OETCS/WP3/D3.5.2

openETCS

ITEA2 Project
Call 6 11025
2012 – 2015

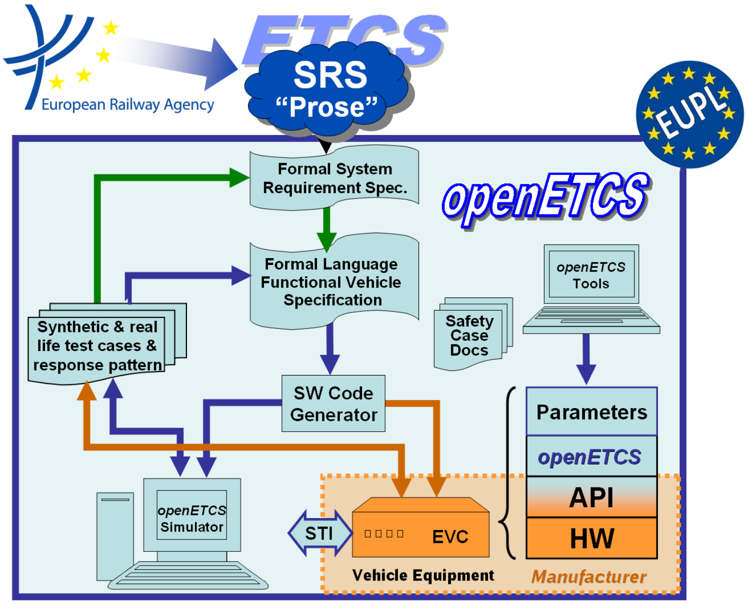
Work Package 3: “Modeling”

openETCS Design Specification

Software Component Design and Internal Interface Specification

Baseliyos Jacob, Peter Mahlmann, Bernd Hekele, Peyman Farhangi, Stefan Karg, Valerio D´Angelo, Uwe Steinke, Christian Stahl, Jakob Gärtner, Jos Holtzer, Jan Welvaarts, Vincent Nuhaan, Benjamin Beichler, Thorsten Schulz, Marielle Petit-Doche, Matthias Gudemann, Vernique Gontier, Ghristian Giraud, Fausto Cochetti and Alexander Stante

June 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/D3.5.2
June 2015

openETCS Design Specification

Software Component Design and Internal Interface Specification

Document approbation

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

Baseliyos Jacob, Peter Mahlmann, Bernd Hekele, Peyman Farhangi, Stefan Karg, Valerio D´Angelo

DB Netz AG

Uwe Steinke

Siemens AG

Christian Stahl

TWT-GmbH

Jakob Gärtner

LEA Railergy

Jos Holtzer, Jan Welvaarts, Vincent Nuhaan

Nederlands Spoorwegen

Benjamin Beichler, Thorsten Schulz

University of Rostock

Marielle Petit-Doche, Matthias Gudemann

Systerel

Vernique Gontier

All4Tec

Ghristian Giraud, Fausto Cochetti

Alstom

Alexander Stante

Fraunhofer ESK

Architecture and Design Specification

Prepared for openETCS@ITEA2 Project

Abstract: This document gives an introduction to the software and component design of the openETCS OBU model. 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 (EUPL v.1.1+) AND Creative Commons Attribution-ShareAlike 3.0 – (cc by-sa 3.0)

Modification History

Version	Section	Modification / Description	Author	Date
0.1	Document	Initial document providing structure	Peter Mahlmann	27.05.2015
0.2	2	New template for design descriptions	Peter Mahlmann	10.06.2015

Table of Contents

Figures and Tables

Figures

Tables

Part I

Functional Breakdown

1 openETCS API Runtime System and Input to the EVC)

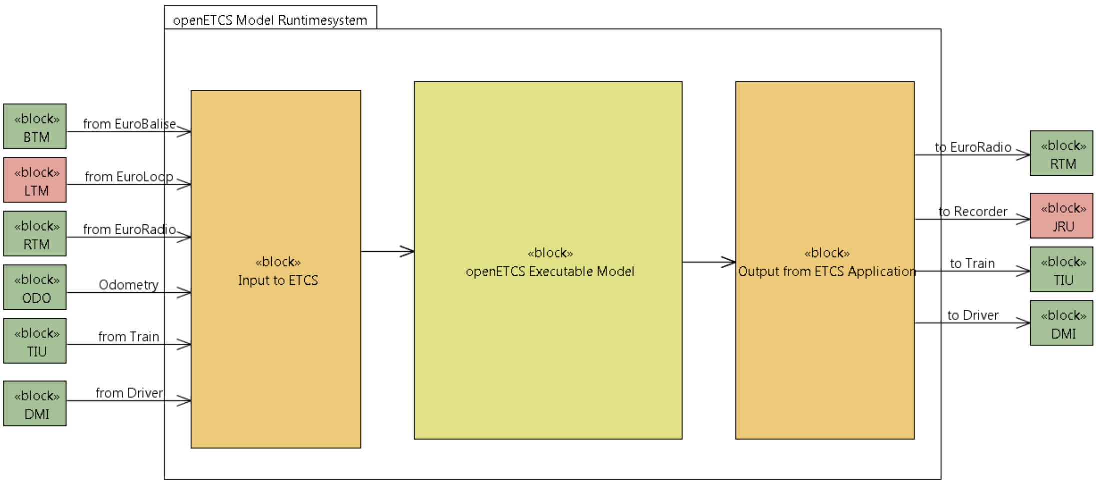


Figure 1. openETCS API Highlevel View

Figure 1 shows the structure of API with respect to the software architecture. Note that red input and output modules are were not yet implemented and thus are not part of the openETCS OBU model. 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.

1.1 Principles for Interfaces (openETCS API)

Information is exchanged via asynchronous *messages*. A message is a set of information corresponding to an event of a particular unit, e.g. a balise message received from the BTM. For possible types of messages please refer to 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.

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 Ouput 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 dictonary.

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 SRS and passed to the buffers to the units.

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
BTM	Balise Telegram	Receive Messages
DMI	Driver Machine Interface	DMI Manager
EURORADIO	Communication Management	Communication Management
EURORADIO	Radio Messages	Receive Messages
ODO	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.

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

55 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.

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.

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 bounderies 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 bounderies 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.

1.7 Interfaces to the Train Interfaces (TIU)

The following infomration 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.

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	DMI	
	Train Data	TIU	

Packets: to be completed

Radio Messages to be completed

Part II

Design Description

2 F1: Receive information from Trackside

3 F2: ETCS Kernel

3.1 Manage_TrackSideInformation_Integration

125 3.1.1 Component Requirements

Component name	Manage_TrackSideInformation_Integration
Link to SCADE model	???
SCADE designer	[Name, affiliation]
Description	The block "Manage_TrackSideInformation_Integration" is responsible for receiving Eurobalise telegrams and Euroradio messages from the API and performs several consistency checks on the inputs. The block collects the telegrams of balises in order to build balise group messages. Euroradio messages are always delivered as a whole message. On each message, a consistency check is performed, before the data is validated according to the driving direction of the train. In general, messages not designated for the current driving direction of the train are not forwarded to the further processing. After applying consistency checks, the data direction is validated.
Input documents	See sub-components.
Safety integrity level	4
Time constraints	n/a
API requirements	n/a

3.1.2 Interface

An overview of the interface of component Manage_TrackSideInformation_Integration is shown in Figure 2. The inputs and outputs are described in detail in Section 3.1.2.1 respectively 3.1.2.2.

3.1.2.1 Inputs

130 3.1.2.1.1 fullChecks

Input name	fullChecks
Description	Indicates, if all checks on the message should be performed.
Source	Configuration
Type	bool

Valid range of values	**true** All checks are performed. **false** Component InformationFilter is deactivated.
Behaviour when value is at boundary	n/a
Behaviour for values out of valid range	n/a

3.1.2.1.2 API_trackSide_Message

Input name	API_trackSide_Message
Description	Track side message received from the API. The API performs pre-processing of RTM and BTM messages and deliveres a maximum of a single message per cycle.
Source	API
Type	API_Msg_Pkg::API_TrackSideInput_T
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.1.2.1.3 ActualOdometry

Input name	ActualOdometry
Description	Provided by the external odometry module of the train. It contains relative location information with inaccuracies.
Source	Odometer
Type	Obu_BasicTypes_Pkg::odometry_T
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

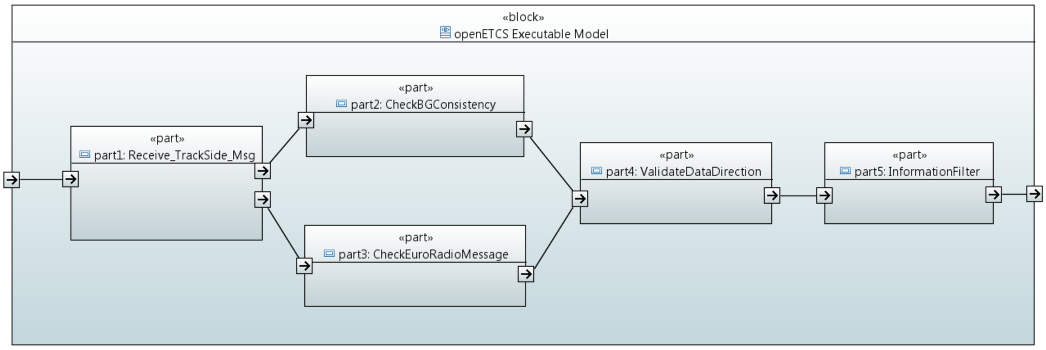

«block»
openETCS Executable Model
«part»
part1: Receive_TrackSide_Msg
«part»
part2: CheckBGConsistency
«part»
part3: CheckEuroRadioMessage
«part»
part4: ValidateDataDirection
«part»
part5: InformationFilter

Figure 2. SysML diagram of Manage_TrackSideInformation_Integration module.

3.1.2.1.4 reset

Input name	reset
Description	To delete all data stored in the module (e.g. collected balise telegrams, which do not yet form a complete message), a reset input can be used. If the input is set to true, all data kept in the module is deleted and no input is accepted.
Source	Environment
Type	bool
Valid range of values	**true** All data kept in the module is deleted and no input is accepted. **false** No action. Data at input is accepted.
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.1.2.1.5 trainPosition

Input name	trainPosition
Description	Contains the current position of the train.
Source	CalculateTrainPosition
Type	TrainPosition_Types_Pck::trainPosition_T
Valid range of values	
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

135 #### 3.1.2.1.6 modeAndLevel

Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.1.2.1.7 tNvContact

Input name	tNvContact
Description	For monitoring the safe radio connection, this national value is needed as an input.
Source	Database
Type	Obu_BasicTypes_Pkg::T_internal_Type
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.1.2.1.8 lastRelevantEventTimestamp

Input name	lastRelevantEventTimestamp
Description	For monitoring the safe radio connection, it is necessary that the time between two packets is less than the value of T_NVCONTACT. In situations like level-changes or announced radio holes, not the timestamp of the last message is relevant for comparison, but the timestamp of the last relevant event. This can for example be the timestamp of the level change or the timestamp of the moment, when the train was passing the end of the radiohole. For performing this check, the timestamp of the last relevant event is provided to the model as an T_internal_Type-type.
Source	Database
Type	Obu_BasicTypes_Pkg::T_internal_Type
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.1.2.1.9 connectionStatus

Input name	connectionStatus
Description	Status information about the radio connection. The information is needed to perform the timing check, which depends on the connection state.
Source	ManageRadioCommunication
Type	Radio_Types_Pkg::sessionStatus_Type
Valid range of values	**DISCONNECTED** The OBU is currently not connected to a RBC. **CONNECTING** The OBU is currently connecting to the RBC. Received messages belong to the process of establishing a connection. **CONNECTION_ESTABLISHED** The connection to the RBC is established.
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.1.2.1.10 inSupervisingRbcId

Input name	inSupervisingRbcId
Description	For the sub component InformationFilter, the information which radio messages are sent by the supervising RBC is needed. To recognize these messages, the identifier of the supervising RBC is needed.
Source	Database
Type	int
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

140 #### 3.1.2.1.11 inAnnouncedBGs

Input name	inAnnouncedBGs
Description	Provides information about balise groups which will be passed by the train soon. This information is generated by Calculate Train Position based on the linking information received from trackside.
Source	CalculateTrainPosition
Type	TrainPosition_Types_Pck::positionedBGs_T
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.1.2.1.12 q_nvlocacc

Input name	q_nvlocacc
Description	The national value determines the location accuracy.
Source	Database
Type	Q_NVLOCACC
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.1.2.2 Outputs

3.1.2.2.1 outputMessage

Output name	outputMessage
Description	Combines both balise and radio messages to one common datatype. This datatype contains all variables and packets, which are possible for the given scenario.
Destination	[Name of the destination component(s)]
Type	Common_Types_Pkg::ReceivedMessage_T
Valid range of values	[Complete list of valid values]

Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.1.2.2.2 ApplyServiceBrake

Output name	ApplyServiceBrake
Description	Indicates if the balise group the train just passed could not be processed correctly. The check results in the request for a service break.
Destination	[Name of the destination component(s)]
Type	bool
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.1.2.2.3 BadBAliseMessageToDMI

Output name	BadBAliseMessageToDMI
Description	Information to be passed to the DMI to indicate the reception of a “bad balise” to the driver.
Destination	DMI
Type	bool
Valid range of values	**true** ??? **false** ???
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.1.2.2.4 errorLinkedBG

Output name	errorLinkedBG
Description	[Brief description of the output]
Destination	[Name of the destination component(s)]
Type	[Type of the output]
Valid range of values	**true** An error in a linked balise group was detected. **false** No error in a linked balise group was detected.
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.1.2.2.5 errorUnlinkedBG

