptographic keys, which are used to secure the EURORADIO communications between the ERTMS/ETCS entities (ERTMS/ETCS on-board equipments, RBCs and RIUs).

2.5.2 On-board sub-system *kernel part of the project*

2.5.2.1 Depending of the application level (see further sections), the on-board sub-system can be composed of:

- a) the ERTMS/ETCS on-board equipment;
- b) the on-board part of the GSM-R radio system;

2.5.2.2 ERTMS/ETCS on-board equipment

- 2.5.2.2.1 The ERTMS/ETCS on-board equipment is a computer-based system that supervises the movement of the train to which it belongs, on basis of information exchanged with the trackside sub-system.
- 2.5.2.2.2 The interoperability requirements for the ERTMS/ETCS on-board equipment are related to the functionality and the data exchange between the trackside sub-systems and the on-board sub-system and to the functional data exchange between the on-board sub-system and:
 - a) the driver;
 - b) the train;
 - c) the onboard part of the existing national train control system(s).

2.5.2.3 Onboard radio communication system (GSM-R)

- 2.5.2.3.1 The GSM-R on-board radio system is used for the bi-directional exchange of messages between on-board sub-system and RBC or radio infill unit.
- 2.5.2.3.2 Intentionally deleted.

2.5.3. ERTMS/ETCS Reference Architecture

The figure below shows the system scope for design of the ETCS Subsystem regarding the openETCS Onboard Unit and the Test Environment within the scope of the openETCS project proof of concept on ETCS Level 2 Utrecht - Amsterdam.

4.1.3 System Structure from the subchapter 2.4. of ERA TSI Subset 26 chapter 2

See Figure 9.

4.2 System Architecture SysML View

The SysML System view of the architecture will reflect the scope according to 4.1 and is a top down breakdown to the design layer. The functional breakdown has been done in Scade System and is part of the design model. Furthermore it will reflect all the external and internal interface that will be described in 4.3. Another goal of the System Architecture SysML view is to explain and set the boundaries for the ETCS Kernel development "F2 Kernel" as the main design part of the openETCS@ITEA2 project.

4.2.1 1st Level System Architecture View

All subsystem of the ETCS/ERTMS Basic System according in the scope of the openETCS@ITEA2 project will be reflected in this 1st level view. Furthermore the interlocking as part of a full Rail Signalling System, but not part of the openETCS scope, will be highlighted in this view.

Interlocking = interlocking is an arrangement of signal apparatus that prevents conflicting movements through an arrangement of tracks such as junctions or crossings. The signalling appliances and tracks are sometimes collectively referred to as an interlocking plant. An interlocking is designed so that it is impossible to display a signal to proceed unless the route to be used is proven safe.

4.2.2 2nd Level System Architecture View

The 2nd level system view will provide a decomposition of the ETCS on-board unit systems and the Kernel of the ETCS. The kernel is the main part of the ETCS Onboard Unit system and reflects the functions specified in the ERA TSI Subset 26. Therefore, the boundaries and interfaces to the other subsystems of the ETCS on-board unit needs to be fully described and formal. At least the formalisation kernel functions and boundaries should be realized in the openETCS project.

