OETCS/WP3/D3.5 – 01.01

ITEA2 Project
Call 6 11025
2012 – 2015

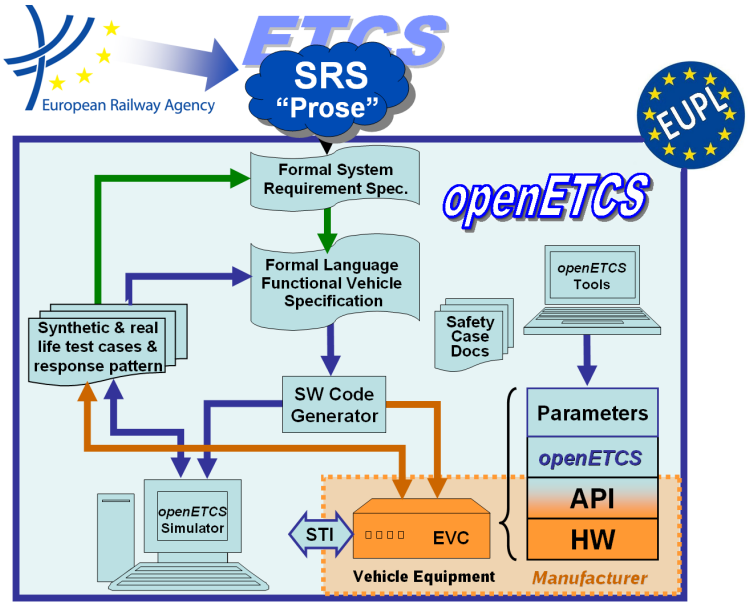
Work-Package 3: “Modelling”

openETCS System Architecture and Design Specification

First Iteration: ETCS Kernel Functions

Bernd Hekele, Uwe Steinke and Christian Stahl

September 2014



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: "Modelling"

OETCS/WP3/D3.5 – 01.01
September 2014

openETCS System Architecture and Design Specification

First Iteration: ETCS Kernel Functions

Document approbation

Lead author:	Technical assessor:	Quality assessor:	Project lead:
location / date	location / date	location / date	location / date
signature Bernd Hekele (DB-Netz)	signature [assessor name] ([affiliation])	signature Izaskun de la Torre (SQS)	signature Klaus-Rüdiger Hase (DB Netz)

Bernd Hekele

DB-Netz AG, Peter Mahlmann, Peyman Farhangi
Völckerstrasse 5
D-80959 München Freimann, Germany

Uwe Steinke

Siemens AG

Christian Stahl

TWT-GmbH

Description of work

Prepared for openETCS@ITEA2 Project

Abstract: This document gives an introduction to the architecture of the first openETCS iteration, the openETCS kernel functions. It has to be read as an add-on to the models in SysML, Scade and to additional reading referenced from the document.

Modification History

Version	Section	Modification / Description	Author
0.1	Document	Initial document providing the structure	Bernd Hekele

Table of Contents

Figures and Tables

Figures

Tables

1 Introduction

1.1 Motivation

The openETCS work package WP3 aims to provide the architecture and design of the openETCS OBU software as mainly specified in UNISIG Subset_026 version_3.3.0.

The appropriate functionality has been divided into a list of subfunctions of different complexity (see `https://github.com/openETCS/SRS-Analysis/blob/master/SystemAnalysis/List\_Functions.xlsx`).

All these functions are object of the openETCS project and have to be analyzed from their requirements and subsequently modelled and implemented. With limited manpower, a reasonable selection and order of these functions is required for the practical work that allows the distribution of the workload, more openETCS participants to join and leads to an executable– limited – kernel function as soon as possible.

While the first version of this document focuses on the first version of the limited kernel function, it is intended to grow in parallel to the growing openETCS software.

1.2 Objectives

The first objective of WP3 software shall be

- "Make the train run as soon as possible, with a very minimum functionality, and in the form of a rapid prototype."

This does not contradict the openETCS goal to provide conformance to EN50128.

- After a phase of prototyping, the openETCS software shall be implemented compliant to EN50128 for SIL4 systems.

Additional goals for this document are

- Identification of the functions required for a minimum OBU kernel
- Architecture overview regarding the minimum OBU kernel
- Technical approach: Description of the proceeding and methods to be used
- Road map of the minimum OBU kernel functions
- Road map thereafter

Note: This document will be extended according to the progress of WP3.

1.3 History

1.4 Goals of the openETCS Modelling Work

1.4.1 Functional Scope: The Minimum OBU Kernel Function

The objective “Make the train run with a very minimum functionality” shall be in terms of on ETCS OBU translated into

- The Train moves on a track equipped with balises and determines its position.

That means, for this very first step the train shall not supervise the maximum speed, shall not activate the brakes. Instead, the minimum function set shall be limited to (see https://github.com/openETCS/SRS-Analysis/issues/9)

- Receive, filter and manage balise information, received from track (see https://github.com/openETCS/SRS-Analysis/issues/12)
- Calculate the actual train position based on balise and odometry information (see https://github.com/openETCS/SRS-Analysis/issues/8)
- Calculate the distances between the actual train position to track elements in its front

A more detailed architectural breakdown of these functions is available in the form of a SysML model at (see https://github.com/openETCS/modeling/tree/master/model/sysml.