Output name	errorUnlinkedBG
Description	[Brief description of the output]
Destination	[Name of the destination component(s)]
Type	bool
Valid range of values	**true** An error in a unlinked balise group was detected. **false** No error in a unlinked balise group was detected.
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.1.2.2.6 passedBG

Output name	passedBG
Description	Provides the received balise group message in a special format needed by the component CalculateTrainPosition.
Destination	[Name of the destination component(s)]

Type	BG_Types_Pkg::passedBG_T
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.1.2.2.7 outPositionParams

Output name	outPositionParams
Description	Provides the parameters for the position report in a special format needed by the component ProvidePositionReport.
Destination	[Name of the destination component(s)]
Type	Common_Types_Pkg::PositionReportParameter_T
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.1.2.2.8 outRadioManagement

Output name	outRadioManagement
Description	Provides the messages for radio session management in a special format needed by the component ManagementOfRadioCommunication.
Destination	[Name of the destination component(s)]
Type	Common_Types_Pkg::radioManagementMessage_T
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.1.2.2.9 radioSequenceError

Output name	radioSequenceError
Description	[Brief description of the output]
Destination	[Name of the destination component(s)]
Type	bool
Valid range of values	**true** A sequence error or a timeout has been detected in the radio message. **false** No error in the radio message sequence was detected.
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.1.2.2.10 radioMessageConsistencyError

Output name	radioMessageConsistencyError
Description	[Brief description of the output]
Destination	[Name of the destination component(s)]
Type	bool
Valid range of values	**true** A consistency error has been detected in the radio message. **false** No consistency error in the radio message was detected.
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.1.3 Sub Components

3.1.3.1 Receive_TrackSide_Msg

155 #### 3.1.3.1.1 Component Requirements

Component name	Receive_TrackSide_Msg

Link to SCADE model	`https://github.com/openETCS/modeling/tree/master/model/Scade/System/ObuFunctions/ManageLocationRelatedInformation/BaliseGroup/Receive_TrackSide_Msg`
SCADE designer	[Name, affiliation]
Description	This function defines the interface of the OBU model to the openETCS generic API for Eurobalise and Euroradio messages. On the interface, either a valid telegram/message is provided or a telegram/message is indicated which could not be received correct when passing the balise or receiving the radio message. The function passes a balise telegram without major changes of the information to the next entity for collecting the balise group information. This entity collects telegrams received via the interface into Balise Group Information. In case of a radio message, the message is converted to an internal format for further processing and passed without changing the information contained. • The decoding of balises is done at the API. Also, packets received via the interface are already transformed into a usable shape. • Only packets used inside the current model are passed via the interface. • Treatment of Packet 5: Linking Information. Linking Information is added to 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. • Telegrams received as invalid are passed to the “Check-Function” to process errors in communication with the track side according to the requirements and in a single place. Telegrams are added to 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. • This function does not process information from the packets. The information is passed to the check without further processing of the values.
Input documents	Subset-026, Chapter 7 and 8: Definition of the Balise Telegram Subset-026, Chapter 4.2.2, 4.2.4, 4.2.9: Interface to the BTM Subset-026, Chapter 3.4.1 - 3.4.3, 3.16.2: Handling of Balise Telegrams Subset-026, Chapter 3.16.2: Check of the balise group Subset-026, Chapter 3.4.2: Determining the orientation Subset-026, Chapter 4.5.2 Active Functions Table Subset-026, Chapter 8.4.4: Rules for Euroradio messages
Safety integrity level	4
Time constraints	n/a

API requirements	n/a

3.1.3.1.2 Interface

For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

3.1.3.2 CheckBGConsistency

3.1.3.2.1 Component Requirements

Component name	CheckBGConsistency
Link to SCADE model	`https://github.com/openETCS/modeling/tree/master/model/Scade/System/ObuFunctions/ManageLocationRelatedInformation/BaliseGroup/CheckBGConsistency`
SCADE designer	[Name, affiliation]
Description	This function verifies the completeness and correctness of the received messages from balise groups. A message consists of at least a telegram and a maximum of 8 telegrams. • A message is still complete and correct, if a telegram is missing (or not decoded or incomplete decoded), and this telegram is duplicated within the balise group and the duplicating one is correctly read. • By more than one telegram, the order of the telegrams must be either ascending (nominal) or descending(reverse). • A message is correct, if all message counters (M MCUNT) do not equal 254 (that means: The telegram never fits any message of the group). A message counter can be equal 255 (that means: The telegram fits with all telegrams of the same balise group) and all other values must be the same. The orientation of the BG will also be calculated in this block. The check, if the message has been received in due time and the right at the right expected location, will be performed in "Calculate Train Position". The checks on the validity of the data in the packets and the validity with respect to the direction of motion will be performed in other modules, e.g. "Validate Data Direction".
Input documents	Subset-026, Chapter 7 and 8: Definition of the Balise Telegram Subset-026, Chapter 3.4.1-3, 3.16.2: Handling of Balise Telegrams Subset-026, Chapter 3.16.2: Check of the balise group Subset-026, Chapter 4.5.2: Active Functions Table
Safety integrity level	4
Time constraints	n/a
API requirements	n/a

3.1.3.2.2 Interface

For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

3.1.3.3 CheckEuroradioMessage

165 #### 3.1.3.3.1 Component Requirements

Component name	CheckEuroradioMessage
Link to SCADE model	https://github.com/openETCS/modeling/tree/b9c31ce6fdf702b412bbeab3032a8a4dc7c92e5c/model/Scade/System/ObuFunctions/ManageLocationRelatedInformation/BaliseGroup/CheckEuroRadioMessage
SCADE designer	Stefan Karg, DB Netz AG
Description	The component “CheckEuroradioMessage” performs several checks on the received radio message. These checks include checking of the message sequence, completeness of messages. Invalid messages are marked as invalid in the message header.
Input documents	Subset-026, Chapter 3.16 Subset-026, Chapter 8.4.4
Safety integrity level	4
Time constraints	n/a
API requirements	n/a

3.1.3.3.2 Interface

For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

3.1.3.4 ValidateDataDirection

170 #### 3.1.3.4.1 Component Requirements

Component name	CheckEuroradioMessage
Link to SCADE model	https://github.com/openETCS/modeling/tree/master/model/Scade/System/ObuFunctions/ManageLocationRelatedInformation/BaliseGroup/ValidateDataDirection
SCADE designer	???

Description	The component filters an input message in order to mark all elements as invalid, which are not designated for the current driving direction of the train. • The operator contains two processing paths for different message types. Radio messages and balise group messages are handeled in a different way. For validating the data direction of a radio message, the check is performed using the balise group referenced in the radio message header as relevant balise group. For balise group message, the LRBG is used. • The metadata of packets, which are recognized as not valid for the current driving direction, is invalidated.
Input documents	Subset-026, Chapter 3.6.3
Safety integrity level	4
Time constraints	n/a
API requirements	n/a

3.1.3.4.2 Interface

For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

3.1.3.5 InformationFilter

175
3.1.3.5.1 Component Requirements

Component name	CheckEuroradioMessage
Link to SCADE model	`https://github.com/openETCS/modeling/tree/master/model/Scade/System/ObuFunctions/ManageLocationRelatedInformation/BaliseGroup/InformationFilter`
SCADE designer	Alexander Stante, FhG

Description	The filter receives track information (balise and radio) and filters them depending of the mode, level and source of the message. Only messages that pass the filter are valid and should be considered by other ETCS subsystems. Figure 3 shows the highlevel decomposition of the functionality. The filter is consists of four components: FirstFilter, SecondFilter, ThirdFilter and TransitionBuffer. **FirstFilter** This filter performs filtering of messages based on the current ETCS level. The decisions taken process is described via a big decision table which contains rows for every packet and columns for every ETCS level. This table encodes also if certain additional information is necessary to filter a message like pending ETCS Level transitions. Based on this filter packets of an incoming message is either rejected, accepted or the whole message is put in the TransitionBuffer. Messages are put in the TransitionBuffer if there is an announced level transition and the received message is only valid for the upcoming level. **SecondFilter** The SecondFilter mainly considers messages that are received via Euroradio. Certain messages are directly rejected while other may be stored in the TransitionBuffer. The buffer is used to store messages that are received from non supervising RBCs, but will be reevaluated after a RBC transition. **ThirdFilter** The last filter is functionally very similiar the the FirstFilter, however it filters depending on the mode. It also contains a decision table with rows for every packet but the columns are modes. **TransitionBuffer** The InformationFilter uses two TransitionBuffers. One is used to store up to three messages for the ETCS level transition and the other buffer is used for RBC transitions. The buffer is designed as a ring buffer and message are read in FIFO order.
Input documents	Subset-026, Chapter 4.8
Safety integrity level	4
Time constraints	n/a
API requirements	n/a

3.1.3.5.2 Interface

For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

3.2 Train_Supervision

3.2.1 Component Requirements

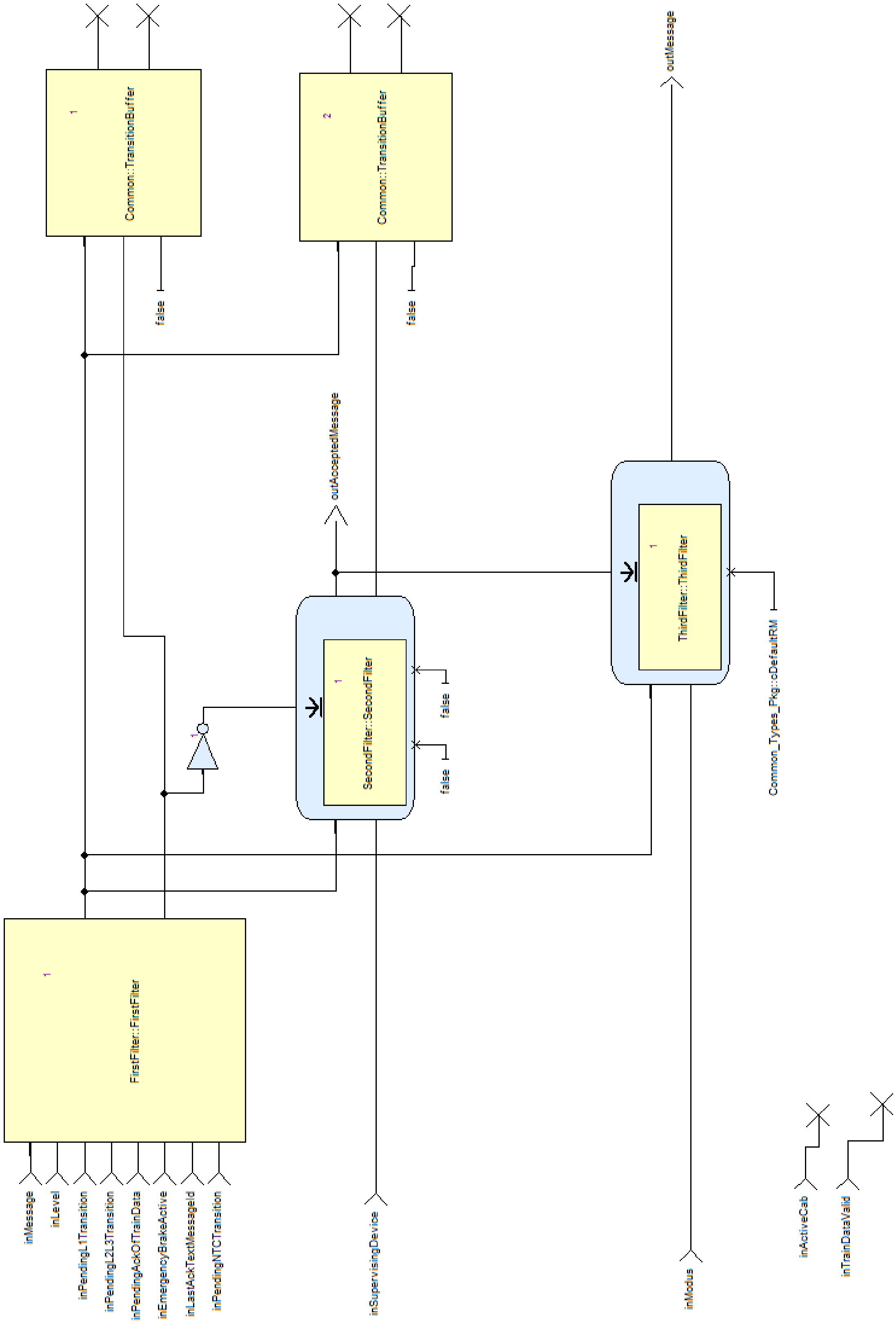


Figure 3. High level overview of the InformationFilter components.

Component name	TrainSupervision
Link to SCADE model	???
SCADE designer	Christian Stahl, TWT
Description	The task of block “Train Supervision” is to monitor the speed of the train and the train location and as such to ensure that the speed remains within the given speed and distance limits. This block is mainly based on [1, Chapt. 3.13]. The block “Train Supervision” takes as input (1) movement related information such as train speed, train position and acceleration, (2) train related information such as brake information and train length, and (3) track related information such as speed and distance limits and national values. Based on this information a speed profile is calculated. Speed restrictions create target speeds (targets) that have to be followed. For each such target braking curves are generated to supervise at which location of the track the train must perform the brake. In case of no target restrictions the train may accelerate to the supervised maximum speed of the speed profile. These calculations lead to commands being sent to the driver and the brake system. The functionality is modeled using eight operators, as shown in Figure 4, which are explained below. The current status of the analysis of “Train Supervision” and a functional breakdown can be found in a separate document, `SpeedSupervision_analysis.pdf`.
Input documents	Subset-026, Chapter 3.13: Speed and distance monitoring
Safety integrity level	4
Time constraints	[If applicable description of time constraints, otherwise n/a]
API requirements	[If applicable description of API requirements, otherwise n/a]

3.2.2 Interface

An overview of the interface of component [component name] is shown in Figure 4. The inputs and outputs are described in detail in Section 3.2.2.1 respectively **??**.

