OETCS/WP5/D5.1

openETCS

ITEA2
INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

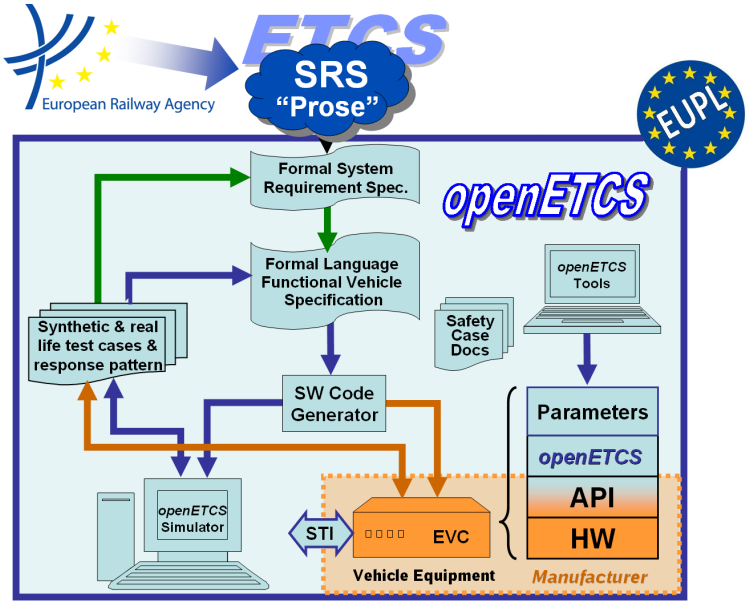
ITEA2 Project
Call 6 11025
2012 – 2015

Work-Package 5: “Demonstrator”

WP5 Demonstrator - Functional Specifications

Patrick Deutsch, Nicolas Van Landeghem, Didier Weckmann, Alexis Julin

June 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 5: “Demonstrator” **OETCS/WP5/D5.1**
June 2014

WP5 Demonstrator - Functional Specifications

Document approbation

Lead author:	Technical assessor:	Quality assessor:	Project lead:
location / date	location / date	location / date	location / date
signature Alexis Julin (ERSA)	signature Nicolas Van Landeghem (ERSA)	signature Alain Masson (ERSA)	signature Klaus-Rüdiger Hase (DB Netz)

Patrick Deutsch, Nicolas Van Landeghem, Didier Weckmann, Alexis Julin

ERSA
5 Rue Maurice Blin
67500 Haguenau, France

Description of work

Prepared for openETCS@ITEA2 Project

Modification History

Version	Section	Modification / Description	Author
0.1	All part	Creation	Patrick Deutsch
0.2	All part	First draft	Patrick Deutsch
0.3	All part	Second draft	Patrick Deutsch
0.4	All part	Third draft	Didier Weckmann
1.0	All part	First official version	Patrick Deutsch
1.1	All part	Version for review	Patrick Deutsch
1.2	All part	Update for review	Alexis Julin
1.3	All part	Update with review remarks	Alexis Julin

Table of Contents

Figures and Tables

Figures

Tables

1 Introduction

The purpose of this document is to provide a functional specification of the demonstrator included in WP5. The Demonstrator includes two main parts:

- a simulation of the ETCS on-board equipment including an EVC kernel software and a module equipment interfacing with the EVC kernel.
- a test environment for the on-board simulation.

This Functional Specification will be the main input for the development of the Demonstrator.

2 Interaction with other workpackages

This WP interacts with some other WPs of the project as follows:

2.1 Non direct interaction with WP5

2.1.1 WP1 - Project management/Administration

The purpose of this work package is to perform the project management activities and to establish and leverage the communication and controlling infrastructure by a well-defined governance concept. Reporting will be a key issue of the project management. Furthermore, the general guidelines for performing quality management will be defined and monitored. Finally, the work package will take care of intellectual property rights in terms of foreground and background knowledge.

2.1.2 WP2 - Requirements for open proof

The objective of this work package is to define the requirements for the modelling, verification and validation process in order to satisfy the safety properties of the ERTMS system.

2.1.3 WP6 - Dissemination, exploitation and standardisation

The purpose of this work package is to organize the market, exploitation, dissemination and standardization activities of the project. The respective activities will be based on individual plans and roadmaps which will be developed at the beginning of the project and which will be tracked and monitored by the Project Steering Board thereafter. The work package will follow an open innovation based approach and provide projects results to the public where appropriate.

2.1.4 WP7 - OpenETCS-Language, toolschain and Open Source ecosystem

This work package will provide the tool chain, based on formal methods, that is necessary to design, develop, verify and validate the ETCS system. This tool chain will encompass all description levels of the system design from holistic viewpoint to code generation. Specific attention shall be paid to the semantics and traceability of each part of the chain. The formal specification will be used further for verification and code generation. Other uses of the formal specification are conceivable.

2.2 Direct interaction with WP5

2.2.1 WP3 - Modelling of (part of) ETCS specification

This work package will provide the formal model of the ETCS system and the source code of the ETCS software. The tool chain developed in work package 7 will be used to model the ETCS system from system specification to software specification and then to generate the code. The modelling process will first be applied completely on a Pilot fragment of the subset-026 in order to validate the process, tools, etc... As a second iteration, the other part of subset-026 will be modelled.

Work package 5 shall provide a way to execute code generated from formal methods. Each integration step matches to a more-less independent fragment of subset-026. At the end of

OpenETCS project, all EVC code linked to subset-026 will be generated from formal methods, provided by work package 3.

2.2.2 WP4 - Validation and verification strategy

This work package will focus on the validation and verification of the model. At the very first beginning the target and requirements of the V&V strategy have to be described i.e., what should be checked? Depending on the modelling framework, the modelling language and formalization of the system requirements strategy in form of a concept has to be defined how the consistency, coherence of the model as well as the coverage of system requirements will be transparently verified. Additionally, it is important to validate the model i.e., to evidence the equivalence of the model and the ETCS system requirement specification (subset-026). In other words the reliability and acceptance of the model has to be generated, e.g. nothing is lost or added or mutated and so on. Additionally, it has to be checked that the code is consistent with the model. The work package is intended to be performed in parallel with the modelling in order to apply the strategy and to generate feedback to the modelling process as well as to measure the quality and maturity of the model. Beside a subtask will manage the consideration of all relevant safety requirements in the modelling process.

Work package 4 shall provide to work package 5 scenarios to play in the test environment. Scenario syntax will be defined with work package 5. Work package 4 shall provide a tool able to manage all scenarios created by user.

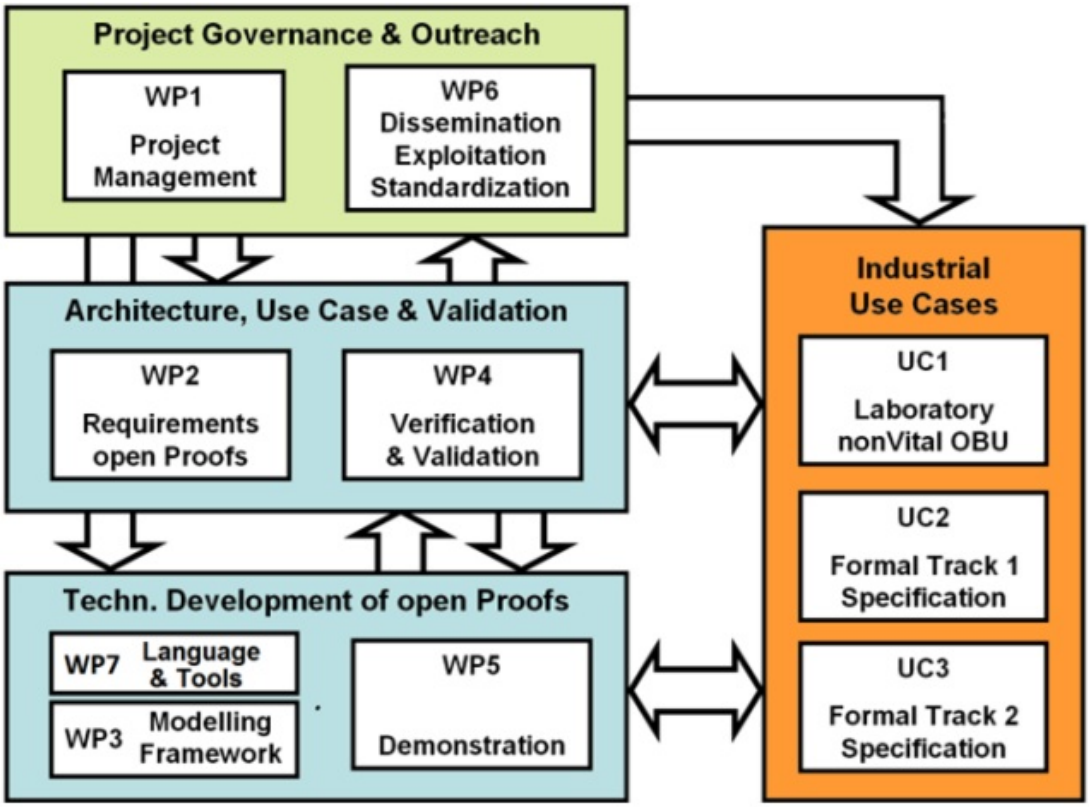


Figure 1. Work packages interactions

3 Overview of ETCS

3.1 Architecture

ETCS includes an on-board and a trackside part. An overview of the ETCS on-board and its interfaces is provided in paragraph 2 of subset-026 (see /1/).