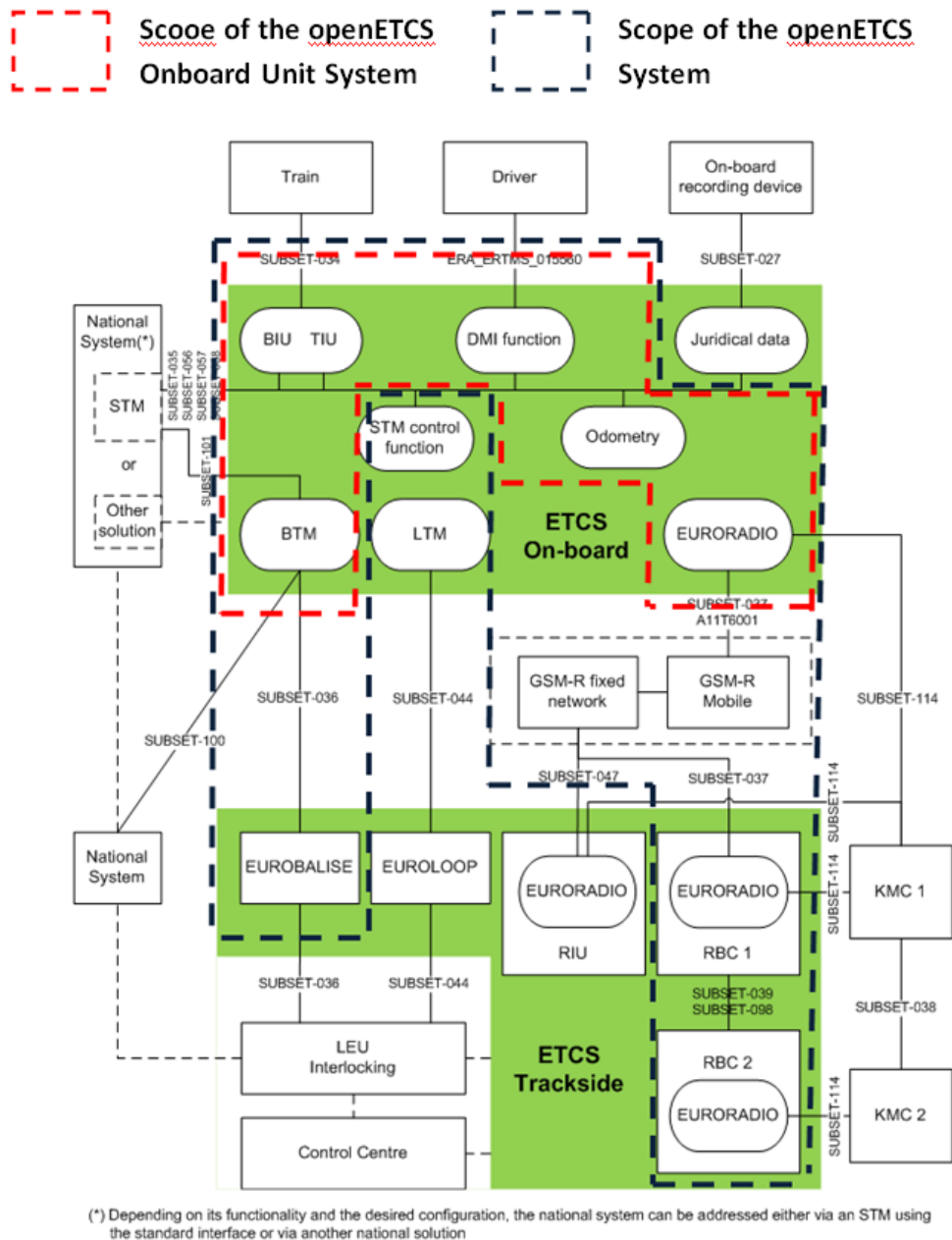


Figure 9. Scope of system according to ERA TSI Chapter 2.