3.2.2.1 Inputs

185
3.2.2.1.1 NationalValues

Input name	NationalValues
Description	This input is packet 3 of [1, Chapt. 8], describing the national values.
Source	???
Type	P3_NationalValues_T

Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.2.2.1.2 TrainPosition

Input name	TrainPosition
Description	This input is the current train position.
Source	Manage Track Data
Type	trainPosition_T
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.2.2.1.3 odometry

Input name	odometry
Description	This input is the odometry data.
Source	Odometry
Type	odometry_T
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.2.2.1.4 m_level

Input name	m_level
Description	This input is the current level of the train.
Source	Mode and Level

Type	M_LEVEL
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.2.2.1.5 trainProps

Input name	trainProps
Description	This input is a set of train related properties.
Source	Database
Type	trainProperties_T
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.2.2.1.6 MRSP

Input name	MRSP
Description	This input is the most restrictive speed profile.
Source	???
Type	MRSP_Profile_t
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.2.2.1.7 MA

Input name	MA
Description	This input is a movement authority.

Source	???
Type	MAs_t
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.2.2.1.8 MA_updated

Input name	MA_updated
Description	This flag is true if the movement authority has been updated in this clock cycle and false otherwise.
Source	internal
Type	bool
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.2.2.1.9 MRSP_updated

Input name	MRSP_updated
Description	This flag is true if the most restrictive speed profile has been updated in this clock cycle and false otherwise.
Source	internal
Type	bool
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.2.2.2 Outputs

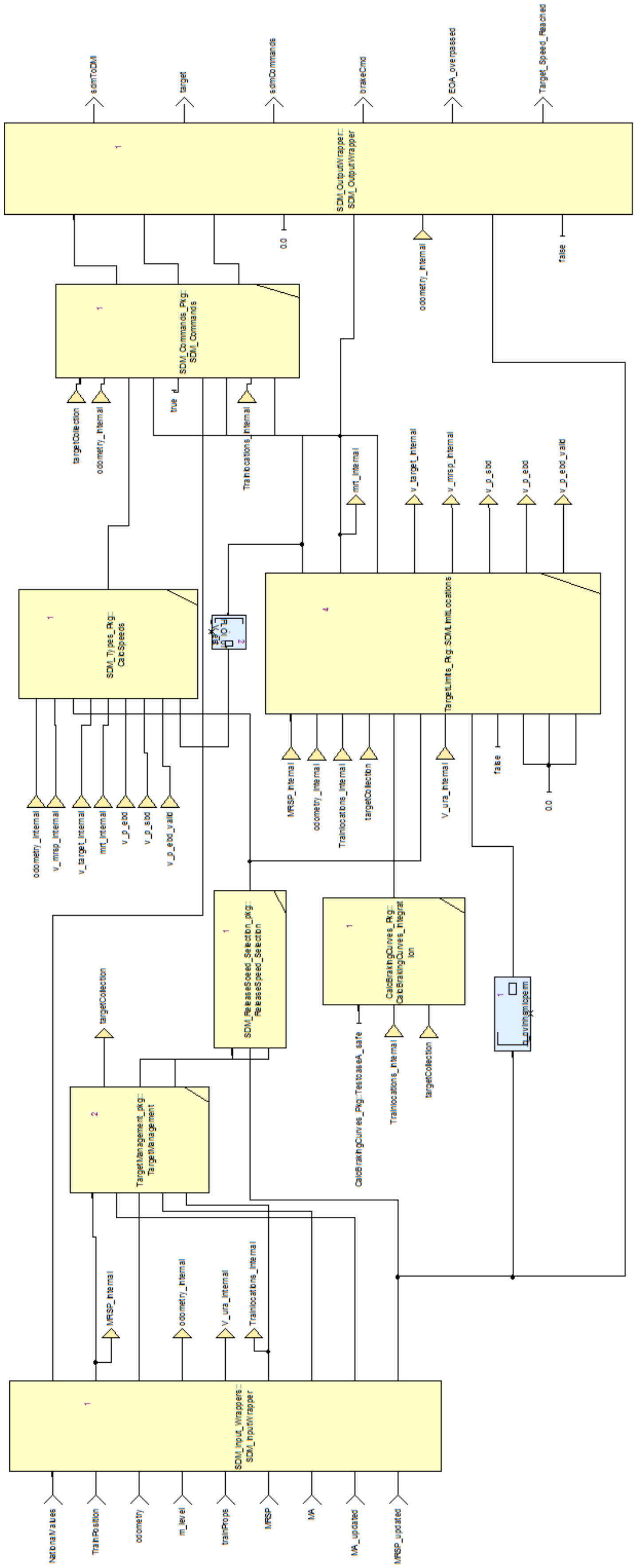


Figure 4. Structure of component ProvidePositionReport.

195 3.2.2.2.1 sdmToDMI

Output name	sdmToDMI
Description	This output contains information about different speeds and positions, on the one hand and the current supervision status, on the other hand. This information shall be displayed to the driver.
Destination	[Name of the destination component(s)]
Type	speedSupervisionForDMI_T
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.2.2.2.2 target

Output name	target
Description	This output is the most restrictive displayed target (MRDT).
Destination	[Name of the destination component(s)]
Type	Target_T
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.2.2.2.3 sdmCommands

Output name	sdmCommands
Description	This output gives some intermediate results of operator SDM_Commands. It is currently used for test purposes only.
Destination	[Name of the destination component(s)]
Type	SDM_Commands_T
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]

Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.2.2.2.4 brakeCmd

Output name	brakeCmd
Description	This output is the brake command, indicating whether performing the service brake or the emergency brake have been commanded.
Destination	[Name of the destination component(s)]
Type	Brake_command_T
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.2.2.2.5 EOA_overpassed

Output name	EOA_overpassed
Description	This output is true if the end of authority has been overpassed and false otherwise.
Destination	[Name of the destination component(s)]
Type	bool
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.2.2.2.6 Target_Speed_Reached

Output name	Target_Speed_Reached
Description	This output is true if the current speed is greater than or equal the target speed and false otherwise.
Destination	[Name of the destination component(s)]
Type	bool

Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.2.3 Sub Components

3.2.3.1 Receive_TrackSide_Msg

3.2.3.1.1 Component Requirements

Component name	SDM_InputWrapper
Link to SCADE model	???
SCADE designer	Christian Stahl, TWT
Description	The motivation for this operator is to convert all inputs of block “Speed Supervision” that contain information about length, speed, distance, and acceleration defined as integer into `real` to allow automatically the highest precision in the calculations by the meaning of floating point operations. In addition, to ease the modeling, inside block “Speed Supervision” only units meters ([m]), seconds([s]), meters per second([$\frac{m}{s}$]), and meters per square second([$\frac{m}{s^2}$]) are used. This operator forwards input messages, takes data from complex data types or transforms inputs messages into an internal type thereby converting int to real.
Input documents	Subset-026, Chapter ?.? Subset-026, Chapter ?.? Subset-026, Chapter ?.?.?
Safety integrity level	4
Time constraints	[If applicable description of time constraints, otherwise n/a]
API requirements	[If applicable description of API requirements, otherwise n/a]

3.2.3.1.2 Interface

205 For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

3.2.3.2 TargetManagement

3.2.3.2.1 Component Requirements

Component name	TargetManagement
Link to SCADE model	???
SCADE designer	Christian Stahl, TWT
Description	This operator calculates/updates the list of targets to be supervised by the block “Train Supervision”. Taking the current movement authority, the most restrictive speed profile and the current maximum safe front end position as an input, the operator outputs a single End of Authority target, a list of all MRSP-Targets and a list of all LoA-Targets. **Derivation of Targets from Movement Authority Sections** The sections of the *Movement Authority* could cause two types of targets: **End Of Authority(EoA)** only one could exist and this is only in the *end section* of the *MA* **Limit of Authority (LoA)** is possibly in every section of the *MA* except the end section In every cycle in which the MA is updated, the operator iterates through the entire MA and puts all speed limitations by *LoA*s into a list of targets. The end section is used to derived the *EoA* target. All LoA targets are sorted by location. **Derivation of Targets from MRSP** According to [1, Chapt. 3.13.8.2], every speed decrease of the MRSP is used to derive a target. Therefore in every cycle in which the MRSP is updated, the operator iterates through the entire MRSP searching for all MRSP targets. For this purpose, every element of the MRSP is compared with its successor. **Update of Targets** In every cycle the operator monitors whether all targets are already passed. To this end, it iterates over the list of targets comparing the current max safe front end position with the target position.
Input documents	Subset-026, Chapter 3.13.8.2: Determination of the supervised targets
Safety integrity level	4
Time constraints	[If applicable description of time constraints, otherwise n/a]
API requirements	[If applicable description of API requirements, otherwise n/a]

3.2.3.2.2 Interface

For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

3.2.3.3 CalcBrakingCurves_Integration

3.2.3.3.1 Component Requirements

Component name	CalcBrakingCurves_Integration
Link to SCADE model	???
SCADE designer	Christian Stahl, TWT
Description	For each type of target a certain braking curve has to be calculated. This curve enables proactive monitoring of the train's speed. A reverse lookup on this braking curve indicates, where the train has to start braking given the current speed. The braking curve does not depend on the actual train status. As a consequence the braking curve stays constant over time. As a legitimate simplification the calculation of the braking curve is not extended after the estimated front end position of the train has been passed.
Input documents	Subset-026, Chapter 3.13.8.3: Emergency Brake Deceleration curves (EBD) Subset-026, Chapter 3.13.8.4: Service Brake Deceleration curves (SBD) Subset-026, Chapter 3.13.8.5: Guidance curves (GUI)
Safety integrity level	4
Time constraints	[If applicable description of time constraints, otherwise n/a]
API requirements	[If applicable description of API requirements, otherwise n/a]

3.2.3.3.2 Interface

215 For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

3.2.3.4 SDMLimitLocations

3.2.3.4.1 Component Requirements

Component name	SDMLimitLocations
Link to SCADE model	???
SCADE designer	???

Description	This operator calculates the various locations needed to determine the speed and distance monitoring commands. The current implementation of functionality is stateless and requires a complete recalculation each cycle. This operator gathers all necessary input values and computes some frequently used intermediate values in the operators `surplusTractionDeltas` and v_{bec}. The other input preparation operator is the `TargetSelector` whose main task is to dissect the list of targets to find the Most Restrictive Target. The accompanying braking curves are extracted and promoted to trailing location calculations. Also the special values of the EOA are exposed. The operator creates the requested values for the commands package. These are in particular the preindication locations for EBD and SBD based targets, the release speed monitoring start locations, the locations for target speed monitoring of the I-, W-, P- and FLOI-curve, the related FLOI speed and the location of the permitted speed supervision limit. Included in the output are also certain flags for the validity of linked values.
Input documents	Subset-026, Chapter 3.13.9: Supervision Limits Subset-026, Chapter 5.3.1.2: f_{41} – accuracy of speed known onboard Subset-026, Chapter 3.13.10: Monitoring Commands as reference for required outputs of this module
Safety integrity level	4
Time constraints	[If applicable description of time constraints, otherwise n/a]
API requirements	[If applicable description of API requirements, otherwise n/a]

3.2.3.4.2 Interface

220 For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

3.2.3.5 CalcSpeeds

3.2.3.5.1 Component Requirements

Component name	CalcSpeeds
Link to SCADE model	???
SCADE designer	???
Description	This operator calculates the various speeds needed to determine the speed and distance monitoring commands. This operator will be integrated into other operators in the next iteration.
Input documents	Subset-026, Chapter 3.8: Movement authority
Safety integrity level	4

Time constraints	[If applicable description of time constraints, otherwise n/a]
API requirements	[If applicable description of API requirements, otherwise n/a]

3.2.3.5.2 Interface

225 For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

3.2.3.6 ReleaseSpeed_Selection

3.2.3.6.1 Component Requirements

Component name	ReleaseSpeed_Selection
Link to SCADE model	???
SCADE designer	???
Description	This operator outputs the release speed which can be given either by the national values or the movement authority. This operator will be integrated into other operators in the next iteration.
Input documents	Subset-026, Chapter 3.8: Movement authority
Safety integrity level	4
Time constraints	[If applicable description of time constraints, otherwise n/a]
API requirements	[If applicable description of API requirements, otherwise n/a]

3.2.3.6.2 Interface

230 For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

3.2.3.7 SDM_Commands

3.2.3.7.1 Component Requirements

Component name	SDM_Commands
Link to SCADE model	???
SCADE designer	???