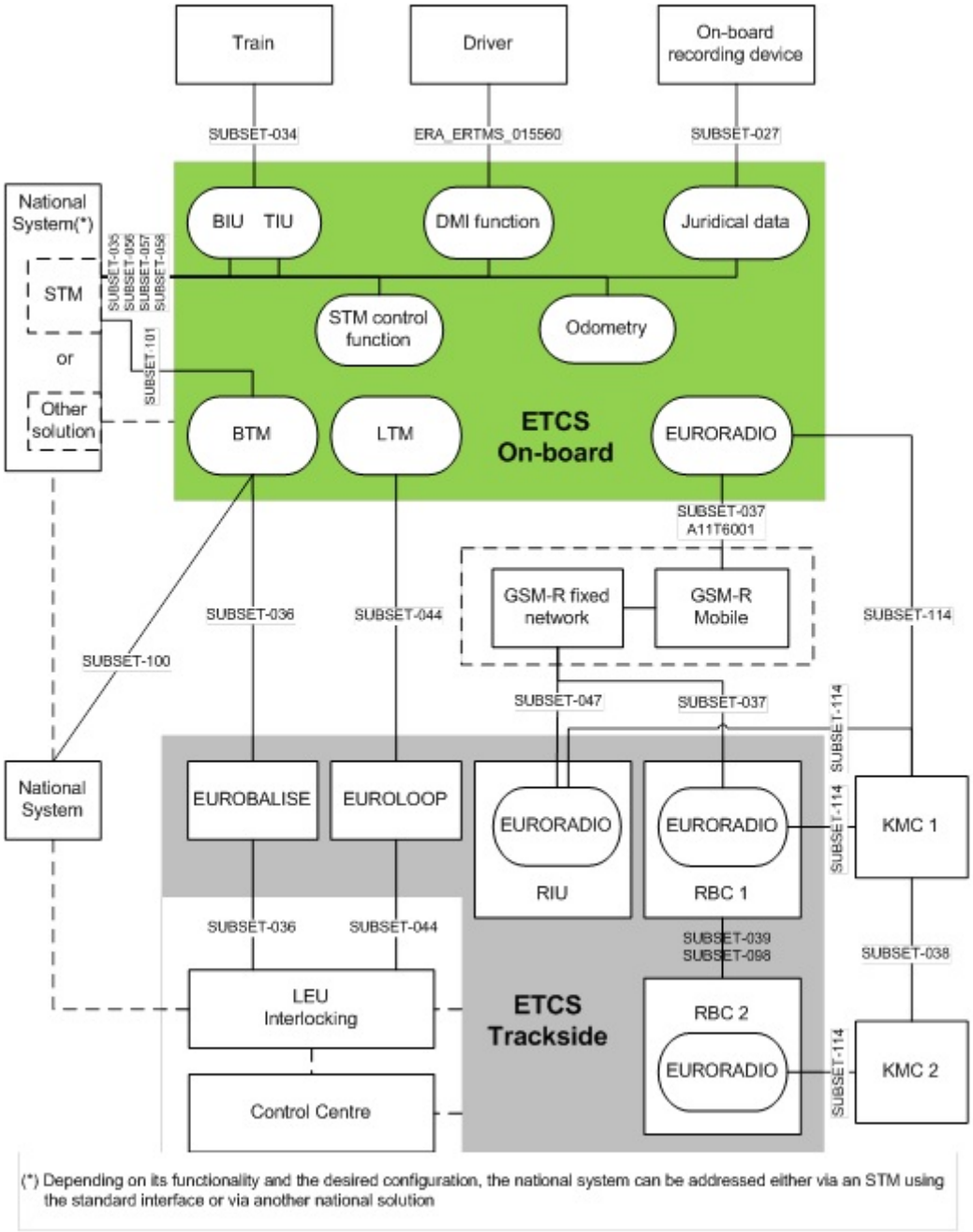


Figure 2. ERTMS/ETCS system and its interfaces (subset-026)

3.2 ETCS on-board part

The purpose of the openETCS project is to implement as open source the core of the ETCS on-board, also named EVC kernel. It does not cover the implementation of the following components, which are module resources used by the EVC kernel:

- Balise Transmission Module (BTM)
- Loop Transmission Module (LTM
- EuroRadio (RIM)
- Driver Machine Interface (DMI)
- Odometer (ODO)
- Train Interface Unit (TIU)
- Specific Transmission Module (STM), these are optional
- Juridical Recorder Unit (JRU)

For testing purpose this components may be simulated. However, the interface to communicate to them (API) shall be normalised, published and documented.

3.3 Principle of API

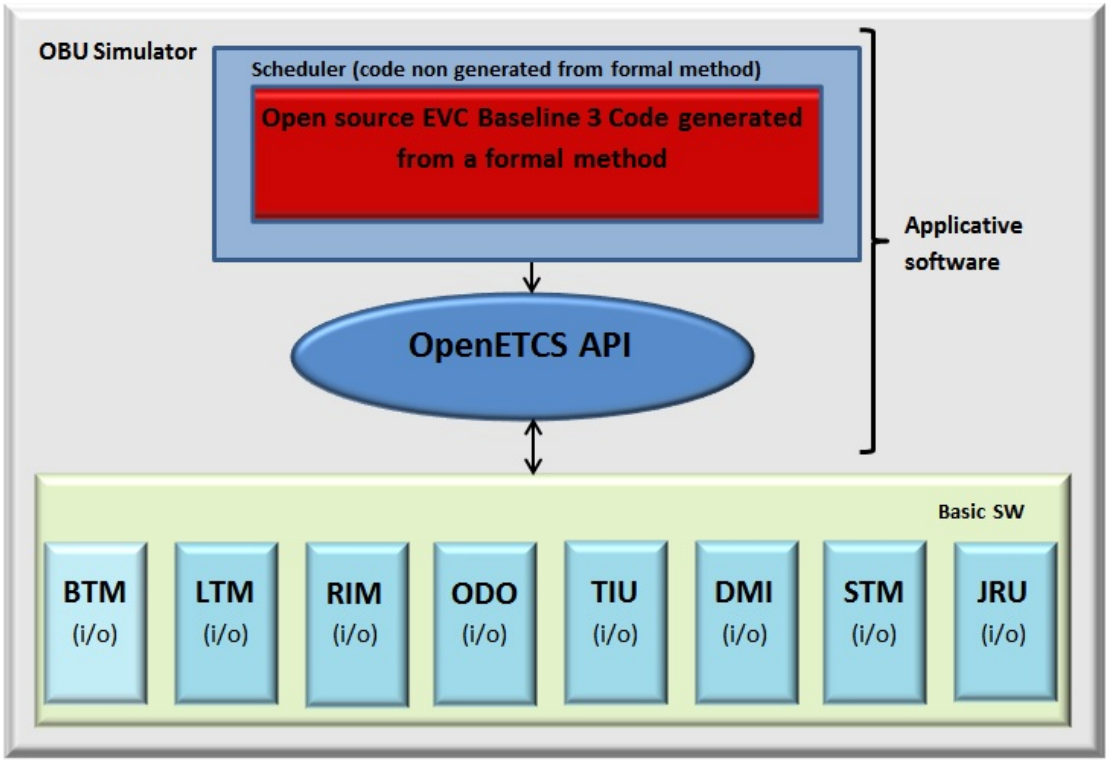


Figure 3. openETCS Application interfaces

4 Objectives of the demonstrator

4.1 Project aims

The Demonstrator provided by ERSA will be a hosting platform for the results produced by other WPs, especially the EVC code generated from formal methods by the tools chain (providers: WP3 for code and WP 7 for tools chain). The Demonstrator will be validated by using the EVC code for baseline 3 produced by ERSA as the first EVC application hosted in the Demonstrator. The EVC application developed by ERSA is a full functional implementation of subset-026 (SRS 3.3.0). The test environment allows playing scenarios provided by work package 4. EVC and test environment provided by work package 5 will help the sector for the creation, test and consolidation of the Test Sequences which are part of subset-076. Note subset-076 is not yet up to date but can be used to define guide lines. EVC code generated from formal methods will be produced in the project. This code shall replace the equivalent ERSA EVC code. Subfunctions and subcomponents will be integrated step by step in an iterative process. An integration step corresponds to a part of subset-026, for example decoding of Eurobalise's message or level transition management. The Demonstrator shall be used throughout the project and beyond the project as a test platform for the continuous integration of newly generated code.

Test environment shall communicate with EVC by an open logical interface (aim 1) for a full simulation. Later, test environment shall interact with hardware interfaces. These interfaces shall communicate with EVC input/output module in order to test code generated from formal method in a hardware context (aim 2).