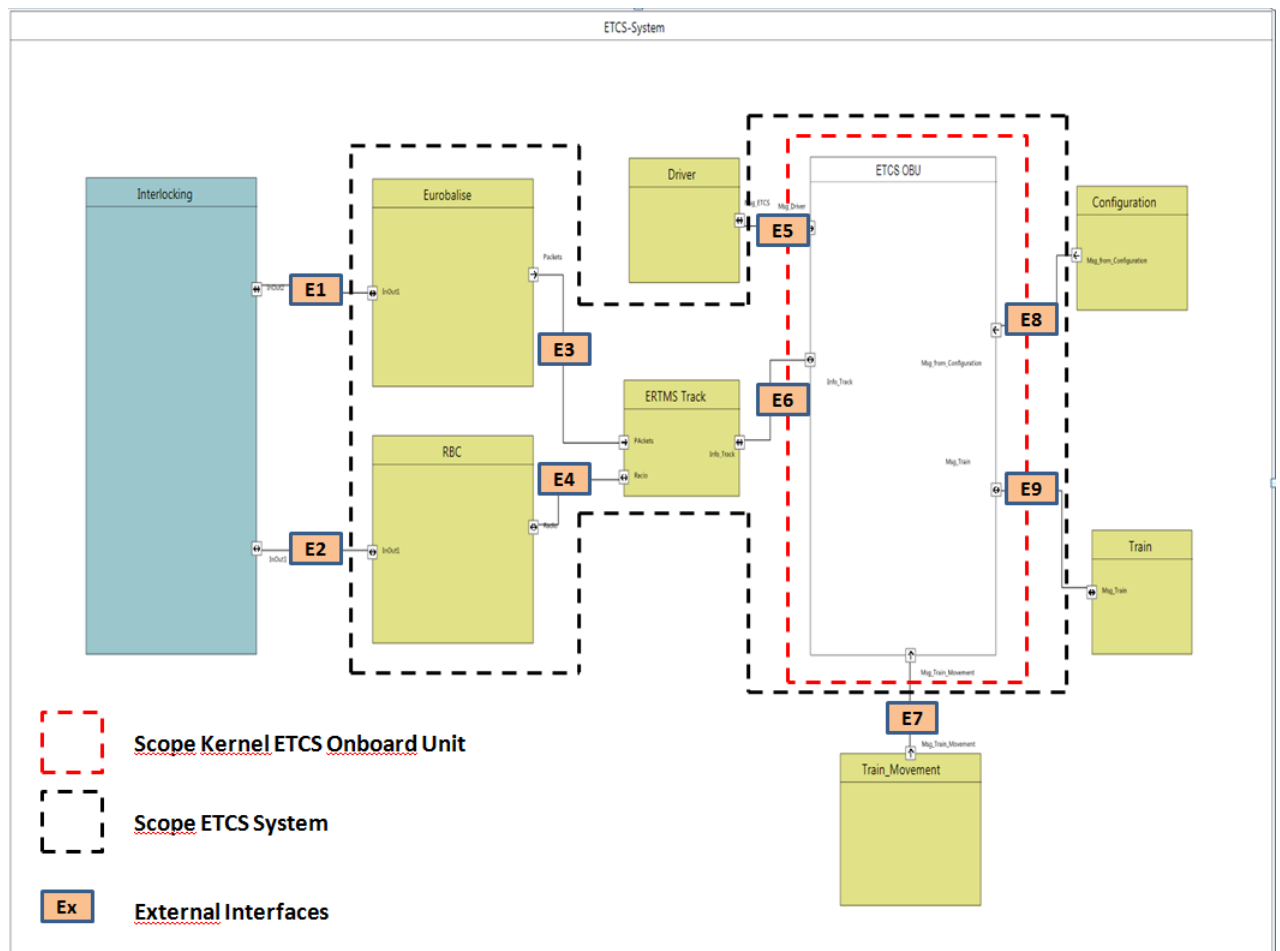


Figure 10. 1st level system architecture view.