Description	This operator models the speed and distance monitoring commands. More precisely, it triggers the service or emergency brake and outputs the current supervision status of the OBU together with information on speeds and locations to the driver. The OBU can be in any of three types of speed and distance monitoring modes: ceiling speed monitoring, release speed monitoring and target speed monitoring. We use a state machine to model the switching between the three modes: each state models a mode and a transition between to states is enabled if the condition two switch between the two corresponding modes is evaluated to true. In each mode, the OBU can be in up to five different supervision stati. The behavior of changing from one status to another is also modeled as a state machine. As a result, the model is a hierarchical state machine.
Input documents	Subset-026, Chapter 3.13.10: Speed and distance monitoring commands
Safety integrity level	4
Time constraints	[If applicable description of time constraints, otherwise n/a]
API requirements	[If applicable description of API requirements, otherwise n/a]

3.2.3.7.2 Interface

235 For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

3.2.3.8 SDM_OutputWrapper

3.2.3.8.1 Component Requirements

Component name	SDM_OutputWrapper
Link to SCADE model	???
SCADE designer	???
Description	This operator is the counterpart to operator SDM_OutputWrapper—that is, it converts all internal outputs of block “Speed Supervision” that contain information about length, speed, distance, and acceleration defined as real into int, such that all other blocks can stick to their types and also performs the calculation into units used by the environment. This operator forwards input messages and transforms inputs messages into an internal type thereby converting real to int.
Input documents	Subset-026, Chapter 3.13: Speed and distance monitoring
Safety integrity level	4
Time constraints	[If applicable description of time constraints, otherwise n/a]

[Put SysML diagram of component here]

Figure 5. Component SysML diagram

API requirements	[If applicable description of API requirements, otherwise n/a]

3.2.3.8.2 Interface

For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

3.3 Manage_ETCS_Procedures

3.3.1 Component Requirements

Component name	Manage_ETCS_Procedures
Link to SCADE model	???
SCADE designer	???
Description	???
Input documents	Subset-026, Chapter ???
Safety integrity level	4
Time constraints	[If applicable description of time constraints, otherwise n/a]
API requirements	[If applicable description of API requirements, otherwise n/a]

3.3.2 Interface

An overview of the interface of component [component name] is shown in Figure **??**. The inputs and outputs are described in detail in Section 3.3.2.2 respectively **??**.

3.3.2.1 Inputs

3.3.2.1.1 [Input 1 name]

Input name	[Name of the input]
Description	[Brief description of the input]
Source	[Name of the source component]
Type	[Type of the input]
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]

Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.3.2.1.2 [Input 2 name]

Input name	[Name of the input]
Description	[Brief description of the input]
Source	[Name of the source component]
Type	[Type of the input]
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.3.2.2 Outputs

3.3.2.2.1 [Output 1 name]

Output name	[Name of the output]
Description	[Brief description of the output]
Destination	[Name of the destination component(s)]
Type	[Type of the output]
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.3.2.2.2 [Output 2 name]

Output name	[Name of the output]
Description	[Brief description of the output]
Destination	[Name of the destination component(s)]
Type	[Type of the output]
Valid range of values	[Complete list of valid values]

Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.3.3 Sub Components

3.3.3.1 Awakening_of_train

3.3.3.1.1 Component Requirements

Component name	Manage_ETCS_Procedures
Link to SCADE model	`https://github.com/openETCS/modeling/blob/master/model/Scade/System/ObuFunctions/Procedures/ManageProcedure_Pkg.xscade`
SCADE designer	???
Description	This component describes the Start of Mission procedure of the train until the status of the awakeness. From this point of the awakness the train will be able to start different modes, levels and further procedure. See scope of the Start of Mission - Awakness of train in the figure below. For the third iteration just a part of the Scope has been design. To complete the scenario in the third iteration the ideal path to the awakness of train until the state "waiting for Driver selection of "Start"" have been realized. Furthermore the initial data from the persistend database such as Level, Driver ID, Train Number, Train Data, Radio Number, RBC ID hase been consider as constants.
Input documents	Subset-026, Chapter 5, § 5.4
Safety integrity level	4
Time constraints	[If applicable description of time constraints, otherwise n/a]
API requirements	[If applicable description of API requirements, otherwise n/a]

3.3.3.1.2 Interface

For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

3.3.3.2 SOM_Level_2_3

3.3.3.2.1 Component Requirements

Component name	SOM_Level_2_3

Link to SCADE model	`https://github.com/openETCS/modeling/blob/master/model/Scade/System/ObuFunctions/Procedures/SoM_SR_FS_OS_LS_SH_SN_UN.xscade`
SCADE designer	???
Description	This functionality describes the Start of Mission procedure of the train in Level 2 or 3 and the Modes SR FS OS LS SH where the train under the defined Mode Level supervision starts running. For the this iteration just a part of the Scope has been design. To complete the scenario in the third iteration the path "Full Supervision Movement Authority received from RBC" has been realized. The state will end after the train receives the Change Authority to FS and will be ready to run.
Input documents	Subset-026, Chapter 5, § 5.4
Safety integrity level	4
Time constraints	[If applicable description of time constraints, otherwise n/a]
API requirements	[If applicable description of API requirements, otherwise n/a]

3.3.3.2.2 Interface

For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

3.4 Manage_Track_Data

265 3.4.1 Component Requirements

Component name	Manage_Track_Data
Link to SCADE model	???
SCADE designer	???
Description	???
Input documents	Subset-026, Chapter ???
Safety integrity level	4
Time constraints	[If applicable description of time constraints, otherwise n/a]
API requirements	[If applicable description of API requirements, otherwise n/a]

3.4.2 Interface

An overview of the interface of component [component name] is shown in Figure **??**. The inputs and outputs are described in detail in Section 3.4.2.2 respectively **??**.

[Put SysML diagram of component here]

Figure 6. Component SysML diagram

3.4.2.1 Inputs

3.4.2.1.1 [Input 1 name]

Input name	[Name of the input]
Description	[Brief description of the input]
Source	[Name of the source component]
Type	[Type of the input]
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.4.2.1.2 [Input 2 name]

Input name	[Name of the input]
Description	[Brief description of the input]
Source	[Name of the source component]
Type	[Type of the input]
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.4.2.2 Outputs

3.4.2.2.1 [Output 1 name]

Output name	[Name of the output]
Description	[Brief description of the output]
Destination	[Name of the destination component(s)]
Type	[Type of the output]
Valid range of values	[Complete list of valid values]

Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.4.2.2.2 [Output 2 name]

Output name	[Name of the output]
Description	[Brief description of the output]
Destination	[Name of the destination component(s)]
Type	[Type of the output]
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.4.3 Sub Components 275

3.4.3.1 Calculate_Train_Position

3.4.3.1.1 Component Requirements

Component name	Calculate_Train_Position
Link to SCADE model	???
SCADE designer	???

Description	The main purpose of the function is to calculate the locations of linked and unlinked balise groups (BGs) and the current train position while the train is running along the track. In detail, the calculateTrainPosition function provides a couple of essential subfunctions for the onboard unit. These are mainly • creating and maintaining an obu internal coordinate system for all types of location based data • storing all linked and unlinked balise groups resulting from over passing or from announcements (linking information) from the track • calculating and maintaining the locations of all stored balise groups during the train trip, based on odometry and linking information • permanently calculating the current train position based on odometry and passed balise group information • providing the last recently passed linked balise group as the LRBG • providing additional position attribute information • deleting stored balise groups, when appropriate • detecting linking consistency errors • determining, if linking is used on board The calculation algorithms for locations and positions are implemented as specified in `https://github.com/openETCS/SRS-Analysis/blob/master/System%20Analysis/WorkingRepository/Group4/SUBSET_26_3-6/DetermineTrainLocationProcedures.pdf`
Input documents	Subset-026, Chapter 3.6
Safety integrity level	4
Time constraints	[If applicable description of time constraints, otherwise n/a]
API requirements	[If applicable description of API requirements, otherwise n/a]

3.4.3.1.2 Interface

For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

3.4.3.2 Provide_Position_Report

3.4.3.2.1 Component Requirements

Component name	Provide_Position_Report

Link to SCADE model	???
SCADE designer	???
Description	This function takes the current train position and generates a position report which is sent to the RBC. The point in time when such a report is sent is determined by events, on the one hand, and position report parameters—which are basically triggers—provided by the RBC or a balise group passed, on the other hand. The functionality is modeled using four operators, which are explained below. **CalculateSafeTrainLength** Calculates the the safeTrainLength and the MinSafeRearEnd according to [1, Chapter 3.6.5.2.4/5]. `safeTrainLength = absolute(EstimatedFrontEndPosition - MinSafeR` where `MinSafeRearEnd = minSafeFrontEndPosition - L_TRAIN`. **EvaluateTriggerAndEvents** Returns a Boolean modelling whether the sending of the next position report is triggered or not. This value is the conjunction of the evaluation of all triggers (PositionReportParameters, i.e., Packet 58) and events (see [1, Chapter 3.6.5.1.4]). **ErrorManager** Takes a boolean flag for each possible error that has been occurred and outputs the respective error using type `M_ERROR` **CollectData** This operation aggregates data of Packet 0, . . . , Packet 5 and the header to a position report.
Input documents	Subset-026, Chapter 3.6.5
Safety integrity level	4
Time constraints	[If applicable description of time constraints, otherwise n/a]
API requirements	[If applicable description of API requirements, otherwise n/a]

3.4.3.2.2 Interface

For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

3.5 Mode_and_Level

3.5.1 Component Requirements

Component name	Mode_and_Level
Link to SCADE model	???
SCADE designer	???
Description	???

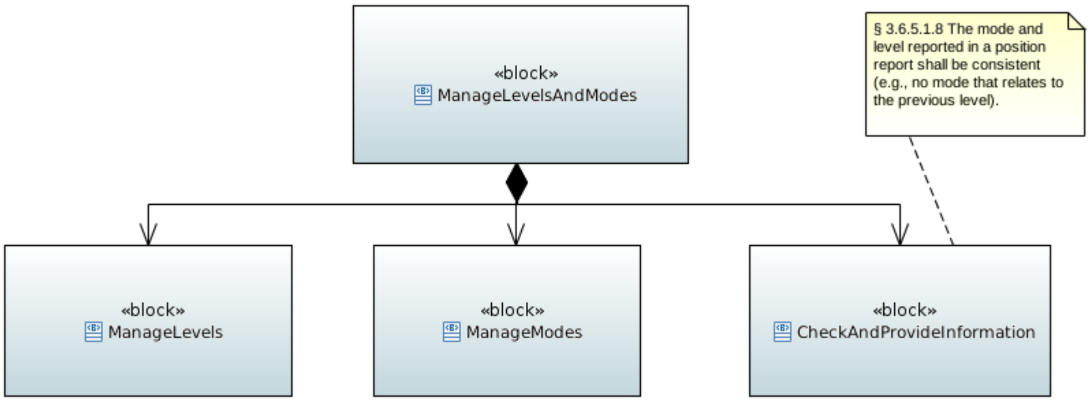

«block»
ManageLevelsAndModes
§ 3.6.5.1.8 The mode and level reported in a position report shall be consistent (e.g., no mode that relates to the previous level).
«block»
ManageLevels
«block»
ManageModes
«block»
CheckAndProvideInformation

Figure 7. Component SysML diagram

Input documents	Subset-026, Chapter 4 Subset-026, Chapter 5
Safety integrity level	4
Time constraints	[If applicable description of time constraints, otherwise n/a]
API requirements	[If applicable description of API requirements, otherwise n/a]

3.5.2 Interface

An overview of the interface of component Mode_and_Level is shown in Figure 7. The inputs and outputs are described in detail in Section 3.5.2.1 respectively 3.5.2.2.

3.5.2.1 Inputs

3.5.2.1.1 [Input 1 name]

Input name	[Name of the input]
Description	[Brief description of the input]
Source	[Name of the source component]
Type	[Type of the input]
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

3.5.2.1.2 [Input 2 name]

Input name	[Name of the input]
Description	[Brief description of the input]
Source	[Name of the source component]

3.5.2.2 Outputs

295 ##### 3.5.2.2.1 [Output 1 name]

Output name	[Name of the output]
Description	[Brief description of the output]
Destination	[Name of the destination component(s)]
Type	[Type of the output]
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.5.2.2.2 [Output 2 name]

Output name	[Name of the output]
Description	[Brief description of the output]
Destination	[Name of the destination component(s)]
Type	[Type of the output]
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

3.5.3 Sub Components

3.5.3.1 Level_Management

3.5.3.1.1 Component Requirements

Component name	Level_Management
Link to SCADE model	`https://github.com/openETCS/modeling/tree/master/openETCSArchitectureAndDesign/WorkGroups/Group3/SCADE/LevelManagement/`
SCADE designer	Marielle Petit-Doche, Systerel

Description	The level management subsystem receives level transition order tables and selects the order with the highest probability. It stores the information about the selected transition order and transits to the requested level once the train passes the location of the level transition. If required, the driver is asked to acknowledge the transition, in case of no acknowledge or if conditions for the level transition are not fulfilled, the train gets tripped. On the most abstract level the design consists of the *manage_priorities* function which takes the level transition order priority tables as inputs and computes the highest priority transition. This transition order is the fed to the *computeLevelTransitions* operator. This operator consists of three main parts. The *ComputeTransitionConditions* operator that emits the fulfilled conditions to change from a given level to a new level, the *LevelStateMachine* that stores the current level and takes the computed change conditions as input for possible level transitions and finally the *driverAck* operator which contains a state machine that stores the information whether the system is currently waiting for a driver acknowledge and emits the train trip information if necessary.
Input documents	Subset-026, Chapter 5.10
Safety integrity level	4
Time constraints	[If applicable description of time constraints, otherwise n/a]
API requirements	[If applicable description of API requirements, otherwise n/a]

3.5.3.1.2 Interface

For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

3.5.3.2 Mode_Management

3.5.3.2.1 Component Requirements

Component name	Mode_Management
Link to SCADE model	`https://github.com/openETCS/modeling/tree/master/model/Scade/System/ObuFunctions/ManageLevelsAndModes/Modes`
SCADE designer	Marielle Petit-Doche, Systerel

Description	This function is in charge of the computation of new mode to apply according to conditions from inputs (track information, driver interactions, train data,...) and other functions. Three subfunctions are defined: **Inputs** proceeds to inputs check and preparation. **ComputeModesCondition** performs all specific procedure linked to mode management and defined in [1] sections 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.11, 5.12, 5.13, 5.19 and specifies the conditions to define a mode transition according condition table of section 4.6.3 of [1] **SwitchModes** performs the mode selection according the conditions and priorities defined in transition table section 4.6.2 of [1] **Outputs** prepares packet of outputs.
Input documents	Subset-026, Chapter 4.4, 4.6, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.11, 5.12, 5.13, 5.19
Safety integrity level	4
Time constraints	[If applicable description of time constraints, otherwise n/a]
API requirements	[If applicable description of API requirements, otherwise n/a]

3.5.3.2.2 Interface

For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

3.5.3.3 Check_and_Provide_Mode_and_Level

3.5.3.3.1 Component Requirements

Component name	Check_and_Provide_Mode_and_Level
Link to SCADE model	`https://github.com/openETCS/modeling/tree/master/model/Scade/System/ObuFunctions/ManageLevelsAndModes/Modes`
SCADE designer	Marielle Petit-Doche, Systerel
Description	Checks compatibility between mode and level and provides outputs.
Input documents	Subset-026, Chapter 3.6.5
Safety integrity level	4
Time constraints	[If applicable description of time constraints, otherwise n/a]
API requirements	[If applicable description of API requirements, otherwise n/a]

310 3.5.3.3.2 Interface

For an overview of the interface of this internal component we refer to the SCADE model (c.f. link above) respectively the SCADE generated documentation.