4.1.1 Aim1 - Full simulation

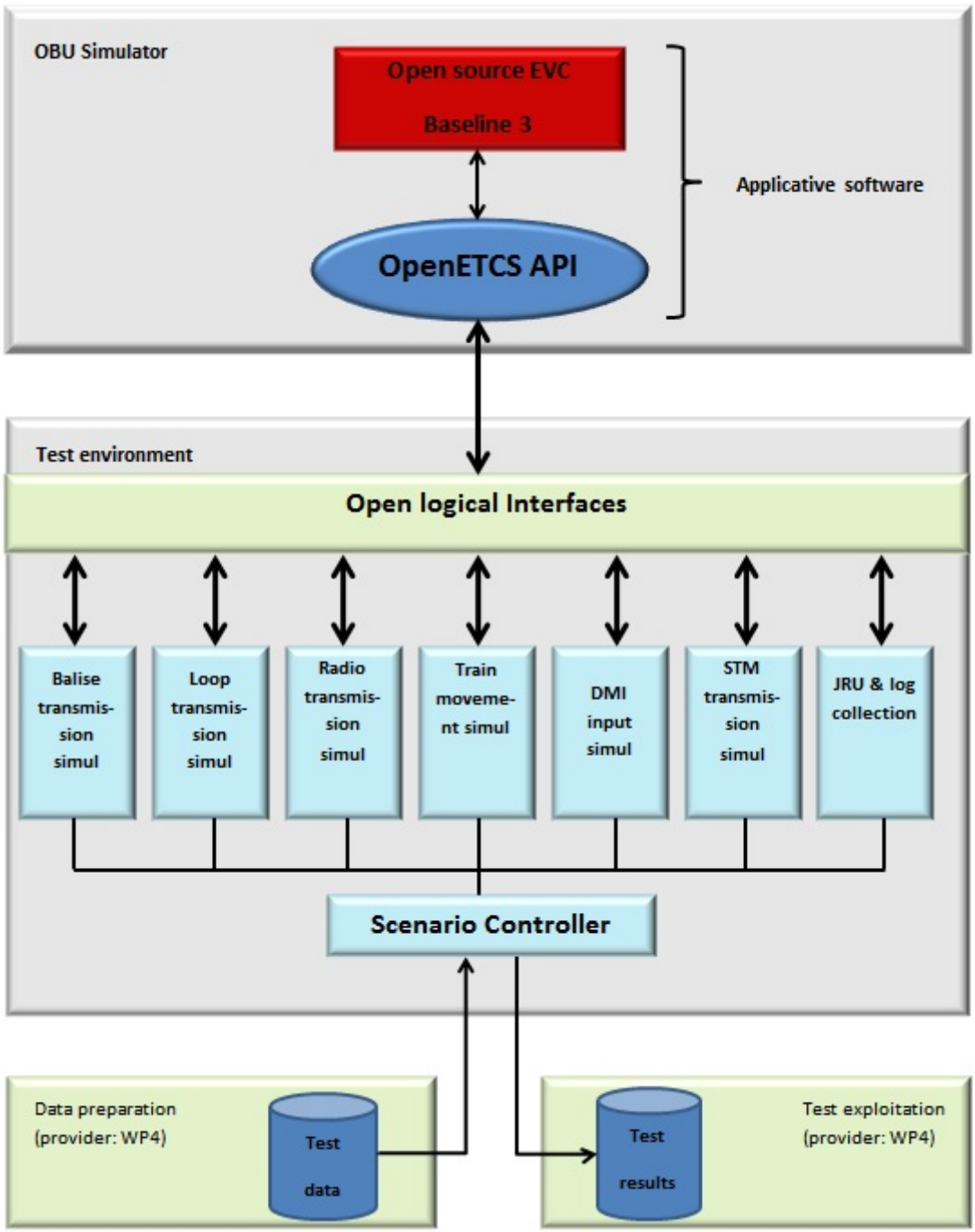


Figure 4. Project aim 1

4.1.2 Aim2 - Hardware simulation

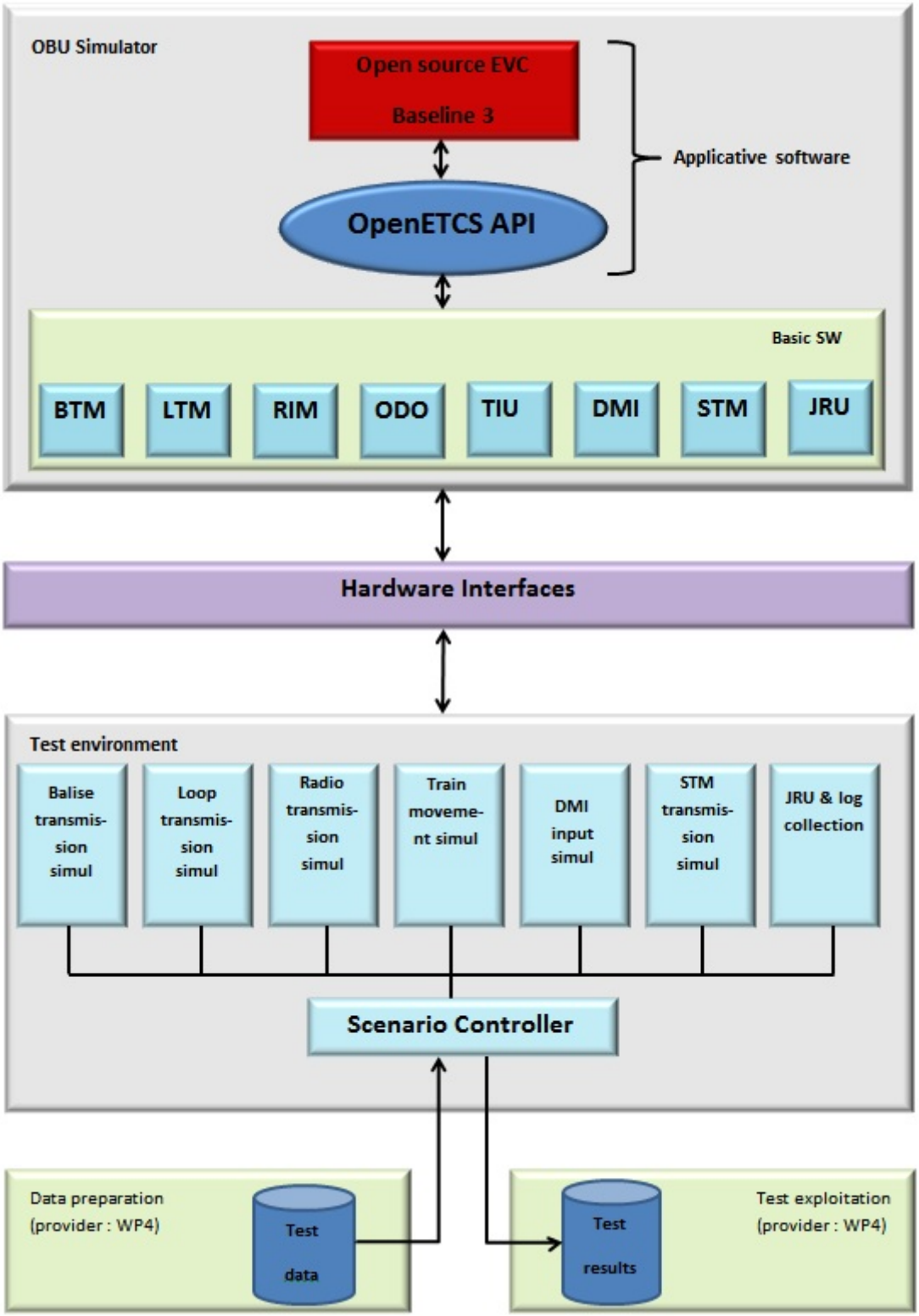


Figure 5. Project aim 2

4.1.3 Public interfaces overview (Aim1/Aim2)

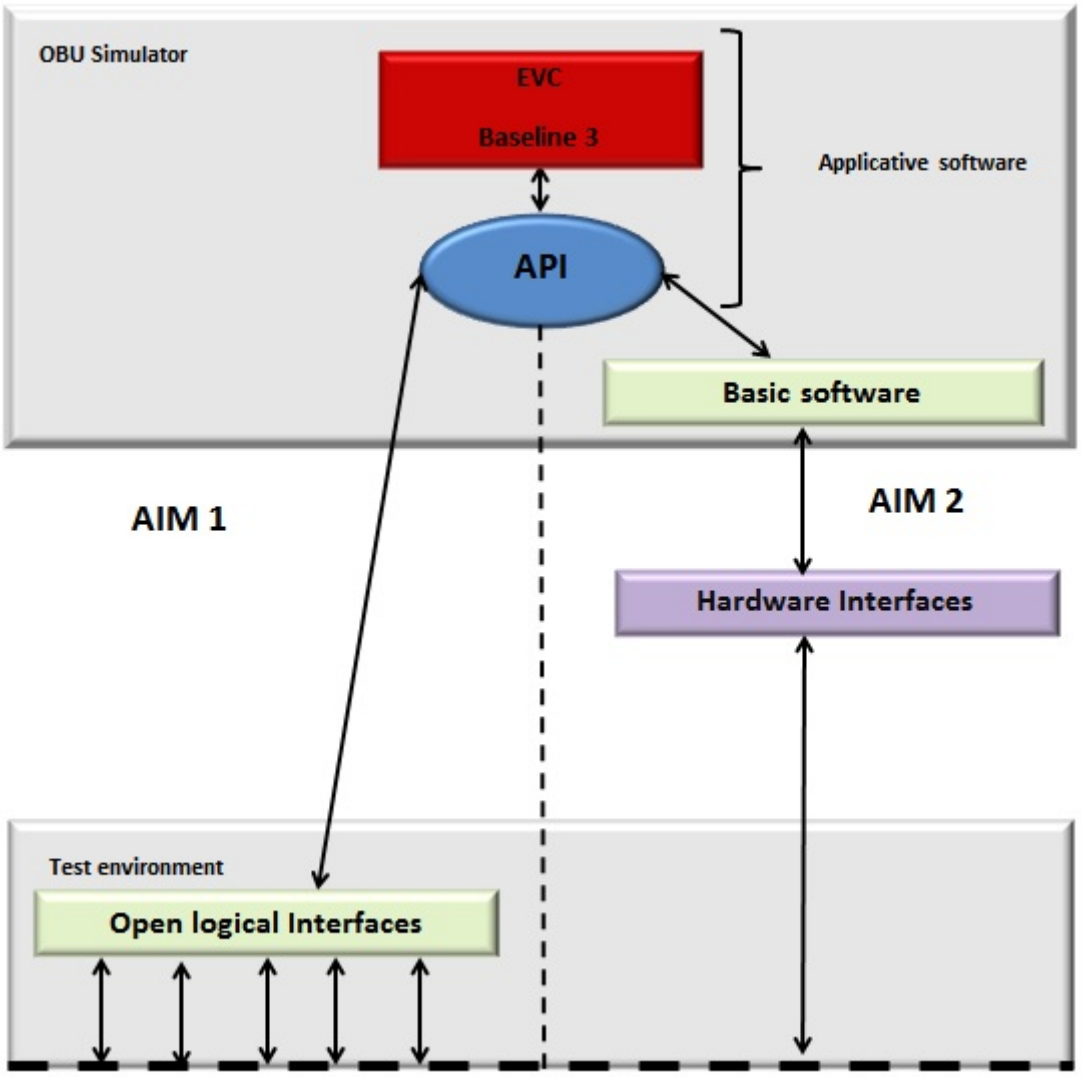


Figure 6. Interface aim 1 / aim 2

4.2 Work package 5 deliverables

4.2.1 On board unit simulator

Work package 5 shall deliver a dynamic library with communication modules able to interact with open logical interface (aim 1) and with hardware interfaces (aim 2).

4.2.2 Test environment

Work package 5 shall deliver an executable with simulator modules able to interact with open logical interface (aim 1) and in a second step with hardware interfaces (aim 2). The open logical interface can be considered as an adaptation layer between the API and the Test Environment when no hardware is present. This module is part of the Test Environment deliverable.

4.2.3 Test scenarios and reports

Test Environment will use test scenarios as input and produce output logs after execution of these scenarios. Inputs and outputs to/from have to be handled and analysed by work package 4.

4.3 Implementation steps

4.3.1 Principles