In addition, the work on this minimum functionality requires to be supported by

- The availability of the ETCS language as specified in Subset UNISIG Subset_026, chapter 7 and 8
- The abiltiy to link intermediate and final results with the requirements of the ETCS specification (subset_026, ..)
- The usability of a data dictionary (see https://github.com/openETCS/dataDictionary)

These supporting prerequisites are under construction and therefore not completely operable actually. How to deal with these restrictions, will be outlined in chapter ???

1.4.2 Actual Status

Some first analyis steps for the required minimum functionality have been gone as results from the SRS-Analysis task force. These results are available on https://github.com/openETCS/SRS-Analysis

1.4.3 Practical Approach

The architecture and design of the minimum OBU kernel shall be developed in consideration of the actual status, restricted prerequisites and limited resources as follows.

1.5 Glossary and Abbreviations

1.6 References

SRS-Subset 26
QA-Plan: D1.3.1
Process: D2.3
Methods: D2.4
API: D2.7

2 The openETCS Architecture of the initial kernel functions

2.1 The openETCS Tools-Chain and its impacts on the actual model

For understanding the modelling process and the modeling guidelines, please have a look at the
`https://github.com/openETCS/modeling/blob/master/DescriptionOfWork/NewModelingDescriptio`
`pdf`

To summarize the design process, the following ruls are in use:

- Papyrus / SysML is used for modelling the architecture. Functions are visible in this SysML level.
- No behaviour model is allowed on SysML level.
- For referencing the requirements links from the SysML model to the requirements document (in ProR) are being used.
- Details and especially behaviour is part of the Scade models.
- All interfaces (see also data-dictionary below) are available on bit-level.
- In the architecture model in SysML, all interfaces are available on a functional level for interfaces inside and outside the model and for interfaces between dedicated functions. Due to tool constrains the current model does not show all details for all interfaces (see dataDictionary).

The openETCS tools-chain for doing the modelling work consists of the following components:

Papyrus : for modelling the architecture (Kepler version).
In this phase only the Kepler version of the tool can be used due to incompatibilities of the Kepler and the Luna version on the SysML model. The SysML models are stored in the following location: `https://github.com/openETCS/modeling/tree/master/model/sysml`.

ProR : for keeping the requirements (REQIF).
The subset 26 is converted into a REQIF-format and also stored in the modeling repository on Github. The openETCS toolchain supports the linking of SysML model parts to SRS-Requirements. These results are also part of the architecture.

Scade : for designing and formalisation of the functions Scade version 15.2 is in use.
The models are stored in this location: `https://github.com/openETCS/modeling/`

`tree/master/model/Scade`. With the component Scade System Scade also has a component for architecture work.

In principle, the synch-mechanism of Scade was planned to be used for synchronisation of SysML architecture and Scade models. The idea is to automatically synchronise the SysML types and blocks with the Scade type definitions and the Scade Operators. Unfortunately, with the current set of tools this proposal was not possible to be realised. Further analysis on tools and models will look for a solution.

In addition, faults in the Kepler Papyrus version made it difficult for severla members of the team to work on different submodels of the openETCS model. The issue will be solved when changing to the Luna version of Papyrus.

3 Functions of the openETCS Model

3.1 openETCS Data Dictionary

/subsubsectiondataDictionary

3.2 openETCS Generic API

3.2.1 Generic API

Since the API currently is not defined to that level, the following assumptions are implemented:

- **Eurobalise (BTM)**: One telegram per call ...

3.3 openETCS Balise Group

3.3.1 Receive Eurobalise From API

- **Short Description of Functionality**
  This function defines the interface of the OBU model to the openETCS generic API for Eurobalise Messages. On the interface, either a valid telegram is provided or a telegram is indicated which could not be received correct when passing the balise. The function passes the telegram without major changes of the information to the next entity for collecting the balise group information.
- **Design Constrains and Choices**

  1. Decoding of balises is done at the API. Also, packets received via the interface are already transformed into a usable shape.
  2. Only packets used inside the current model are passed via the interface:
     Packet 5: Linking Information.
     Linking Information is filled into the linking array starting from index 0 without gaps. Used elements are marked as valid. Elements are sorted according to the order given by the telegram sequence.

3.3.2 Build BG Group

- **Short Description of Functionality**
  This entity collects telegrams received via the interface into Balise Group Information.
- **Reference to the SRS (or other requirements**
- **Design Constrains and Choices**

1. Telegrams received as invalid are passed to the "Check-Function" in order to process errors in communication with the trackside according to the requiremenmnts and in a single place. Telegrams are filled into the telegram array starting from index 0 without gaps. Used elements are marked as valid. Elements are stored according to the order given by the telegram sequence.
2. Only packets used inside the current model are passed via the interface:
   Packet 5: Linking Information.
3. In this function packets past with the telegrams is accumulated into a balise group information.
4. Further assumption on packet 5 (based on SRS subset 26, section 8.4.1.4)
   (In this statement the term "message" is ambiguous since it can reflect to a telegram or a balise group message) Linking Information can only be passed once. This means, if linking information for the balise group is already collected with one of the earlier telegrams, the information will not be accumulated but overwritten.

3.3.3 Check BG Consistency

3.3.4 Determine BG- Orientation and LRBG

- **Short Description of Functionality**
- **Reference to the SRS (or other requirements**
- **Design Constrains and Choices**

3.3.5 Select Usable Info

3.3.6 Perform Balise Decoding

3.4 openETCS Train Position

/subsubsectionCalculate Train Poition

References

[1] Leslie Lamport, *LaTeX: A Document Preparation System*. Addison Wesley, Massachusetts, 2nd Edition, 1994.