4 F3: Measure Train Movement

5 F4: Manage Radio Communication

Part III

Deprecated Design Description

6 Component Design Template

6.1 Component Requirements

Component name	[Component name]
Link to SCADE model	`http://???`
SCADE designer	[Name, affiliation]
Description	[Brief description of the components functionality]
Input documents	Subset-026, Chapter ?.? Subset-026, Chapter ?.? Subset-026, Chapter ?.?.?
Safety integrity level	4
Time constraints	[If applicable description of time constraints, otherwise n/a]
API requirements	[If applicable description of API requirements, otherwise n/a]

6.2 Interface

An overview of the interface of component [component name] is shown in Figure 8. The inputs and outputs are described in detail in Section 6.2.1 respectively 6.2.2.

6.2.1 Inputs

6.2.1.1 [Input 1 name]

Input name	[Name of the input]
Description	[Brief description of the input]
Source	[Name of the source component]
Type	[Type of the input]
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

6.2.1.2 [Input 2 name]

[Put SysML diagram of component here]

Figure 8: Component SysML diagram

Input name	[Name of the input]
Description	[Brief description of the input]
Source	[Name of the source component]
Type	[Type of the input]
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when input value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when input value is out of valid range]

6.2.2 Outputs

6.2.2.1 [Output 1 name]

Output name	[Name of the output]
Description	[Brief description of the output]
Destination	[Name of the destination component(s)]
Type	[Type of the output]
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

6.2.2.2 [Output 2 name]

Output name	[Name of the output]
Description	[Brief description of the output]
Destination	[Name of the destination component(s)]
Type	[Type of the output]
Valid range of values	[Complete list of valid values]
Behaviour when value is at boundary	[Description of components behaviour when output value is at boundary]
Behaviour for values out of valid range	[Description of components behaviour when output value is out of valid range]

7 F1: Receive information from Trackside

8 F2: ETCS Kernel

8.1 Mode and Level

The "Management of Modes and Levels" function is mainly described in chapter 4 and 5 of [1]. Modes and levels define the status of the ETCS regarding on-board functional status and track infrastructure.

8.1.1 Function Level Management

8.1.1.1 Reference to the SRS or other Requirements

See [1] section 5.10

8.1.1.2 Short description of the functionality

The level management subsystem receives level transition order tables and selects the order with the highest probability. It stores the information about the selected transition order and transits to the requested level once the train passes the location of the level transition.

If required, the driver is asked to acknowledge the transition, in case of no acknowledge or if conditions for the level transition are not fulfilled, the train gets tripped.

8.1.1.3 Interface

The interface consists of the following inputs:

- *conditional transitions:* a priority table containing the conditional level transition orders (from paquet 46)
- *level transition priority table:* a priority table containing the (non-conditional) level transition orders (from paquet 41)
- *train standstill:* a Boolean value indicating whether the train is at standstill (from odometry)

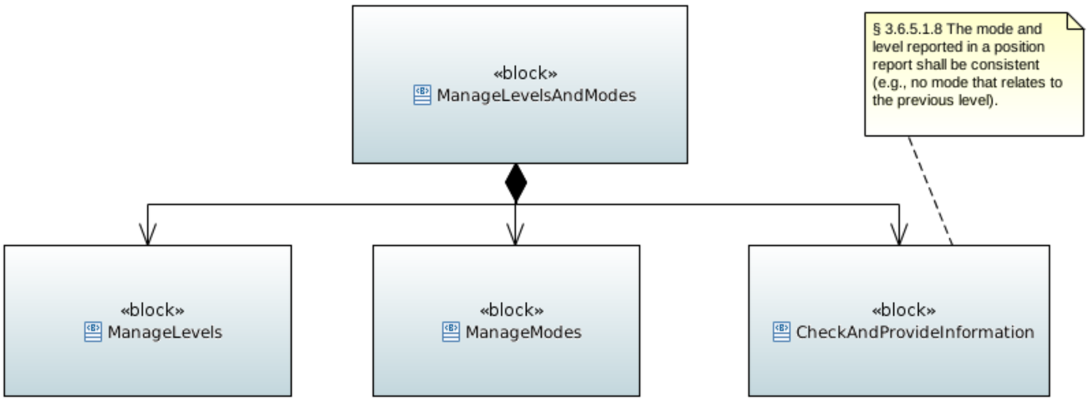

«block»
ManageLevelsAndModes
§ 3.6.5.1.8 The mode and level reported in a position report shall be consistent (e.g., no mode that relates to the previous level).
«block»
ManageLevels
«block»
ManageModes
«block»
CheckAndProvideInformation

Figure 9. High level Architecture

- *driver level transition:* a level transition order selected by the driver (from DMI)
- *ERTMS capabilities:* the ERTMS capabilities of the track
- *getAck:* Boolean input that signals the acknowledgment of the driver (from DMI)
- *resetIdle:* Boolean input to reset without acknowledge
- *currentDistance:* the current position of the train given with the same reference as the position of the level transition order (train position , from localisation)
- *ackDistance:* the maximal distance for driver acknowledge after the level transition (from paquet 41)
- *immediateAck:* a Boolean that signals that an immediate acknowledge is required
- *received L2 L3 MA:* a Boolean that indicates that a level 2 or level 3 movement authority for the track behind the level transition has been received (from paquet 15)
- *received L1 MA:* a Boolean that indicates that a level 1 movement authority for the track behind the level transition has been received (from paquet 12)
- *received target speed:* a Boolean indicating that a target speed for the track behind the level transition has been received (from paquet 27) ?

and the following outputs:

- *next level:* the next level after this computation cycle
- *Trip train:* a Boolean indicating whether the train should be tripped
- *previous level:* the previous level before this computation cycle
- *needsAckFromDriver:* a Boolean that indicates whether an acknowledgment from the driver is necessary

8.1.1.4 Functional Design Description

On the most abstract level the design consists of the *manage_priorities* function which takes the level transition order priority tables as inputs and computes the highest priority transition.

This transition order is the fed to the *computeLevelTransitions* operator. This operator consists of three main parts. The *ComputeTransitionConditions* operator that emits the fulfilled conditions to change from a given level to a new level, the *LevelStateMachine* that stores the current level and takes the computed change conditions as input for possible level transitions and finally the *driverAck* operator which contains a state machine that stores the information whether the system is currently waiting for a driver acknowledge and emits the train trip information if necessary.

8.1.1.5 Reference to the Scade Model

The Scade model is available on GitHub: `https://github.com/openETCS/modeling/tree/master/openETCSArchitectureAndDesign/WorkGroups/Group3/SCADE/LevelManagement/`

8.1.2 Function Mode Management

8.1.2.0.1 Reference to the SRS or other Requirements

see [1] sections 4.4, 4.6, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.11, 5.12, 5.13, 5.19

8.1.2.1 Short description of the functionality

This function is in charge of the computation of new mode to apply according to conditions from inputs (track information, driver interactions, train data,...) and other functions.

8.1.2.2 Interface

The inputs are the following:

- *Cab* identification of the current cabin (A or B)
- *Continue_shunting_Function_Active*: boolean to describe the activation state of the shunting function
- *Current_Level*: outputs of the Level management function
- *Data_From_DMI*: set of data received from the driver via the DMI interface, indeed:
  - *Ack_LS : bool* Driver acknoledges LS mode
  - *Ack_OS : bool*
  - *Ack_RV : bool*
  - *Ack_SH : bool*
  - *Ack_SN : bool*
  - *Ack_SR : bool*
  - *Ack_TR : bool*
  - *Ack_UN : bool*
  - *Req_Exit_SH : bool* driver selects exit of shunting
  - *Req_NL : bool* Driver requests NL mode
  - *Req_Override : bool* Driver requests override function
  - *Req_SH : bool* driver requests SH mode
  - *Req_Start : bool* Driver requests start of mission
  - *ETCS_Isolated: bool*: isolation status of the ETCS
- *Data_From_Localisstion*: set of data received from the function in charge of localistion of the train, indeed:
  - *BG_In_List_Expected_BG_In_SR : bool*: the identity of the overpass balise group is in the list of expected balises related to SR mode (from SR to trip mode condition 36)
  - *BG_In_List_Expected_BG_In_SH : bool*: the identity of the overpass balise group is in the list of expected balises related to SH mode (from SH to trip mode condition 52)
  - *Linked_BG_In_Wrong_Direction : bool* balise group contained in the linking information is passed in the unexpected direction (from FS, LS, OS to trip mode condition 66) *Localisaion function ?*
  - *Train_Position*: output provided by function in charge of computation of train possition (type TrainPosition_Types_Pck::trainPosition_T)

  - *Train_Speed : Obu_BasicTypes_Pkg::Speed_T* provided by odometry function
  - *Train_Standstill : bool* provided by odometry function

- *Data_From_Speed_and_Supervision*: set of data received from the function in charge of speed and supervision management, indeed:
  - *Estim_front_End_overpass_SR_Dist : bool*: the train overpass the SR distance with its estimated front end (from SR to trip mode condition 42)
  - *Estim_Front_End_Rear_SSP : bool*: estimated front end is rear of the start location of either SSP or gradient profile stored on-board (from FS, LS, OS to trip mode condition 69)
  - *Override_Function_Active*: boolean to indicate the state of the activation function
  - *EOA_Antenna_Overpass : bool*: the train overpasses the EOA with min safe antenna position Level 1 (from FS, LS, OS to trip mode condition 12)
  - *EOA_Front_End : bool* the train overpasses the EOA with min safe front end, Level 2 or 3 (from FS, LS, OS to trip mode condition 16)
  - *Train_Speed_Under_Overide_Limit : bool* supervision when override function is active (to SR mode condition 37)
- *Data_From_TIU : TIU_Types_Pkg::Message_Train_Interface_to_EVC_T*:message provided by TIU interface
- *Data_From_Track*: set of data received from track side (via RBC or Balises telegram), indeed:
  - *MA_SSP_Gradiant_Available : bool* MA, SSP and gradient have been received, checked and stored on-board from paquet 12, 15, 21 and 27 or message 3 or 33
  - *Mode_Profile_On_Board : Level_And_Mode_Types_Pkg::T_Mode_Profile* from packet 80
  - *Shunting_granted_By_RBC : bool* from message 27 and 28
  - *Trip_Order_Given_By_Balise : bool*
  - *List_Bg_Related_To_SR_Empty : bool* from packet 63
  - *Stop_If_In_shunting : bool* from packet 135
  - *Stop_If_In_SR : bool* from packet 137
  - *Error_BG_System_Version : bool*
  - *Linking_Reaction_To_Trip : bool*
  - *RBC_Ack_TR_EB_Revocked : bool* from message 6
  - *RBC_Authorized_SR : bool* from message 2
  - *Reversing_Data : Level_And_Mode_Types_Pkg::T_Reversing_Data* from packet 138/ 139
  - *T_NVCONTACT_Overpass : bool* Maximal time without new safe message overpass
  - *Emergency_Stop_Message_Received*: boolean to describe the reception of Emergency Stop message from message 15 or 16
- *Failure_Occured*: boolean to indicate safety failure occurence
- *Interface_To_National_System*: boolean to indicate existance of an interface to a national system
- *National_Trip_Order*: boolean to indicate reception of a trip order from a national system

- *OnBoard_Powered*: boolean to indicate the poxering state of the system
- *Stop_Shunting_Stored*: boolean to store the information in regards of shunting function
- *Valid_Train_Data_Stored*: boolean to indication train data are available and valid.