In a first step ERSA will use its own implementation of the EVC kernel baseline 3. At the moment SRS 3.3.0 is implemented. ERSA will use its API to interface with the hardware (PC based) and its resources and to communicate with the simulated on-board modules (BTM, RTM, TIU, ODO, DMI, etc...).

ERSA has developed its EVC kernel and API for baseline 3 in C language. The intention is to reuse as much as possible existing work and experience in order to achieve concrete results for the project in accordance with the FPP.

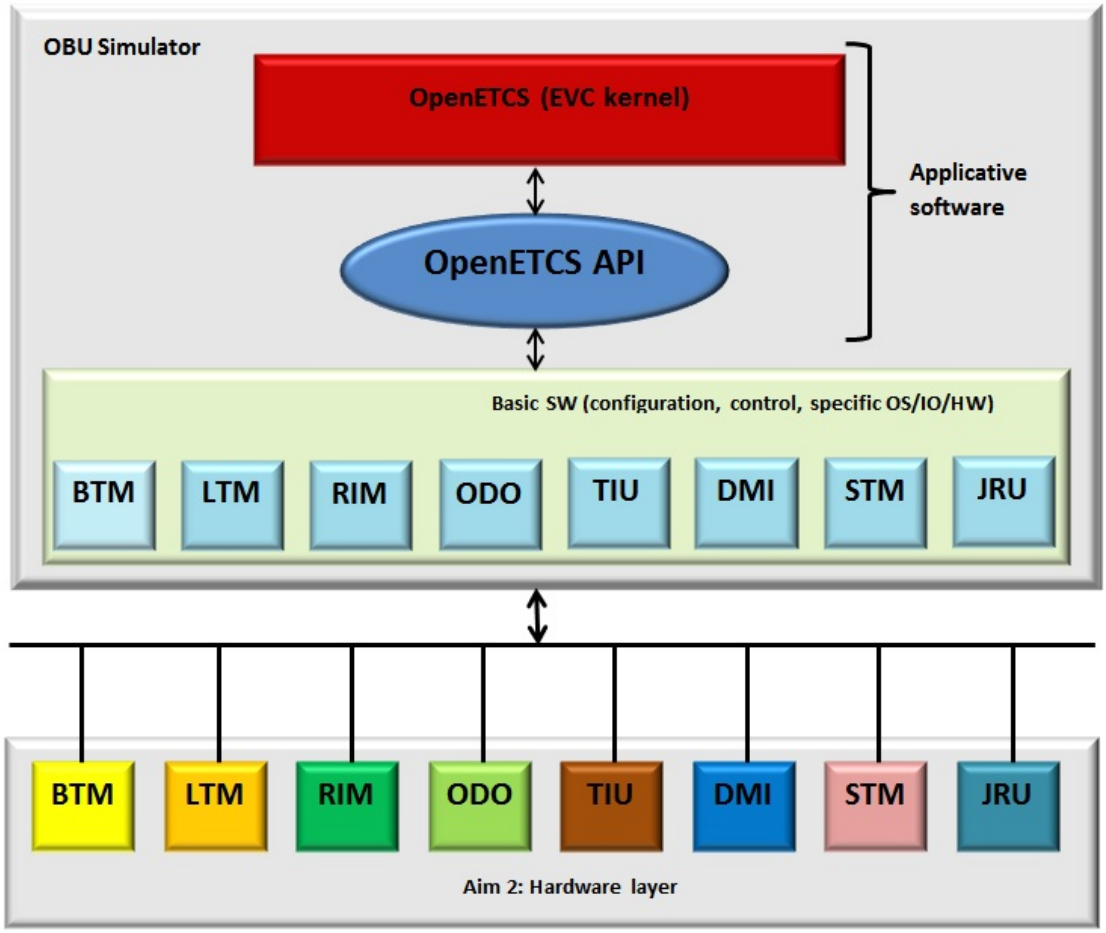


Figure 7. Use of API in the Demonstrator

4.3.2 Implementation step 1

There should be a high level (abstract) definition of the API, independent from any specific software language implementation. This abstract definition should then allow to develop an API, for all languages and hardware platforms of interest (ADA, C ...).

ERSA would then convert the abstract API into a specific dynamic library, usable within the Demonstrator. For example, the test environment application will use this dynamic library to access the EVC kernel. Using the test environment, further scenarios can then be created and executed to ensure a proper implementation of the API. The first implementation step helps to work without any hardware connected.