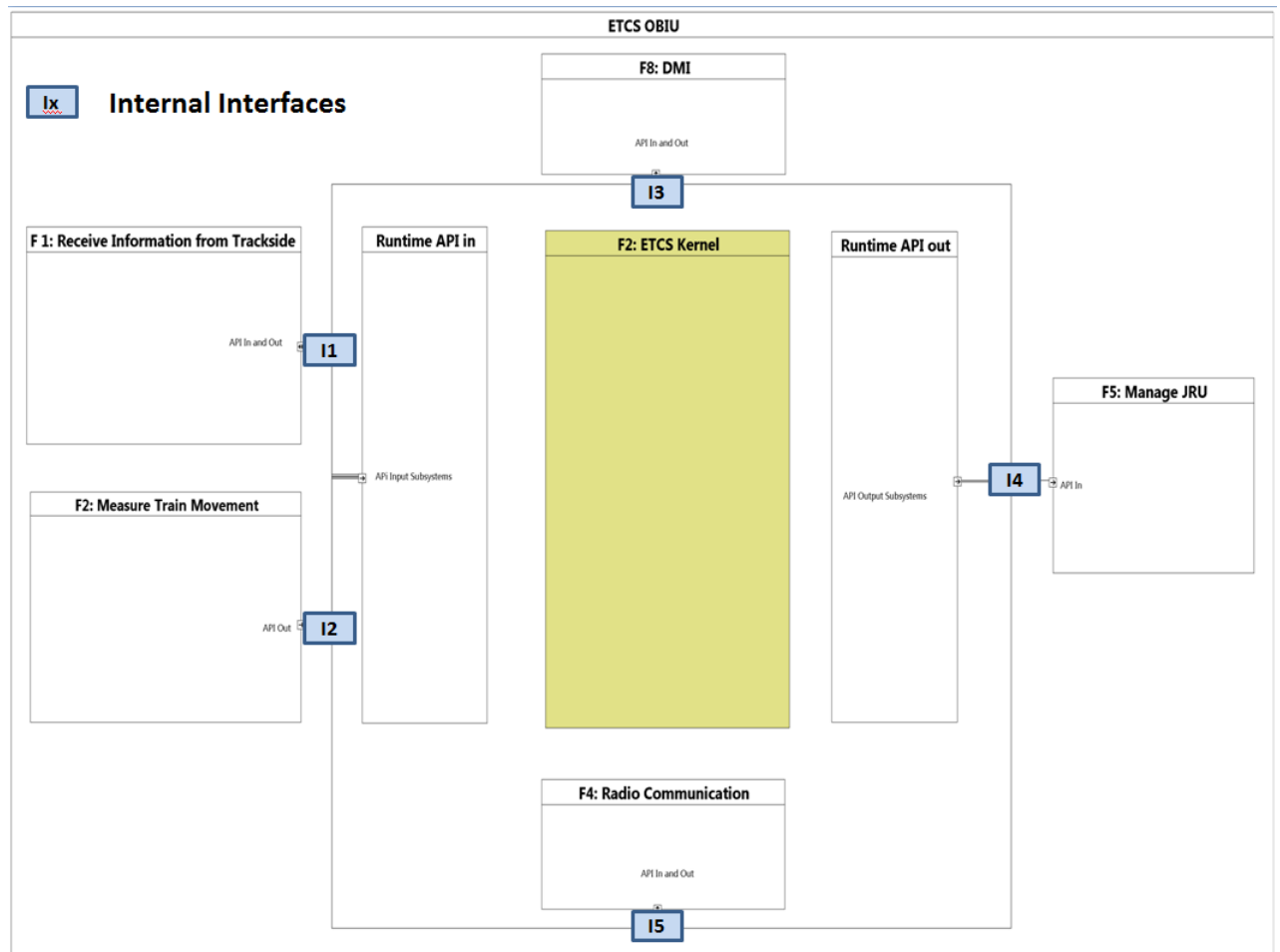


Figure 11. 2nd level system architecture view.