The outputs are the following:

- *currentMode* the new computed mode (typeis Level_And_Mode_Types_Pkg::T_Mode, default value is Level_And_Mode_Types_Pkg::SB)
- *EB_Requested* boolean to request triggering of emergency brake
- *Service_Brake_Command* boolean to request command of service brake
- *Data_To_DMI*: set of data provided to the DMI Level_And_Mode_Types_Pkg::T_Data_To_DMI :
  - *Ack_LS : bool* Driver acknoledges LS mode
  - *Ack_OS : bool*
  - *Ack_RV : bool*
  - *Ack_SH : bool*
  - *Ack_SN : bool*
  - *Ack_SR : bool*
  - *Ack_TR : bool*
  - *Ack_UN : bool*
  - *Req_Exit_SH : bool* driver selects exit of shunting
  - *Req_NL : bool* Driver requests NL mode
  - *Req_Override : bool* Driver requests override function
  - *Req_SH : bool* driver requests SH mode
  - *Req_Start : bool* Driver requests start of mission
  - *ETCS_Isolated: bool*: isolation status of the ETCS
- *Data_To_BG_Management*: set of date to trackside Level_And_Mode_Types_Pkg::T_Data_To_BG_Management :
  - *EoM_Procedure_req : bool* request of end of mission procedure indeed end of the communication session for message 150
  - *Clean_BG_List_SH_Area : bool* request to clean the BG list when entering an SH area §5.6.2
  - *MA_Req : bool* for message 132
  - *Req_for_SH_from_driver : bool* for message 130

8.1.2.3 Functional Design Description

Three subfunctions are defined:

Inputs proceeds to inputs check and preparation.

ComputeModesCondition performs all specific procedure linked to mode management and defined in [1] sections 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.11, 5.12, 5.13, 5.19 and specifies the conditions to define a mode transition according condition table of section 4.6.3 of [1]

SwitchModes performs the mode selection according the conditions and priorities defined in transition table section 4.6.2 of [1]

Outputs prepares packet of outputs.

8.1.2.4 Reference to the Scade Model

The Scade model is available on github: `https://github.com/openETCS/modeling/tree/master/model/Scade/System/ObuFunctions/ManageLevelsAndModes/Modes`

8.1.3 Function Check and Provide Level and Mode

8.1.3.1 Reference to the SRS or other Requirements

see [1] section 3.6.5

8.1.3.2 Short description of the functionality

checks compatibility between mode and level and provides outputs

8.1.3.3 Interface

To design

8.1.3.4 Functional Design Description

To design

8.1.3.5 Reference to the Scade Model

To design

8.2 Manage Radio Communication

8.2.1 Management of Radio Communication (*MoRC*)

8.2.1.1 Reference to the SRS

The management of radio communication is specified in Subset-026, chap. 3.5.

8.2.1.2 Short description of the functionality

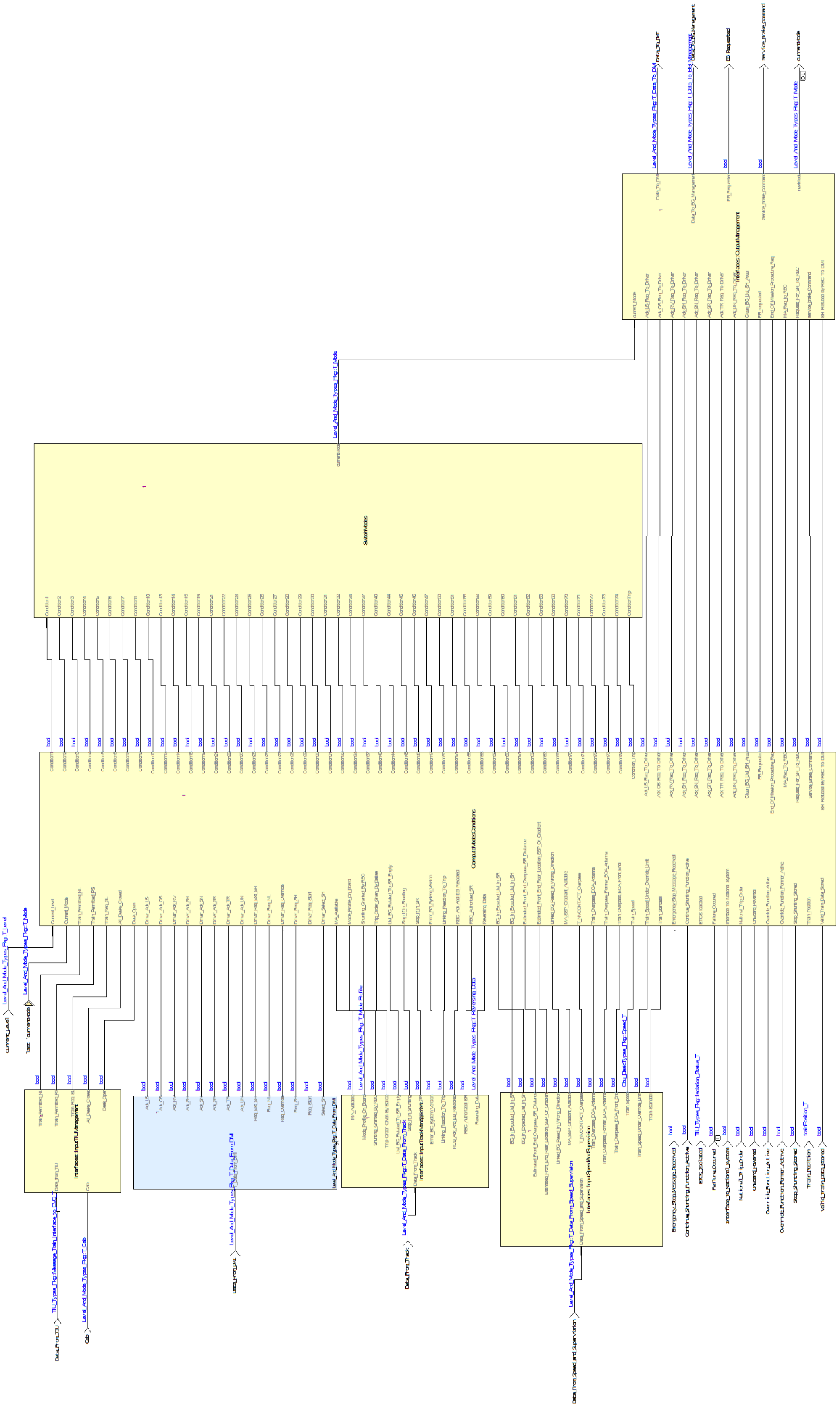

SwitchModes
ComputeModesConditions
Interfaces::OutputManagement
Interfaces::InputTIUManagement
Interfaces::InputTrackManagement
Interfaces::InputSpeedAndSupervision
Data_To_DMI
Data_To_BG_Management
EB_Requested
Service_Brake_Command
currentMode
Current_Level
Data_From_TIU
Cab
Data_From_DMI
Data_From_Track
Data_From_Speed_And_Supervision
Level_And_Mode_Types_Pkg::T_Mode
bool

Figure 10. Modes subfunction architecture.

The management of radio communication *MoRC* implements the on board management part of a single communication session with the track, i.e. a single RBC. It controls the establishing, maintaining and termination process of a radio communication session and steers the underlying communication safety layer and the mobile device. Those and the data transfer itself are not part of the function.

8.2.1.3 Interface

8.2.1.3.1 Inputs

The MoRC function takes as inputs datagrams received from track, OBU internal phases and status information and configuration data:

- Datagrams received from track (*inMessage*):
  - Packet 42 (session management) received from balise group or RBC
  - Packet 45 (radio network registration) received from balise group or RBC
  - Message 32 (RBC/RIU System Version) received from RBC: *MoRC* only needs to know if the system version received from track side is supported by the OBU.
  - Message 38 (initiation of a communication session) received from RBC
  - Message 39 (acknowledgement of termination of a communication session)
- *obuEventsAndPhases*: information about OBU internal events and OBU internal phases:
  - *atPowerDown*
  - *atPowerUp*
  - *atStartOfMission*
  - *startOfMissionProcedureIsGoingOn*
  - *startOfMissionProcedureCompleted*
  - *trainIsRejectedByRBC_duringStartOfMission*
  - *endOfMissionIsExecuted*
  - *driverClosesTheDeskduringStartOfMission*
  - *driverHasManuallyChangedLevel*
  - *afterDriverEntryOfANewRadioNetworkID*
  - *triggerDecisionThatNoRadioNetworkIDAvailable*
  - *isPartOfAnOngoingStartOfMissionProcedure*
  - *trainPassesALevelTransitionBorder*
  - *trainPassesA_RBC_RBC_border_WithItsFrontEnd*
  - *trainExitedFromAnRBCArea*
  - *modeChangeHasToBeReportedToRBC*
  - *trainFrontInsideInAnAnnouncedRadioHole*
  - *trainFrontReachesEndOfAnnouncedRadioHole*
  - *OBU_hasToEstablishANewSession*
  - *isInCommunicationSessionWithAnRIU*
  - *errorConditionRequiringTerminationDetected*

- Current OBU internal states:
  - *currentTime*: current OBU system time
  - *t_train*: current trainborne clock (T_TRAIN) as specified in Subset-026, chap. 7
  - *mode*: current OBU mode
  - *level*: current OBU level
- *statusOfMobile*: status of the associated mobile device
- configuration parameters:
  - *onboardPhoneNumbers* (NID_RADIO)
  - *radioNetworkIDs*: Identities of radio networks (NID_MN): default, memorized or from driver
  - *nid_engine*: Onboard ETCS identity (NID_ENGINE)
  - *connectionStatusTimerInterval*: Connection status timer period

8.2.1.3.2 Outputs

MoRC generates a couple of outputs:

- *MessageToRBC*: messages to be sent to the RBC:
  - Message 155 (initiation of a communication session)
  - Message 156 (termination of a communication session)
  - Message 159 (session established)
  - Message 154 (no compatible version supported)
- Action triggers:
  - *sendAPositionReport*: triggers a position report to be sent to the RBC
  - *memorizeTheLastRadioNetworkID*: triggers to store the last radio network ID for later use
  - *orderTheRegistrationOfItsConnectedMobiles*
  - *rejectOrderToContactRBC_or_RIU*
  - *InformTheDriverThatNoConnectionWasSetup*
  - *requestTheSetupOfASafeRadioConnection*: initiate the setup of a safe radio connection
  - *requestReleaseOfSafeRadioConnectionWithTrackside*: initiate the release of a safe radio connection
  - *ignoreMessagesFromRBC_except_m39_AckOfTerminationOfCommunicationSession*
  - *sessionSuccessfullyEstablished*
- *cmdsToMobile*: control commands to the mobile device
- Status information:
  - *sessionStatus*: current session status
  - *mobileSWStatus*: connection status
  - *currentRadioNetworkID*: current radio network ID

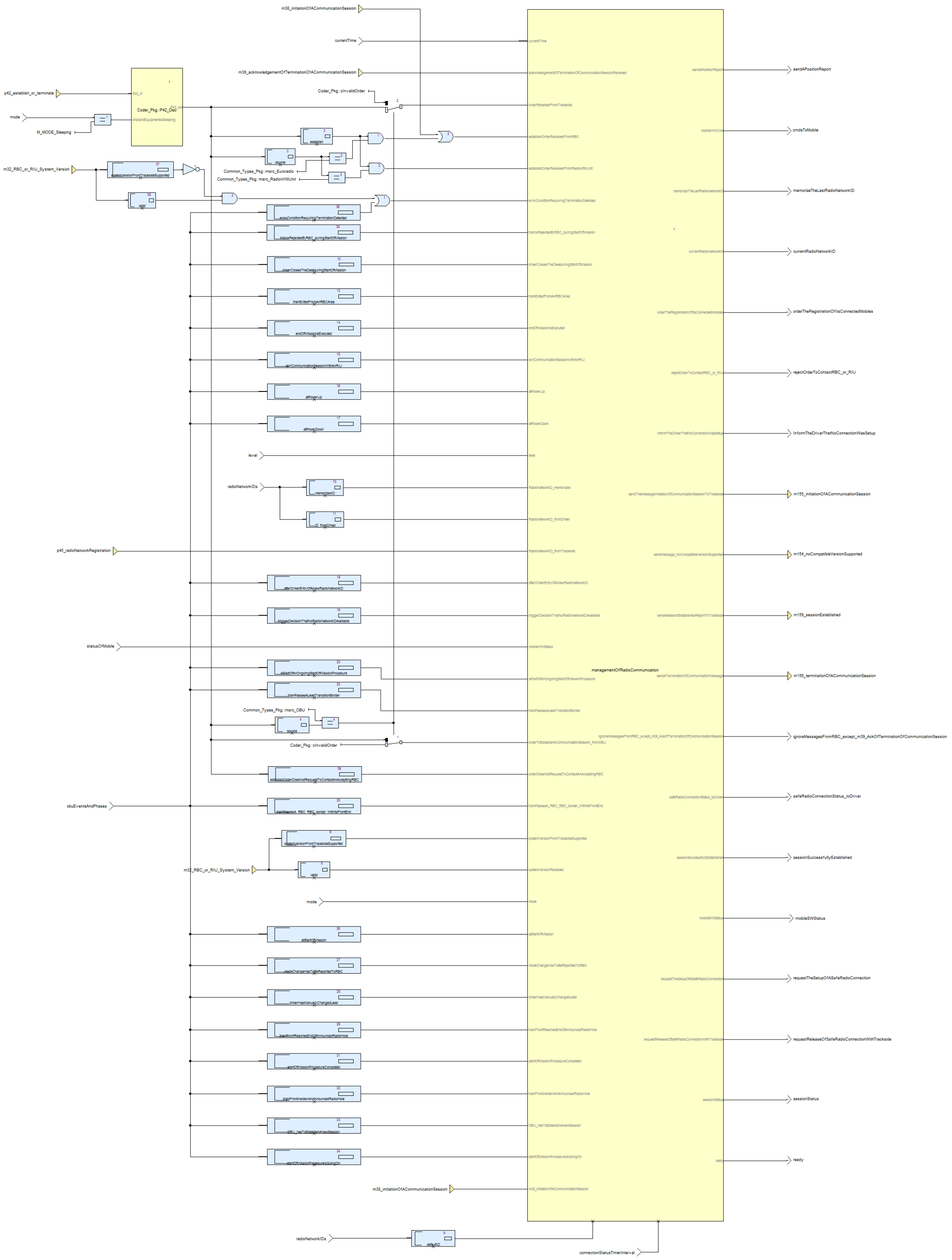

m33_initiationOfACommunicationSession
currentTime
m39_acknowledgementOfTerminationOfACommunicationSession
Coder_Pkg::cInvalidOrder
p42_establish_or_terminate
Coder_Pkg::P42_Dec
mode
M_MODE_Sleeping
m32_RBC_or_RIU_System_Version
Common_Types_Pkg::marc_Euroradio
Common_Types_Pkg::marc_RadioInfillUnit
level
radioNetworkIDs
p45_radioNetworkRegistration
statusOfMobile
Common_Types_Pkg::marc_OBU
Coder_Pkg::cInvalidOrder
obuEventsAndPhases
m32_RBC_or_RIU_System_Version
mode
m33_initiationOfACommunicationSession
radioNetworkIDs
connectionStatusTimerInterval
managementOfRadioCommunication
sendAPositionReport
cmdsToMobile
memorizeTheLastRadioNetworkID
currentRadioNetworkID
orderTheRegistrationOfItsConnectedMobiles
rejectOrderToContactRBC_or_RIU
InformTheDriverThatNoConnectionWasSetup
m155_initiationOfACommunicationSession
m154_noCompatibleVersionSupported
m159_sessionEstablished
m156_terminationOfACommunicationSession
ignoreMessagesFromRBC_except_m39_AckOfTerminationOfCommunicationSession
safeRadioConnectionStatus_toDriver
sessionSuccessfullyEstablished
mobileSWStatus
requestTheSetupOfASafeRadioConnection
requestReleaseOfSafeRadioConnectionWithTrackside
sessionStatus
ready

Figure 11. Main function of *MoRC*.