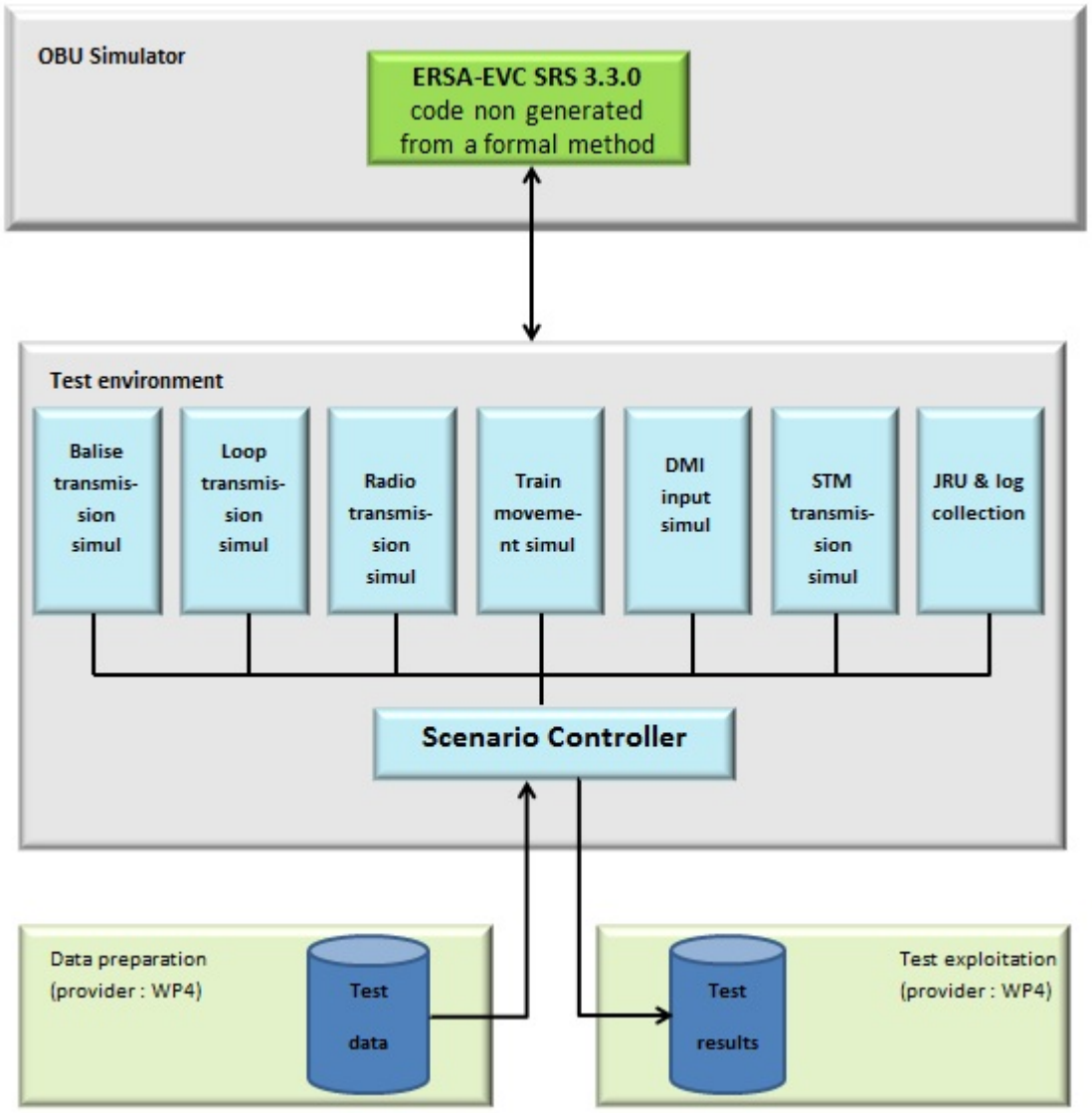


Figure 8. Implementation step 1

4.3.3 Implementation step 2

For this step, two options were identified. The first idea was to integrate code generated from formal methods in existing EVC code. For each WP3 delivery, new code generated from formal methods should have been isolated and merged with current code base. However, this way seems to be unrealistic and unsafe. It creates huge formal interfaces (between historical code and code produced from formal methods) which can introduce high-risk behaviour due to the current EVC architecture. A second option, identified to be the final choice in term of architecture, has purpose to reduce size and number of formal interfaces. A tasks scheduler manages EVC execution; moreover a limited interface allows communicating with code generated from formal methods. The interface between the scheduler and the hosted code shall be defined by both WP3 and WP5. The scheduler describes a set of states, within which standalone functions created by WP3 would be able to run. A WP3 delivery shall provide necessary code to complete one task of subset-026, like for example connect to RBC or compute breaking curves.

Option 1

Once code blocks generated from formal method are available, they should be integrated progressively within the non-formal EVC kernel code. Modification of existing code may be needed to achieve the process, so each integration should be done with a validation using the test environment.

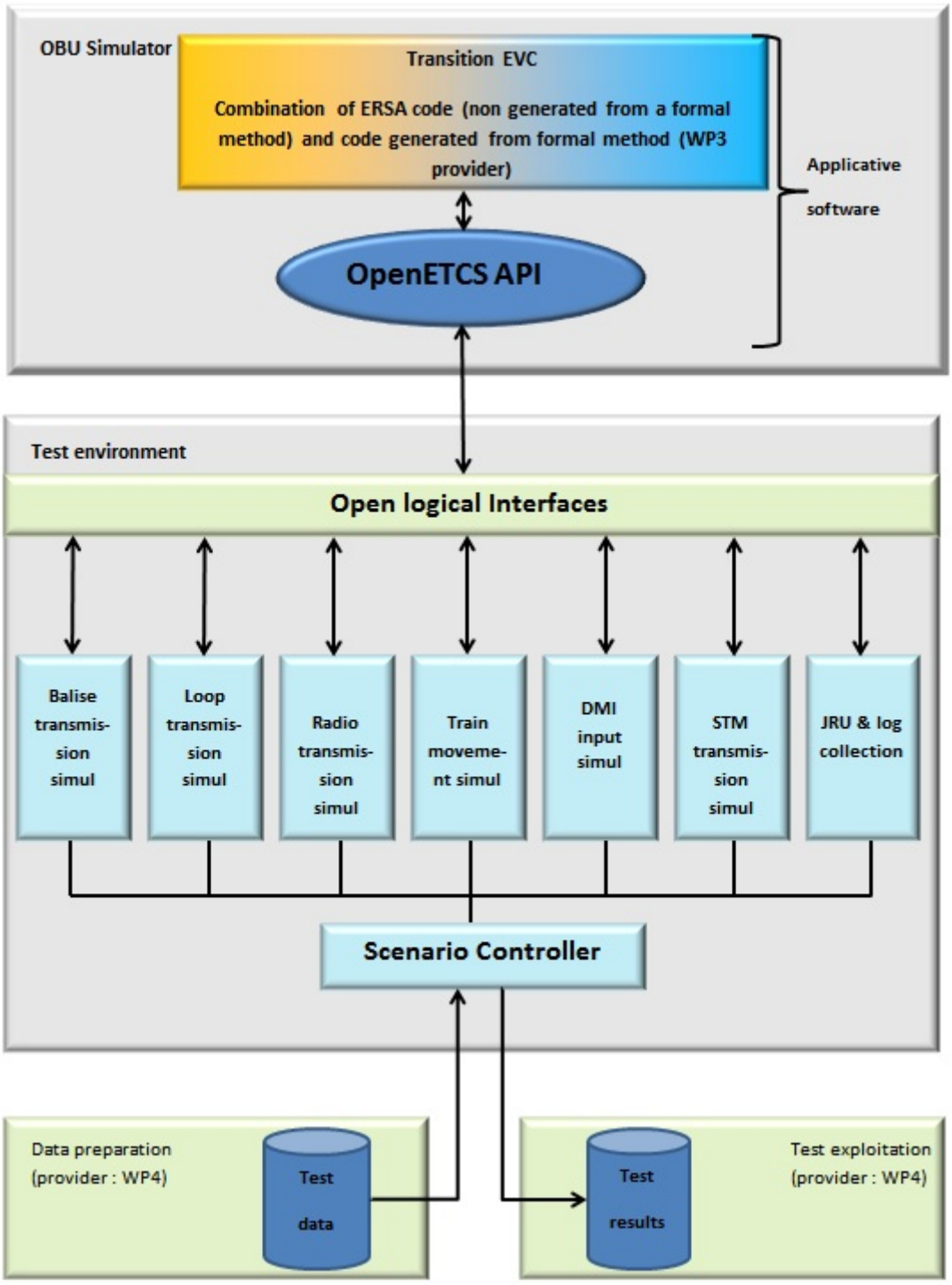


Figure 9. Implementation step 2 option 1

Option 2

There is no difference for test environment part and open logical interface regarding option 1.

Formal interfaces shall be defined by WP3 and WP5 in order to use only one interface between provided scheduler and code generated from formal method provided by WP3. OpenETCS-EVC kernel covers partially subset-026. A step of integration matches to a complete task defined in subset-026 (curves computations, Eurobalise management, RBC management...).