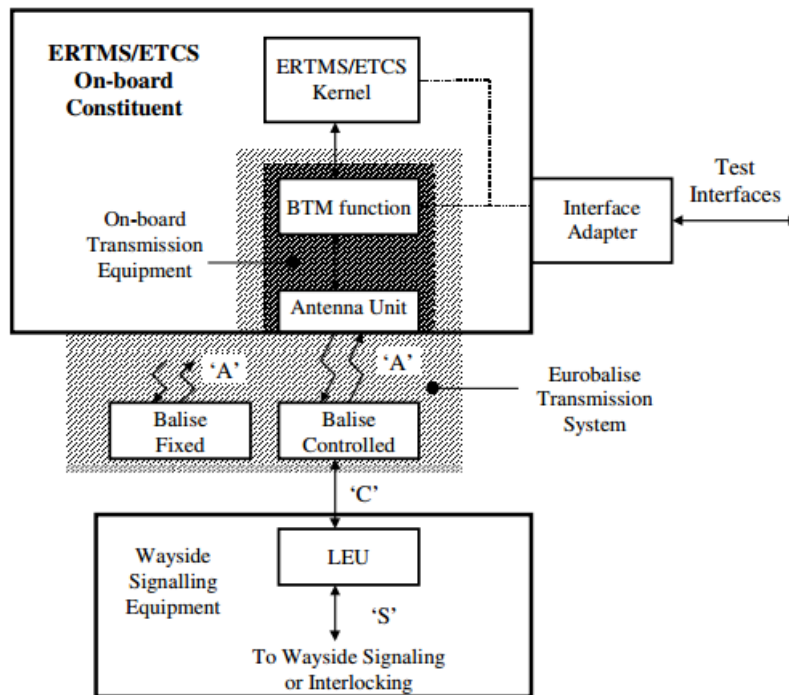


Figure 1: Eurobalise Transmission System, Interfaces

Figure 12. Eurobalise

4.2.3 3rd Level System Architecture View

The 3rd level system view will provide a decomposition of the ETCS Kernel of the ETCS Onboard Unit Systems. The decomposition and further design of the subfunctions of the kernel are part of the chapter 6 in this document. In chapter 6 we will consider the design description that will be completed by every designer itself. The designer can decide in this layer about the decomposition and boundaries of his subsystem, but need to describe the design choices.

4.3 Interfaces

This section will consider the external and internal interfaces as described in the system decomposition figures in 4.2.1 and 4.2.2.

4.3.1 External Interfaces

External interfaces will describe the data flow between the Systems outside of the scope of the openETCS Project and the ETCS Onboard Unit System.

E1: In- and out flow between the Interlocking and Eurobalise. There will be 2 kind of balises

- Fixed Balise: no interaction to the interlocking
- Balise Controlled: interaction to the interlocking through LEU

E2: In- and out flow between the Interlocking and Radio Block Control. This External interface will ensure the states or logics directly to the Radio Block Control and the other way back from the train to the interlocking.

E3: Input flow from the Eurobalise to the Balise Transmission Module or Antenna Unit (BTM) into the ETCS Onboard Unit. As already described on the figure in E1.

E4: In- and out flow between the Radio Block Control and the Euroradio Modul into the ETCS Onboard Unit. This interface is not in Level 0 or 1 active since there is no necessary for ETCS Radio interaction between track and train.

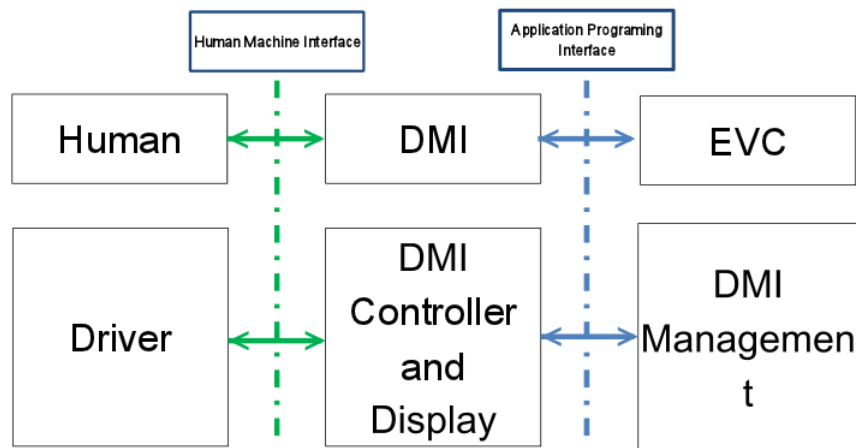


Figure 13. DMI Interfaces.

- E5:** This interface will describe the interaction between the Human and Display (Human Machine Interface or Driver Machine Interface), c.f. Figure 13.
- E6:** This interface is composite the interfaces E3 and E4.
- E7:** Input interface to the odometry Subsystem of the ETCS Onboard System. Will send information to train if there is any movement outside the ETCS System is leading such as "cold movement".
- E8:** Input interface to the ETCS Onboard Unit system to set configuration data such as fixed values, system values, national values and train configuration.
- E9:** In- and Out flow between the ETCS Onboard Unit System and the Train. This interface will describe the interaction between the Train and the ETCS Onboard Unit System such as brake control, traction control, door control,

4.3.2 Internal Interfaces

Internal interfaces will describe the data flow between the ETCS Onboard Unit Kernel and ETCS Onboard Unit Subsystems within the ETCS Onboard Unit System.