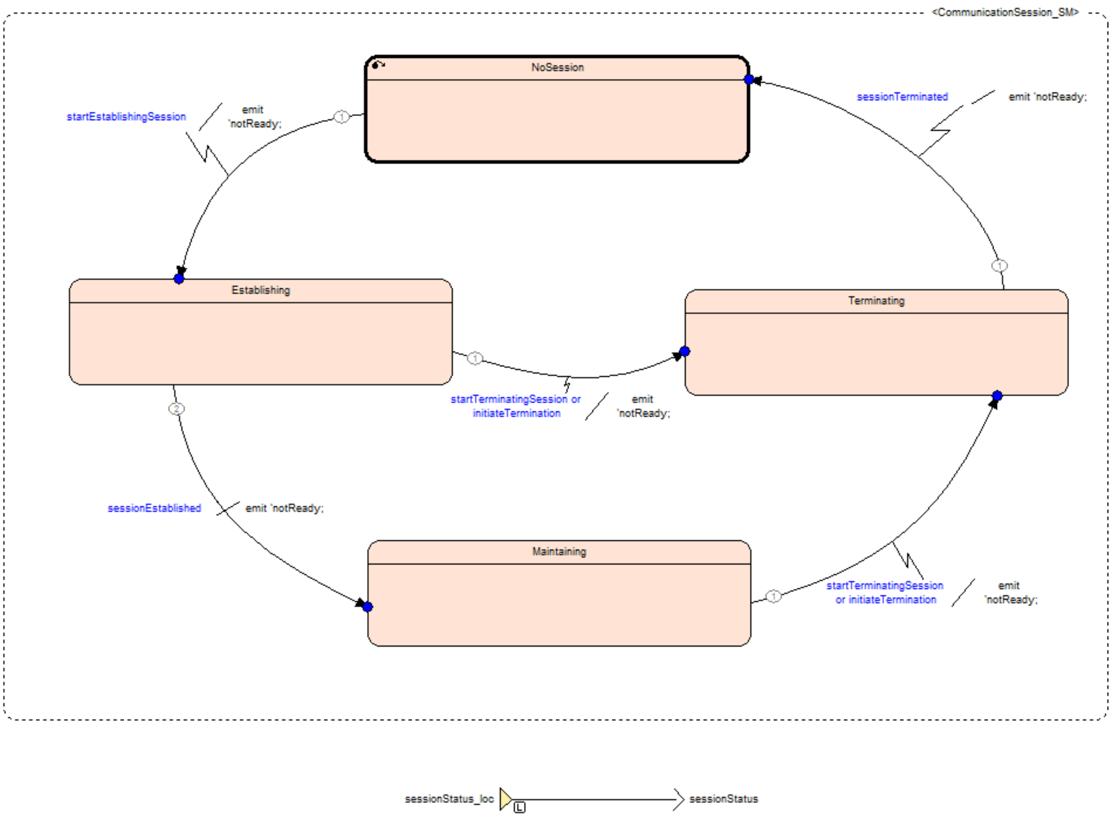


Figure 12. Implementation of session states.

8.2.1.4 Functional Design Description

The kernel function of the *MoRC* component is *managementOfRadioCommunication* (figure ???). The implementation is kept close to the prose of Subset-026, chap. 3.5. Since chap. 3.5 rarely refers to terms, variable types, packets and messages of the ETCS language as specified in Subset-026, chap. 7 and 8, *managementOfRadioCommunication* does neither.

To be capable of being integrated with other OBU software components, *MoRC* had to be wrapped with a transformer between the ETCS and the "chap. 3.5" language. This is the purpose of the main function of *MoRC*, *MoRC_Main*.

The function *managementOfRadioCommunication* implements the session states establishing, maintaining and termination as described in Subset-026, chap. 3.5. A SCADE state machine reflects this state model (figure ???) accurately. Within each of the states, the activities needed as long as the state is active, are performed. When there is no communication session (state *NoSession*) currently, the state machine waits for events that initiate a session (subfunction *initiate_a_Session*). When the appropriate conditions are fulfilled, the state machine moves to the *Establishing* state. Here in, it runs through the sequence required fore establishing a session (subfunction *establish_a_Session*. Dependent on the results, the state machine changes over to the *Maintaining* or *Terminating* state. While in *Maintaining*, the communication connection is monitored. When an event triggering the session termination occurs, the state machine switches to the state *Terminating* with the subfunction *terminating_a_CommunicationSession* and performs the session termination sequence.

In parallel to the main state machine, *managementOfRadioCommunication* monitors all the time whether the session has to be terminated (subfunction *initiateTerminatingASession*) or if the session has the be terminated and subsequently established (subfunction *terminateAndEstablishSession*). *registeringToTheRadioNetwork* is responsible for connection to the radio network. *safeRadioConnectionIndication* controls the radio connection indication for the driver.

8.2.1.5 Reference to the Scade Model

The MoRC SCADE model resides at `https://github.com/openETCS/modeling/tree/master/model/Scade/System/ObuFunctions/Radio/MoRC` .

9 F3: Measure Train Movement

10 F4: Manage Radio Communication

11 F5: Manage JRU

12 F6: DMI Controller

12.1 DMI Controller

12.1.0.6 Reference to the SRS or other Requirements (or other requirements)

ERA_ERTMS_015560

12.1.0.7 Short description of the functionality

The DMI controller interact with the DMI display and is responsible for alls procedures between the DMI display and Driver. Furthermore, the DMI controller will interact with the DMI Management to compute the received information (e.g. driver number request, ...) and send, if necessary, data or reports to the DMI Management (acknowledge, text messages...). The DMI Controller is a passive module, this means that all the processing are performed EVC-side, therefore the DMI Controller simply responds to the requests of the EVC or Driver and performs some checks according with the information received from EVC.

12.1.0.8 Interface

The DMI Controller has two interfaces. One between DMI Controller and DMI Display and one between DMI Controller and DMI Management. The structure of the interface between DMI Controller and DMI Display is driven by the logic of SCADE Display therefore It doesn't follow any standard or constraints (It will not be described in this chapter). DMI Controller and DMI Management exchange packets. Each packet is a structured type with a valid flag (a boolean variable), the DMI controller takes into account the data inside the packet only when the valid flag is true.

The interface between DMI Controller and DMI Management consist of three parts according with the direction of the information:

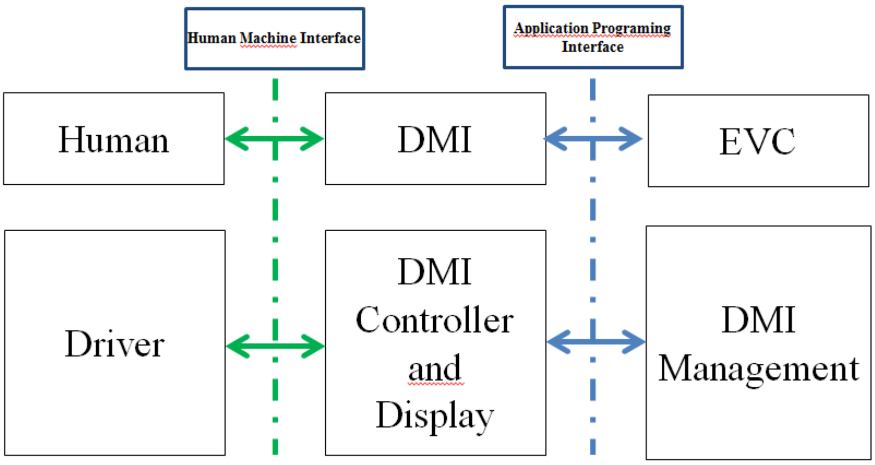


Figure 13. DMI Interfaces

- From DMI Management to DMI Controller
- From DMI Controller to DMI Management
- Both ways directions (You will find the same type both as input than as output)

12.1.0.8.1 From DMI Management to DMI Controller

In the following table are listed the inputs coming from DMI Management with a brief description:

NAME	DESCRIPTION
`DMI_entry_request`	Request to input data (e.g. driver id, Train running number etc.)
`DMI_identifier_request`	Request of the DMI informations
`DMI_menu_request`	Request to enable or disable buttons
`DMI_dynamic`	Contains informations about current speed, current mode etc.
`DMI_text_message`	Contains predefined or plain text messages
`DMI_icons`	Request to display one or more icons in any area

Please note: TIU_trainStatus input is missing in the above table. This is the only input coming directly from TIU and contains the open/close Desk signal.

12.1.0.8.2 From DMI Controller to DMI Management

In the following table are listed the outputs directed to DMI Management with a brief description:

NAME	DESCRIPTION
`DMI_identifier`	Information about DMI (e.g. version, cabin identifier etc.)
`DMI_driver_request`	Driver request or acknowledgement
`DMI_train_data_ack`	Train data acknowledgement
`DMI_status_report`	The actual status of DMI (keep alive)
`DMI_text_message_ack`	Text message acknowledgement
`DMI_icons_ack`	Icon acknowledgement

12.1.0.8.3 Both ways direction

In the following table are listed the outputs/inputs to/from DMI Management with a brief description:

NAME	DESCRIPTION
`DMI_driver_identifier`	Contains the default or entered driver identifier
`DMI_train_running_number`	Contains the default or entered train running number
`DMI_train_data`	Contains the default or entered train data

12.1.0.9 Functional Design Description

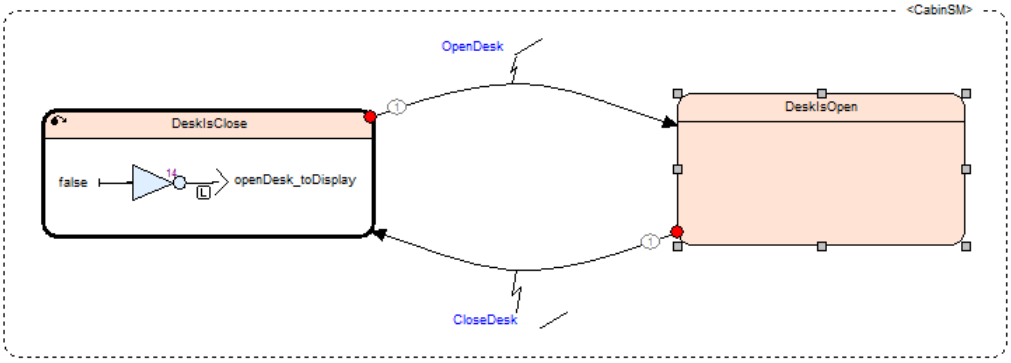


Figure 14. Cabin State Machine.

Please note: *DMI Controller is a project under construction, a lot of features and functionalities are missing, therefore the structure described below is a draft version and will be changing in the future.*

The informations (received and sent) could be divided in two groups: Sporadic and Periodic. The first one are received/sent aperiodically in any time instead the second one are received/sent periodically, with a fixed deadline. Are part of Periodic group the output DM_status_report and the input DMI_dynamic all other are Sporadic. Therefore, the structure of DMI Controller module consists of a first main state machine *CabinSM* (Fig. 14) triggered by a *OpenDesk* signal (from TIU). Inside the *DeskIsOpen* state there are other two state machines :*HandshakeSM* and *DynamicInfoSM* (Fig. 15).

HandshakeSM performs an initial handshake between DMI Controller and DMI Management. Before that, no data has to be sent or received to/from DMI Management. When the transition is fired a DMI_identifier packet is sent to DMI Management with informations about the DMI (e.g. DMI identifier, DMI name etc.). At this point the DMI Controler is ready to manage the sporadic information (e.g. Enter or revalidate DriverID, Enter or revalidate Train running number etc.).