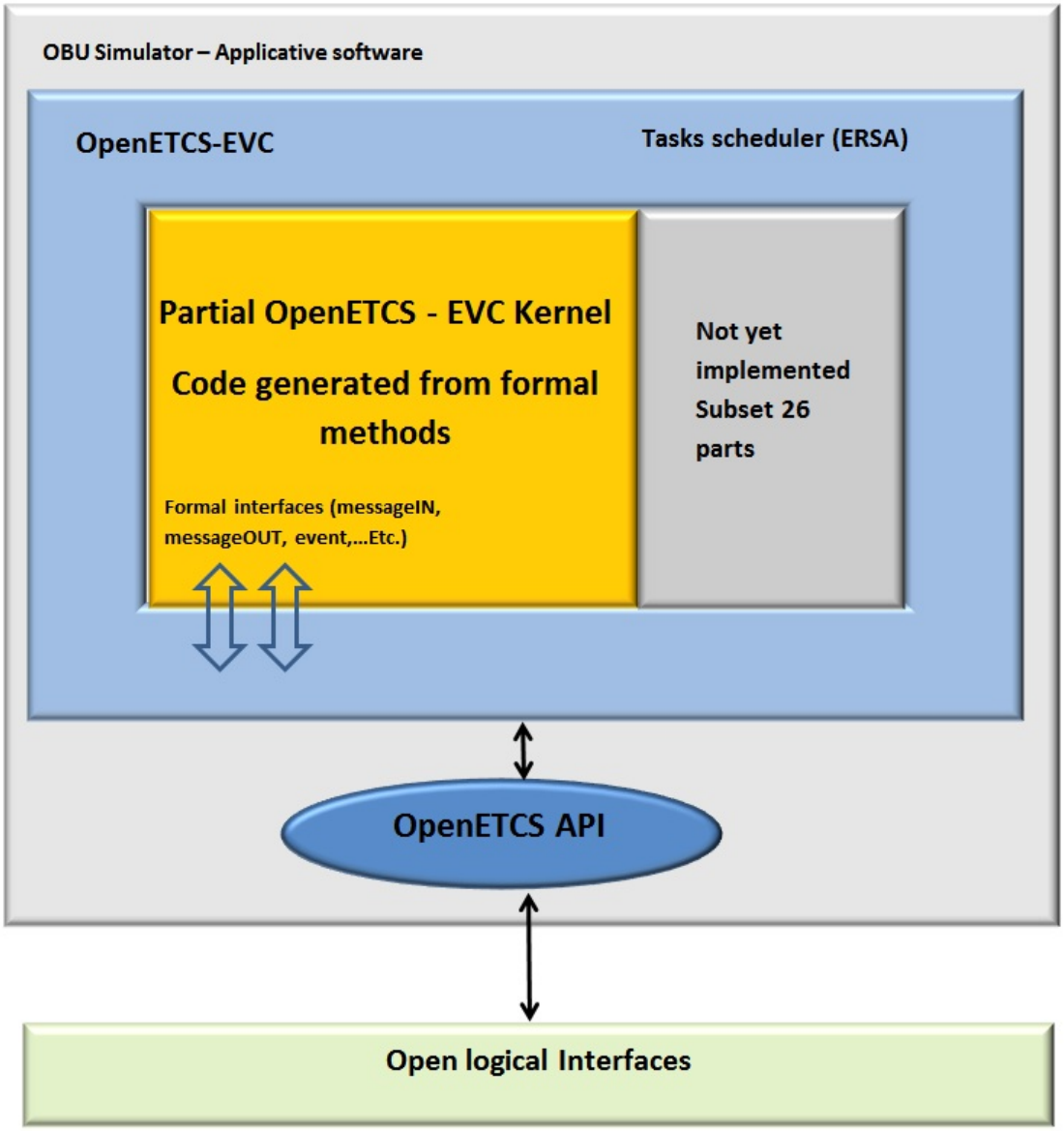


Figure 10. Implementation step 2 option 2

4.3.4 Implementation step 3 (outside project)

Option 1

At the end of the process, it is expected that all EVC kernel code will be generated from the openETCS tool chain. Finally the openETCS EVC kernel shall replace the ERSA EVC kernel. However, regarding time plans and resources available, the aim will most likely be reached after the openETCS project.

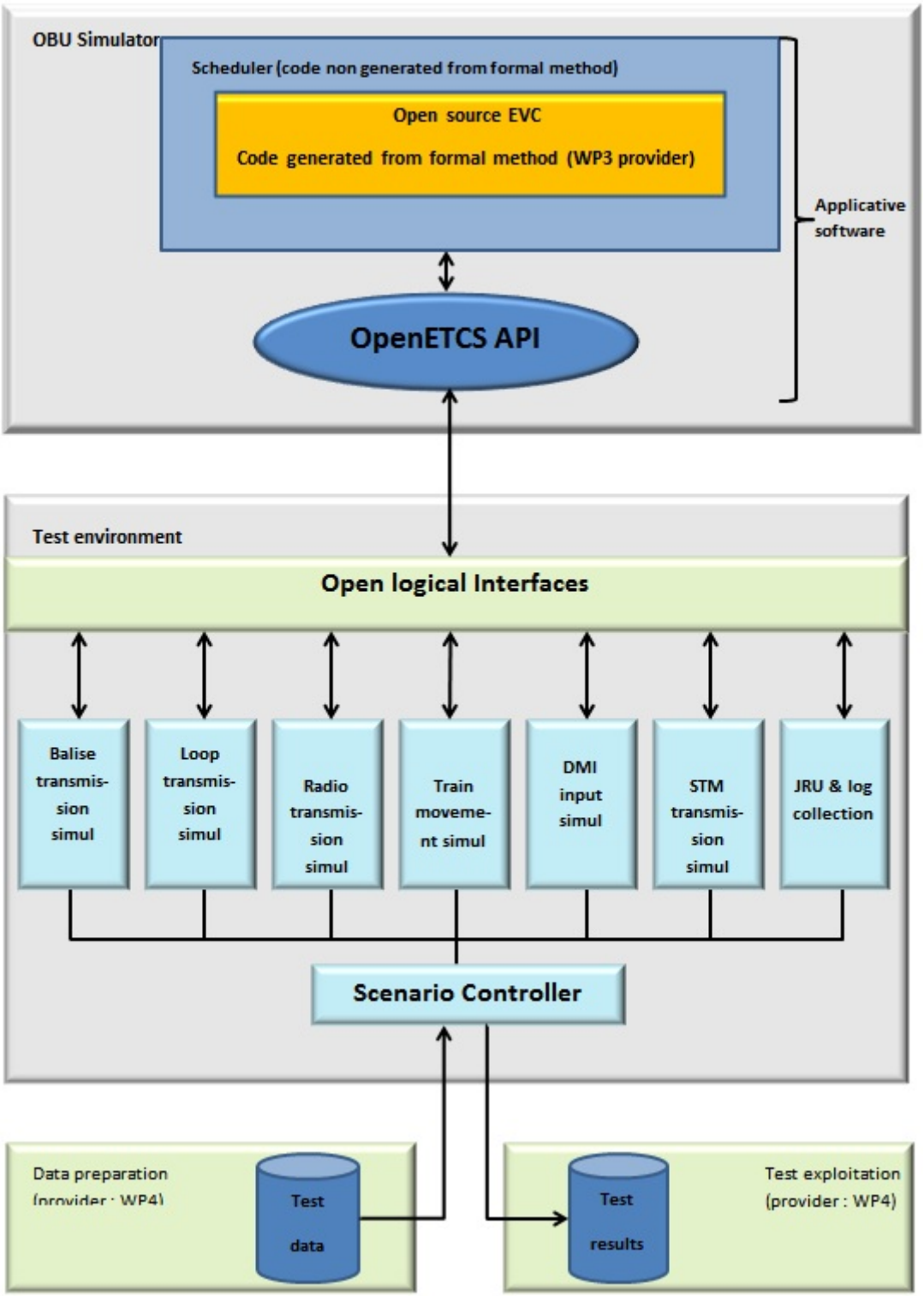


Figure 11. Implementation step 3 option 1

Option 2

There is no difference for test environment part and open logical interface regarding option 1. Formal interfaces shall be defined by both WP3 and WP5 in order to use only one interface between provided scheduler and code generated from formal method provide by WP3. OpenETCS-EVC

kernel covers totally subset-026.