- I1:** In flow from the Balise Transmission Module (BTM or Antenna) to the "F2 ETCS Kernel" trough Runtime API in. Transmitted data are information from the Eurobalise.
- I2:** In flow from the Odometrie (ODO) to the "F2 ETCS Kernel" trough Runtime API in. Transmitted data are information from the Movement of the train.
- I3:** In- and Out flow between the DMI Controller and the "F2 ETCS Kernel" trough Runtime API in and out. Transmitted data are information of driver action and display. See description in figure of "External Interface E5".
- I4:** Out flow from "F2 ETCS Kernel" to the JRU Manager trough Runtime API out. Transmitted data are all necessary information for a juridical recorder unit "black box".
- I5:** In- and Out flow between the Euroradio and "F2 ETCS Kernel" trough Runtime API in and out. Transmitted data are radio track information (RBC) and information to the track (RBC).

5 Runtime API

5.1 Introduction to the Architecture

5.1.1 Abstract Hardware Architecture

For proper understanding of openETCS and of constraints imposed on both sides of the , we need to define a *reference abstract hardware architecture*. This hardware architecture is “abstract” in the sense that the actual vendor specific hardware architecture might be totally different of the abstract architecture described in this chapter. For example, several units might be grouped together on the same processor.

However the actual vendor specific architecture shall fulfil all the requirements and constraints of this reference abstract hardware architecture and shall not request additional constraints.

5.1.2 Definition of the reference abstract hardware architecture

The reference abstract hardware architecture is shown in figure 14.

The reference abstract hardware architecture is made of a bus on which are connected *units* defining the :

- ;
- ;
- ;
- ;
- ;
- ;
- ;
- : Not part of this openETCS implementation;
- EURORADIO;
- : Not part of this openETCS implementation;


Elements not being part of this implementation are marked.

Those units shall working concurrently. They shall exchange information with other units through asynchronous message passing.

5.1.3 Reference abstract software architecture

The *reference abstract software architecture* is shown in figure 15. This architecture is made of following elements:

- *openETCS executable model* produced by the [?] Scade Model. It shall contain the program implementing core ETCS functions;
- *openETCS model run-time system* shall help the execution of the openETCS executable model by providing additional functions like encode/decode messages, proper execution of the model through appropriate scheduling, re-order or prioritize messages, etc.
- *Vendor specific adapter* shall make the link between the Vendor specific platform and the openETCS model run-time system. It can buffer message parts, encode/decode messages, route messages to other components, etc.
- All above three elements shall be included in the ;
- *Vendor specific platform* shall be all other elements of the system, bus and other units, as shown in figure 14.



abstract-hardware-architecture.pdf

Figure 14. Reference abstract hardware architecture



Figure 15. Reference abstract software architecture

We have thus three interfaces:

- *model interface* is the interface between openETCS executable model and openETCS model run-time system.
- *openETCS* is the interface between openETCS model run-time system and Vendor specific adapter.
- *Vendor specific* is the interface between Vendor specific adapter and Vendor specific platform. This interface is not publicly described for all vendors. You can find the Alstom implementation as an example.

The two blocks openETCS executable model and openETCS model run-time system are making the *Application software* part. This Application software might be either openETCS reference software or vendor specific software.

The Vendor specific adapter is making the *Basic software* part.

5.2 Functional breakdown

5.2.1 F1: openETCS Runtime System and Input to the EVC)



Figure 16. openETCS API Highlevel View

Figure 16 shows the structure of API with respect of the software architecture. Input boxes and output boxes not implemented in this stage are marked as red, other interfaces are marked

as green. The System covers functions for processing Inputs from other Units, functions for processing Outputs to other functions and a basic runtime system. Inputs are used to feed the input to the executable model before calling it, outputs are used for collecting information provided by the executable model to be passed to the relevant interfaces after the execution cycle has finished.

5.2.1.1 Principles for Interfaces (openETCS)

Information is exchanged *messages* in an asynchronous way. A message is a set of information corresponding to an event of a particular unit, e.g. a balise received from the . The possible kind of messages are described in chapter ??.

The information is passed to the executable model as parameters to the synchronous call of a procedure (Interface to the executable model). Since the availability of input messages to the application is not guaranteed the parts of the interfaces are defined with a "present" flag. In addition, fields of input arrays quite often is of variable size. Implementation in the concrete interface in this use-case is the use of a "size" parameter and a "valid"-flag.

5.2.1.2 openETCS Model Runtime System

The openETCS model runtime system also provides:

- Input Functions From other Units
In this entity messages from other connected units are received.
- Output Functions to other Units
The entity writes messages to other connected units.
- Conversation Functions for Messages (Bitwalker)
The conversion function are triggered by Input and Output Functions. The main task is to convert input messages from an bit-packed format into logical ETCS messages (the ETCS language) and Output messages from Logical into a bit-packed format. The logical format of the messages is defined for all used types in the openETCS data dictionary.
Variable size elements in the Messages are converted to fixed length arrays with an used elements indicator.
Optional elements are indicated with an valid flag. The conversion routines are responsible for checking the data received is valid. If faults are detected the information is passed to the openETCS executable model for further reaction.
- Model Cycle

The version management function is part of the message handling. This implies, conversions from other physical or logical layouts of messages are mapped onto a generic format used in the EVC. Information about the origin version of the message is part of the messages.

The executable model is called in cycles. In the cycle

- * First the received input messages are decoded
- * The input data is passed to the executable model in a predefined order. (**Details for the interface to be defined**).
- * Output is encoded according to the and passed to the buffers to the units.

5.2.1.3 Input Interfaces of the openETCS API From other Units of the OBU

Interfaces are defined in the Scade project APITypes (package API_Msg_Pkg.xscade).

In the interfaces the following principles for indicating the quality of the information is used:

Indicator	Type	Purpose
present	bool	True indicates the component has been changed compared to the previous call of the routine
valid	bool	True indicates the component is valid to be used.

In the next table we can see the interfaces being used in the openETCS system. Details on the interfaces are defined further down.

Unit	Name	Processing Function
	Balise Telegram	Receive Messages
	Driver Machine Interface	DMI Manager
EURORADIO	Communication Management	Communication Management
EURORADIO	Radio Messages	Receive Messages
	Odometer	All Parts
System TIME	Time system of the OBU	All Parts
TIU	Train Data	All Parts

Information in the following sections gives an more detailed overview of the structure of the interfaces.

5.2.1.4 Message based interface (BTM, RTM)

Balise Message (Track to Train)

Message Name	Optional Packets	Restrictions in the current scope
Balise Telegram	3: National Values 41: Level Transition Order 42: Session Management 45: Radio Network registration 46: Conditional Level Transition Order 65: Temporary Speed Restriction 66: Revoke Temporary Speed Restriction 72: Packet for sending plain text messages 137: Stop if in Staff Responsible 255: End of Information	Used in Scenario

Balise Telegram	0, 2, 3, 5, 6, 12, 16, 21, 27, 39, 40, 41, 42, 44, 45, 46, 49, 51, 52, 65, 66, 67, 68, 69, 70, 71, 72, 76, 79, 80, 88, 90, 131, 132, 133, 134, 135, 136, 137, 138, 139, 141, 145, 180, 181, 254	Not Used in Scenario
-----------------	---	----------------------

Radio Messages (Track to Train)