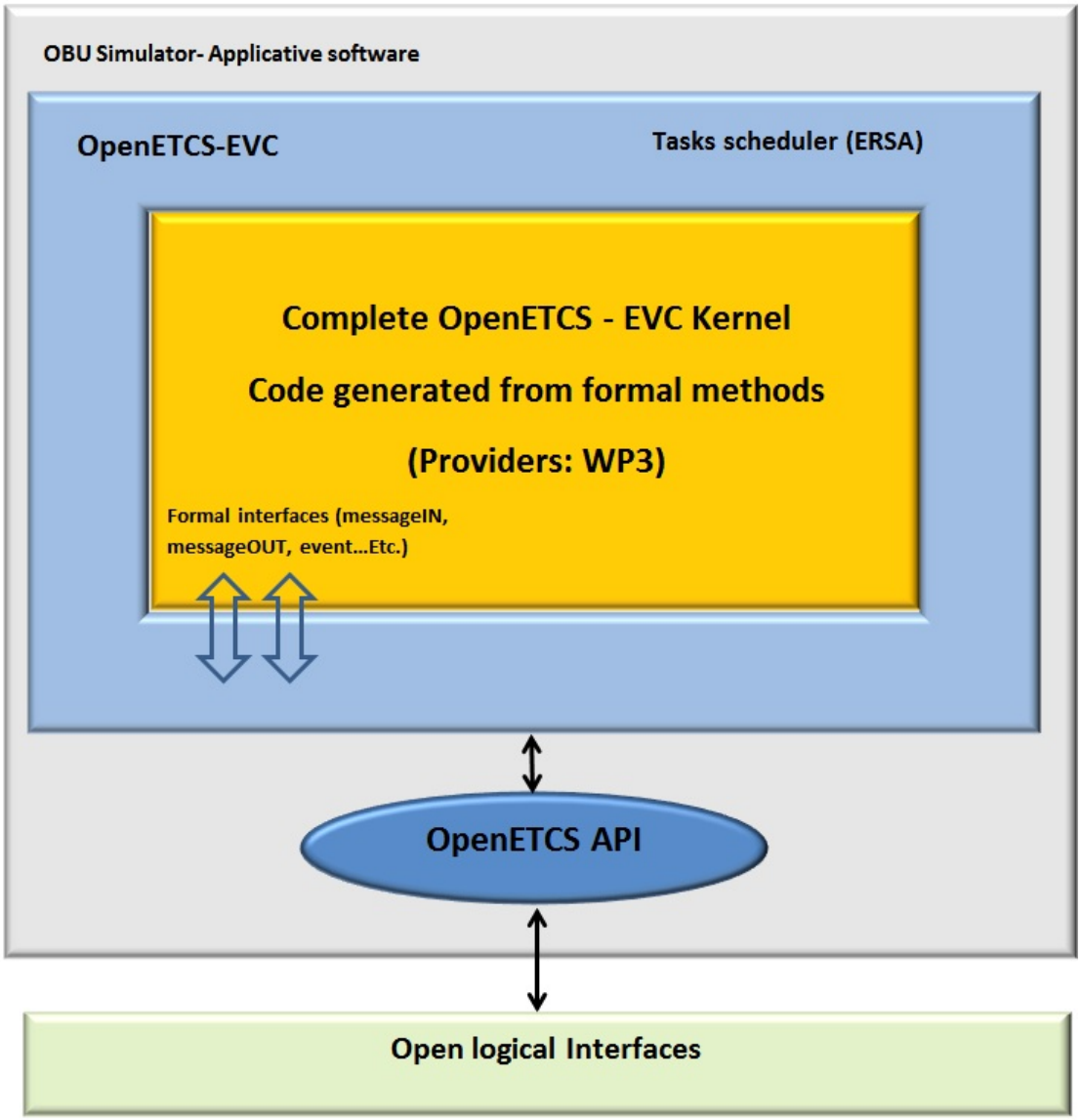


Figure 12. Implementation step 3 option 2

4.3.5 Implementation step 2A

While integrating code produced from formal methods, a reference non-vital platform shall be provided using industrial hardware. This platform shall host the EVC kernel and its modules, the communication between both being performed through the API defined in an earlier step.

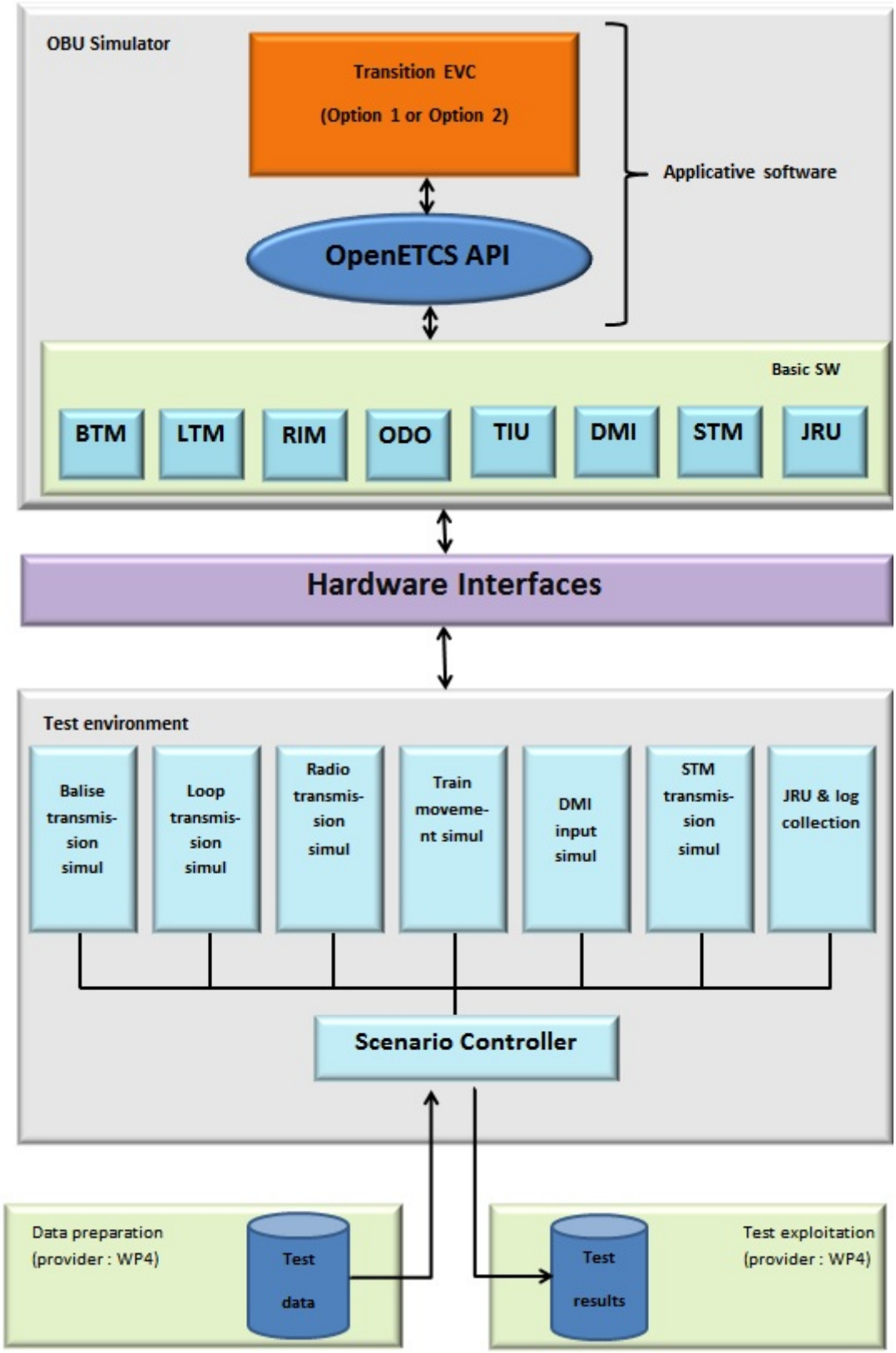


Figure 13. Implementation step 2A

The Demonstrator shall include two main parts: the on-board part and the test environment.

5 Architecture of the demonstrator

5.1 On board part

The on-board part includes the EVC kernel and all its modules. The EVC kernel shall be clearly separated from the modules and shall communicate with them via the API. The following modules will be simulated:

- DMI function
- BTM
- LTM (option)
- Euroradio
- BIU & TIU
- Odometry
- STM control function (option)
- Juridical

The on-board part will be installed either on a PC or on a platform.

5.2 Environment part

The environment part includes an off line part and an on line part. The off line part includes (provider WP4):

- Tools for data preparation; they are used for the creation of test scenarios
- Tools for data exploitation.

The on line part includes (provider WP5):

- A set of tools for execution of scenarios.

The execution of a scenario consists of the stimulation of the on-board modules through standard interfaces. The environment will be installed on a PC.

6 Test Environment for the ON-BOARD

6.1 Off line tools

6.2 Data preparation

A template for test scenarios shall be created. This can be based on a combination of experience and the result of other projects.

At the end the data collected from projects shall be converted into that template. The data shall be stored in a database.

6.3 Data exploitation

During execution of test scenarios, the on-board part will generate data to be stored in the JRU. At the same time, the scenario execution tools will also generate log data. Both types of data shall be stored in a database.

The data exploitation shall consist of the analysis of such stored data. This analysis shall include criteria for defining whether tests are passed and shall define a template for test reports.

6.4 Execution tools environment overview

Test environment shall:

- Load scenario
- Play a scenario
- Export JRU data when simulation ended
- Evaluate intermediate conditions (basd on speed, position, TIU values) during simulation
- Communicate with open logical interface in aim 1 through its simulation modules
- Communicate with hardware interfaces in aim 2 through its simulation modules
- Interact with on board unit simulator if user wants to test only one EVC communication module and not the others
- Evaluate the final result for a scenario (SUCCESS or FAILED)
- Translate all events in scenario into simulation event with its simulation modules and send them to open logical interface (Aim 1) or Hardware interfaces (Aim 2)

Scenario shall describe :

- Eurobalise positions on the tracks
- Eurobalise message on the tracks
- Euroloop data
- Radio Infill Unit data

- All driver actions supported by test environment, like
  - level transition acknowledgement
  - mode transition acknowledgement
  - service brake or emergency acknowledgement
  - change level
  - open cabin
  - set driver Id
  - set train data
  - TIU action (sleeping mode, non leading mode...)
- Train speed profile
- Gradient profile
- Radio messages coming from RBCs
- Intermediate prioprietary conditions based on speed, position, radio message send by EVC...etc. These are user definable check points to ensure critical milestones verification during scenario play.
- Initial train configuration
- Initial national values
- Initial EVC state
- Initial RBCs state

Simulation module shall be used to:

- Interact with the matching communication module in basic software through open logical interface (aim 1) or hardware interface (aim 2)
- Transmit a Eurobalise message to the on board part
- Transmit a loop message to the on board part (option)
- Exchange radio telegrams with the on board part
- Exchange STM messages with the on board part (option)
- Send odometry information to the on board part. In order to be able to simulate all situations, no real train movement simulation is performed and it is not affected by the brake request from the on-board. It only follows a predefined speed profile defined in the scenario.
- Exchange TIU & BIU information with the on board part
- Exchange DMI information with the on board part
- Receive JRU information from the on board part

7 Abstract interface description

The following data should be exchanged between the EVC kernel and its environment through the API. In this description, the data exchanged will be described from the EVC point of view. Its means that:

- Input is data provided by a module to the EVC
- Output is data sent by the EVC to a module

7.1 Configuration

Some data are necessary to initialise and configure the EVC kernel:

- ETCS Identity
- Available train equipment (radio equipment, brake systems, traction systems, STMs, ...)
- Braking parameters and data
- On-board correction factors
- National values
- Default data to use when the EVC memory is lost
- EVC memory (stored data, ...)