Message Name	Optional Packets	Restrictions in the current scope
2: SR Authorisation	63: List of Balises in SR Authority	Message Not Supported
3: Movement Authority	21: Gradient Profile 27: International Static Speed Profile 49: List of balises for SH Area 80: Mode profile plus common optional packets	a
9: Request To Shorten MA	49: List of balises for SH Area 80: Mode profile	
24: General Message	From RBC: 21: Gradient Profile 27: International Static Speed Profile plus common optional packets From RIU: 44, 45, 143, 180, 254	Messages from RIU are not supported
28: SH authorised	3, 44, 49	
33: MA with Shifted Location Reference	21: Gradient Profile 27: International Static Speed Profile 49: List of balises for SH Area 80: Mode profile plus common optional packets	
37: Infill MA	5, 21, 27, 39, 40, 41, 44, 49, 51, 52, 65, 66, 68, 69, 70, 71, 80, 88, 138, 139	Message Not Supported
List of common optional parameters	3, 5, 39, 40, 51, 41, 42, 44, 45, 52, 57, 58, 64, 65, 66, 68, 69, 70, 71, 72, 76, 79, 88, 131, 138, 139, 140, 180	

The runtime system is in charge to transfer the messages from its stream mode first to compressed message format.

5.2.1.5 Interfaces to the Time System

The interface types are defined in the OBU_Basic_Types_Pkg Package. The system time is defined in the basic software.

The system TIME is provided to the executable model at the begin of the cycle. It is not refreshed during the cycle. The time provided to the application is equal to 0 at power-up of the EVC (it is not a “UTC time” nor a “Local Time”), then must increase at each cycle (unit = 1 msec), until it reaches its maximum value (i.e current EVC limitation = 24 hours)

- TIME (T_internal_Type, 32-bit INT)
Standardized system time type used for all internal time calculations: in ms. The time is defined as a cyclic counter: When the maximum is exceeded the time starts from 0 again.
- CLOCK (to be implemented)
The clocking system is provided by the JRU. A GPS based clock is assumed to provide the local time.

5.2.1.6 Interfaces to the Odometry System

The interface types are defined in the OBU_Basic_Types_Pkg Package. The odometer gives the current information of the positing system of the train. In this section the structure of the interfaces are only highlighted. Details, including the internal definitions for distances, locations speed and time are implemented in the package.

- Odometer (odometry_T)
 - * valid (bool)
valid flag, i.e., the information is provided by the ODO system and can be used.
 - * timestamp (T_internal_Type)
of the system when the odometer information was collected. Please, see also general remarks on the time system.
 - * Coordinate (odometryLocation_T)
 - nominal (L_internal_Type) [cm]
 - min (L_internal_Type) [cm]
 - max (L_internal_Type) [cm]

The type used for length values is a 32 bit integer. Min and max value give the interval where the train is to be expected. The boundaries are determined by the inaccuracy of the positioning system. All values are set to 0 when the train starts.
 - * speed (OdometrySpeeds_T) [km/h]
 - v_safeNominal (speed internal type) [km/h]
The safe nominal estimation of the speed which will be bounded between 98% and 100% of the upper estimation
 - v_rawNominal (speed internal type) [km/h]
The raw nominal estimation of the speed which will be bounded between the lower and the upper estimations
 - v_lower (speed internal type) [km/h]
The lower estimation of the speed
 - v_upper (speed internal type) [km/h]
The upper estimation of the speed

The type used for speed values is a 32 bit integer. Min and max value give the interval where the train is to be expected. The boundaries are determined by the inaccuracy of the positioning system. All values are set to 0 when the train starts.
 - * acceleration (A_internal_Type)[0.01 m/s²],
Standardized acceleration type for all internal calculations : in
 - * motionState (Enumeration)
indicates whether the train is in motion or in no motion
 - * motionDirection (Enumeration)
indicates the direction of the train, i.e., CAB-A first, CAB-B first or unknown.

5.2.1.7 Interfaces to the Train Interfaces (TIU)

The following information is based on the implementation of the Alstom API. The interface is organised in packets. The packets of the Alstom implementation are listed in the appendix to this document.

The description of interfaces needed for the current scope will be added according to the use.

5.2.1.8 Output Interfaces of the openETCS API TO other Units of the OBU

From Function	Name	To Unit	Description
	Radio Output Message	EURORADIO	
	Communication Management	EURORADIO	
	Driver Information		
	Train Data	TIU	

Packets: to be completed

Radio Messages to be completed