This preliminary list shall be refined.

7.2 Odometry interface

As the implementation of Odometer functionalities is out of scope of openETCS, only relevant data exchanged with the Odometry functions are described.

7.2.1 Inputs :

- Time and movement data
  - Reference time
  - Train movement direction (Nominal, Reverse, Unknown)
  - Estimated train position
  - Position accuracy (over- and underestimation)
  - Estimated speed
  - Speed accuracy (over- and underestimation)
  - Estimated acceleration
- Odometer status

7.2.2 Outputs :

- Optionally, recalibration data.

This data shall be confirmed.

7.3 Brake/Train interface

The inputs and outputs with the train are described in the subset-034 (see /4/).

7.3.1 Inputs :

- ETCS Main Switch status
- Cab status
- Direction controller
- Train Integrity status
- Sleeping Permission
- Non Leading Permission
- Passive Shunting Permission
- Regenerative Brake status
- Magnetic Shoe Brake status
- Eddy Current Brake status
- Electro Pneumatic (EP) status
- Additional brake status
- Train data information
- Type of train data entry
- Traction status
- National System Isolation status
- Brake pressure

7.3.2 Outputs :

- Service Brake command
- Emergency Brake command
- Traction Cut-Off command

- Regenerative Brake inhibition
- Magnetic Shoe Brake inhibition
- Eddy Current Brake for SB inhibition
- Eddy Current Brake for EB inhibition
- Change of traction system
- Pantograph command
- Air Tightness command
- Main Power Switch command (sometimes named main circuit breaker)
- Isolation status
- Station location
- Allowed current consumption

7.4 Balise interface

As the implementation of BTM functionalities is out of scope of openETCS, only relevant data exchanged with the BTM are described.

7.4.1 Inputs :

- Reception of balise message:
  - Odometer stamp of the centre of the balise and accuracy
  - Time stamp of reception
  - Balise message (according to paragraph 7 and 8 of subset-026)
- BTM status information
  - Failure, alarm

7.4.2 Outputs :

None

7.5 Loop interface (OPTION)

As the implementation of LTM functionalities is out of scope of openETCS, only relevant data exchanged with the LTM are described.

7.5.1 Inputs :

- Reception of loop message
  - Time stamp of reception
  - Loop message (according to paragraph 7 and 8 of subset-026)
- LTM status information
  - Failure, alarm

7.5.2 Outputs :

- Configuration of loop antenna (spread spectrum code)

7.6 Radio interface

As the implementation of EuroRadio functionalities is out of scope of openETCS, only relevant data exchanged with the EuroRadio subsystem are described.

The management of radio network / safe connection and exchange of radio messages is performed via service primitives as described in subset-037 (see /7/).

7.6.1 Service primitives

- Safe connection set-up
  - Request
  - Indication
  - Response
  - Confirmation
- Data transfer
  - Request
  - Indication
- Connection release
  - Request
  - Indication
- Error reporting
  - Indication
- High priority data
  - Request
  - Indication

- Network registration
  - Request
  - Indication
- Network permission
  - Request
  - Indication

7.7 STM interface (OPTION)

The data (STM messages) exchanged with the National Systems are described in subset-035 (see /5/) and subset-058 (see /9/).

7.8 JRU interface

The recording of juridical data is deeply linked to the ETCS core functionalities and should be part of the openETCS application

7.8.1 Inputs :

None

7.8.2 Outputs :

All juridical information described in subset-027 (see /3/) should be transmitted to the Recording Unit:

- General message
- Train data
- Emergency brake command state
- Service brake command state
- Message to radio infill unit
- Telegram from balise
- Message from Euroloop
- Message from radio infill unit
- Message from RBC
- Message to RBC
- DriverŠs actions
- Balise group error
- Radio error

- STM information
- Information from cold movement detector
- Start displaying fixed text message
- Stop displaying fixed text message
- Start displaying plain text message
- Stop displaying plain text message
- Speed and distance monitoring information
- DMI symbol status
- DMI sound status
- DMI system status message
- Additional data
- SR speed/distance entered by the driver
- NTC selected
- Safety critical fault in mode SL, NL or PS
- Virtual balise cover set by the driver
- Virtual balise cover removed by the driver
- Sleeping input
- Passive shunting input
- Non leading input
- Regenerative brake status
- Magnetic shoe brake status
- Eddy current brake status
- Electro pneumatic brake status
- Additional brake status
- Cab status
- Direction controller position
- Traction status
- Type of train data
- National system isolation
- Traction cut off command state
- ETCS on-board proprietary juridical data

7.9 Driver interface

As the implementation of DMI functionalities is out of scope of openETCS, only relevant data exchanged with the DMI will be described. The description of the data flow should cover the functionalities related to the data displayed and driver actions on DMI as described in ERA_ERTMS_015560 (see /2/).

7.9.1 Inputs :

- Activity status of the DMI
- Driver Request or Acknowledgement (other than text)
- Text Message Acknowledgment
- Train Data Validation
- Version information of the DMI
- Icon Acknowledgment
- Indication of audible information on DMI
- Set Virtual Balise Cover
- Remove Virtual Balise Cover
- Entered radio network
- VBC data (set) validation
- VBC data (remove) validation
- Output information related to NTC

7.9.2 Outputs :

- Dynamic Data, like current train speed, target dataĚ
- Request to enable/disable driver menus and buttons
- Request to input certain data (driver id, train dataĚ)
- EVC Coded Train Data to be validated by EVC
- Predefined or Plain Text Message
- Description of track (speed and gradient profileĚ)
- Request for the DMI version information
- Request to display one or more icon(s) in any area
- Display the EVC operated system version
- State of DMI display (cabin activation)

- List of available levels
- List of available radio network
- List of VBCs stored on-board
- Coded VBC data (set) to be validated by driver
- Coded VBC data (remove) to be validated by driver
- Input information related to NTC
- Description of NTC data entry window

7.9.3 Two-way packets

- Default or Entered Driver Identifier
- Default or Entered Train Running Number
- Default or Entered Staff Responsible Data
- Default or Entered Train Data
- Default or Entered Adhesion Factor Data
- Default or entered ETCS Level
- Default or entered RBC contact info (RBC data and radio network ID)

This list shall be refined.

8 Test

8.1 Input (Work package 4)

Track and train data shall be collected from concrete projects.

8.2 Test case creation (Work package 4)

Templates for test cases and/or test scenarios shall be defined. The input data shall be converted into test scenarios. A required test case will be to simulate the Utrecht-Amsterdam track as it is relevant for openETCS simulation.

8.3 Test scenario importation (Work package 5)

Test scenarios shall be imported in the test execution environment.

8.4 Test execution (Work package 5)

Test scenarios shall be executed, and the results shall be stored. Stored information will include data provided by the EVC kernel (JRU files) and data provided by the test tools.

8.5 Test results exportation (Work package 5)

The results of the executed tests shall be exported to templates to be defined in the project.

8.6 Test results ecploitation (Work package 4)

The results shall be analysed and a report shall be generated. A template for the reports has to be defined in the project.

9 Creation of non vital platform

9.1 Input

The NON VITAL platform shall be provided by a project partner. It shall be delivered with all means to enable the implementation of software:

- Development tools for C language and support to their use,
- Module on-board modules, APIs and drivers.

9.2 Software implementation

The EVC software which shall be installed on this platform cannot be defined at this stage. It is likely that a hybrid version of the EVC kernel based on ERSA code and project code shall be implemented in the platform.

9.3 Availability of physical interface

The platform shall be equipped with physical interfaces, to enable their stimulation via ERTMS test tools owned by ERTMS Test Laboratories and/or ERSA. It is not planned to create a complete Test Environment from scratch.

10 Tests with non vital platform

10.1 Test case creation (Work package 4)

The test data produced in the first part of the project (see chapter 8) shall be reused for performing tests. Only a subset shall be used.

10.2 Test execution (Work package 5)

Tests shall be performed and results stored in a database. The same principles as defined in chapter 8 shall be reused.

10.3 Test results analysis (Work package 4)

The test data shall be analysed and conclusions shall be drawn. The templates defined in chapter 8 shall be reused. Only a subset will be needed.