The DynamicInfoSM state machine is triggered after the handshake, exactly when HandshakeSM reaches the DynInfo_Activated state. At the time when the transition is fired a signal is emitted (startDMI_status) and begins a periodic sending of DMI status information (keep alive) to DMI Management. Once reached DynamicInfo_Active, the DMI Controller is ready to receive and manage the dynamic informations.

With the aim to improve the readability and for a better management of complexity, all the functions (modules, state machines etc.) implemented in each state are divided several diagrams.

The *SporadicInfo* consist of:

- **diagram_SporadicInfo_Main**: Contains all the modules to manage the sporadic data like "Enter revalidate Driver ID", "Enter or revalidate train running number", enable buttons in menus. The WindowSM state machine manages the windows that should appear on the DMI(Fig. 16).
- **diagram_SporadicInfo_TrainData**: Contains all the logic to store and adapt the incoming train data to a correct visualization on DMI Display.
- **diagram_SporadicInfo_Icon_Management**: Contains the logic to show/hide one or several icons in area and manage the acknowledgement mechanism if It's required.

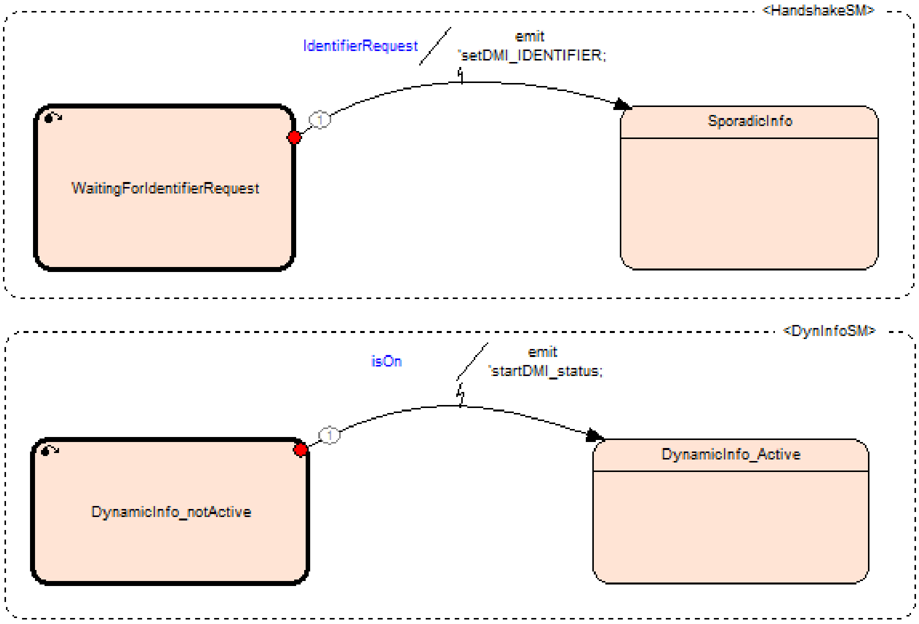


Figure 15. HandshakeSM and DynamicInfoSM State Machines.

- **diagram_SporadicInfo_DriverID_TRN**: Contains the logic to store and sent the Train running number and the Driver ID.
- **diagram_SporadicInfo_Text_Messages**: Contains the modules, state machines and all the logic to manage and display predefined and customized text messages.

The *DynamicInfo_Active* state consists of:

- **diagram_DynamicInfo_Main**: Contains modules to store and display the informations like the current mode, ETCS level, RBC connection status and location brake target.
- **diagram_SpeedSupervision**: Contains the module where are implemented the behaviour of the speed pointer and the circular speed gauge (informations about speed target, speed permitted and speed release).

12.1.0.10 Communication Protocol

This section explains which messages are exchanged among DMI Controller, DMI Management and Start of mission procedure. As mentioned previously the DMI Controller is a passive component, It simply responds to requests, therefore is able to cover different scenarios. Below are some examples.

12.1.0.10.1 Start Of Mission scenario

Are detailed, through a sequence diagram, all the activities (exchanged messages) that should be done to start. In this scenario we have three actors: DMI Controller, DMI Management and SoM procedure (the module where is implemented the start of mission procedure). It's assumed that a OpenDesk signal is received and the system starts in Stand By mode (Fig. 17).

12.1.0.10.2 Cyclic Exchange of messages

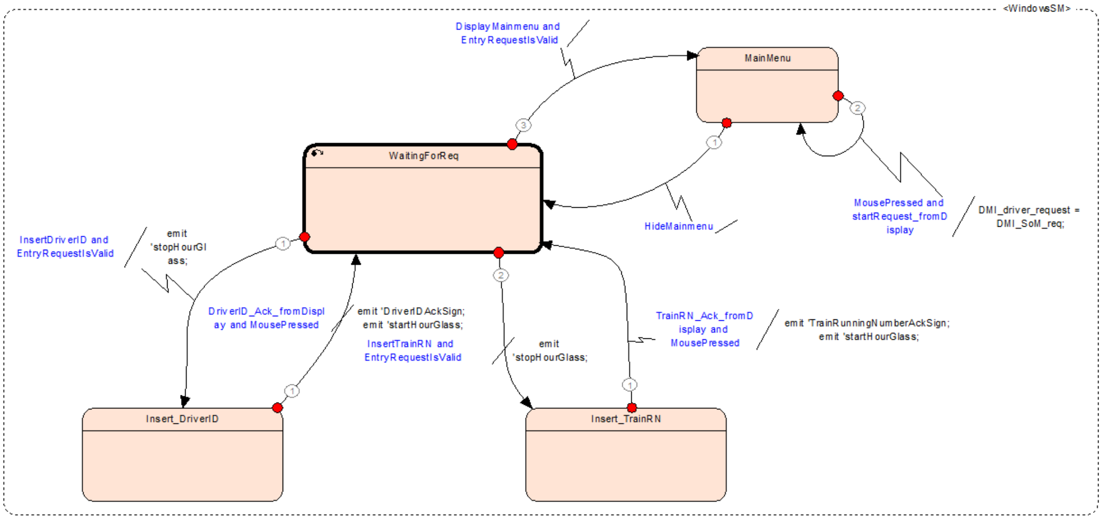

<WindowsSM>
Display Mainmenu and EntryRequestIsValid
MainMenu
WaitingForReq
HideMainmenu
MousePressed and startRequest_fromDisplay
DMI_driver_request = DMI_SoM_req;
InsertDriverID and EntryRequestIsValid
emit 'stopHourGlass;
DriverID_Ack_fromDisplay and MousePressed
emit 'DriverIDAckSign; emit 'startHourGlass;
InsertTrainRN and EntryRequestIsValid
emit 'stopHourGlass;
TrainRN_Ack_fromDisplay and MousePressed
emit 'TrainRunningNumberAckSign; emit 'startHourGlass;
Insert_DriverID
Insert_TrainRN
1
2
3

Figure 16. Windows state machine.

The time between two messages has not yet been definitively established, It might change in the future. The DMI status packet implements a keep alive mechanism, this means, if the EVC does not receive any DMI status signal during the lapse time, It shall consider a failure in DMI. This check is not yet implemented.

12.1.0.11 Reference to the Scade Model

The SCADE model can be found on github under the following path: `https://github.com/openETCS/modeling/tree/master/model/Scade/System/DMI_Control`

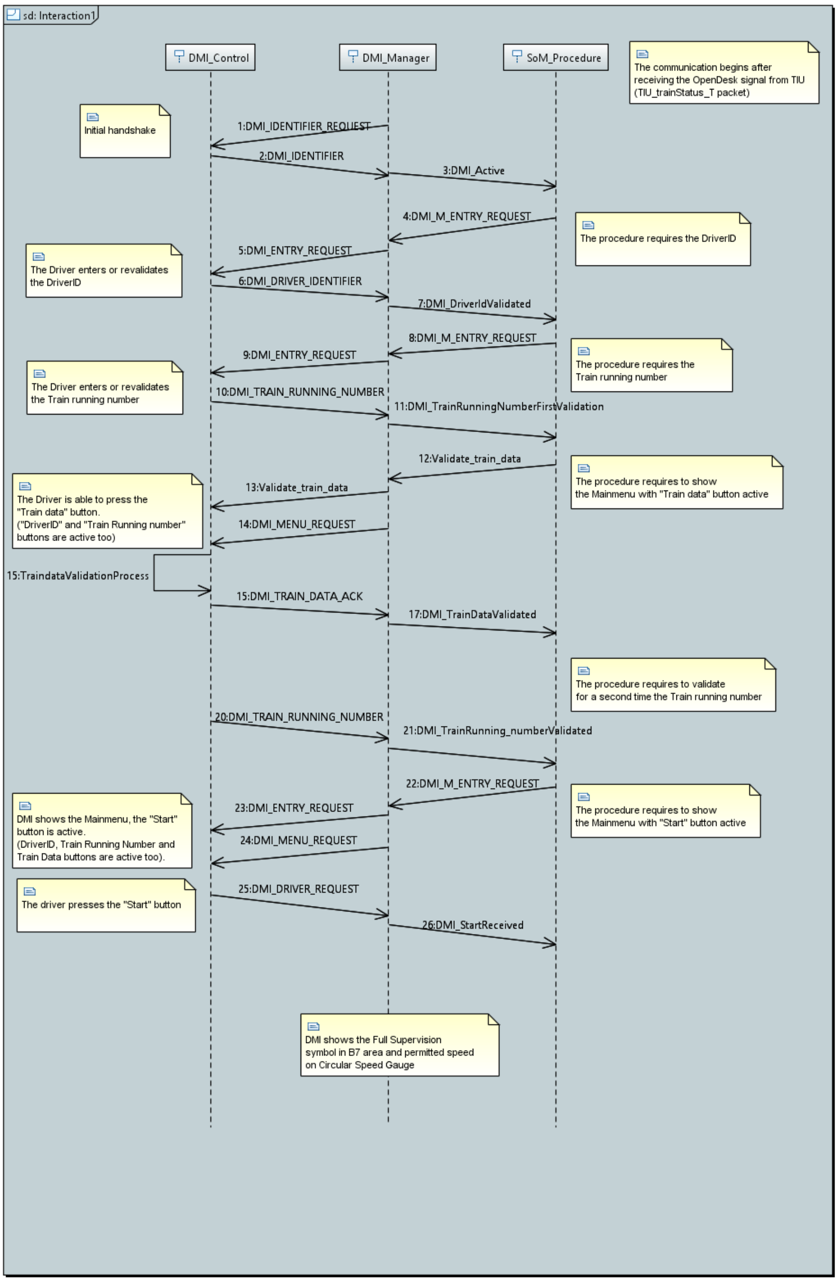

sd: Interaction1
DMI_Control
DMI_Manager
SoM_Procedure
The communication begins after receiving the OpenDesk signal from TIU (TIU_trainStatus_T packet)
Initial handshake
1:DMI_IDENTIFIER_REQUEST
2:DMI_IDENTIFIER
3:DMI_Active
4:DMI_M_ENTRY_REQUEST
The procedure requires the DriverID
5:DMI_ENTRY_REQUEST
The Driver enters or revalidates the DriverID
6:DMI_DRIVER_IDENTIFIER
7:DMI_DriverIdValidated
8:DMI_M_ENTRY_REQUEST
9:DMI_ENTRY_REQUEST
The procedure requires the Train running number
The Driver enters or revalidates the Train running number
10:DMI_TRAIN_RUNNING_NUMBER
11:DMI_TrainRunningNumberFirstValidation
12:Validate_train_data
The procedure requires to show the Mainmenu with "Train data" button active
13:Validate_train_data
The Driver is able to press the "Train data" button. ("DriverID" and "Train Running number" buttons are active too)
14:DMI_MENU_REQUEST
15:TraindataValidationProcess
15:DMI_TRAIN_DATA_ACK
17:DMI_TrainDataValidated
The procedure requires to validate for a second time the Train running number
20:DMI_TRAIN_RUNNING_NUMBER
21:DMI_TrainRunning_numberValidated
22:DMI_M_ENTRY_REQUEST
23:DMI_ENTRY_REQUEST
DMI shows the Mainmenu, the "Start" button is active. (DriverID, Train Running Number and Train Data buttons are active too).
The procedure requires to show the Mainmenu with "Start" button active
24:DMI_MENU_REQUEST
The driver presses the "Start" button
25:DMI_DRIVER_REQUEST
26:DMI_StartReceived
DMI shows the Full Supervision symbol in B7 area and permitted speed on Circular Speed Gauge

Figure 17. Sequence Diagram of start of mission scenario

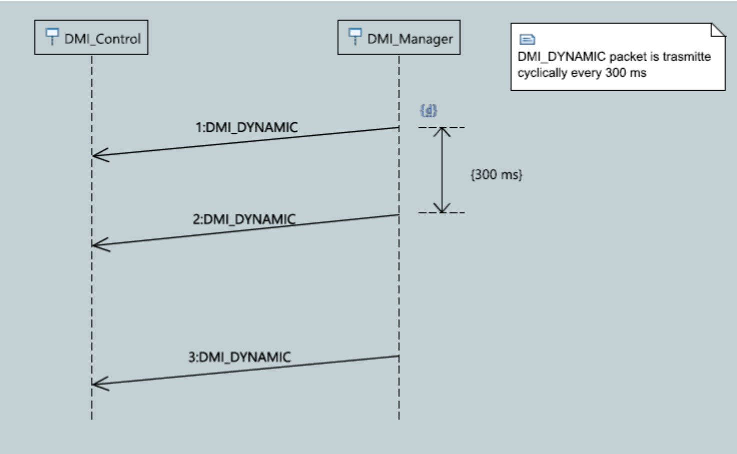


Figure 18. Sequence diagram of Dynamic data.

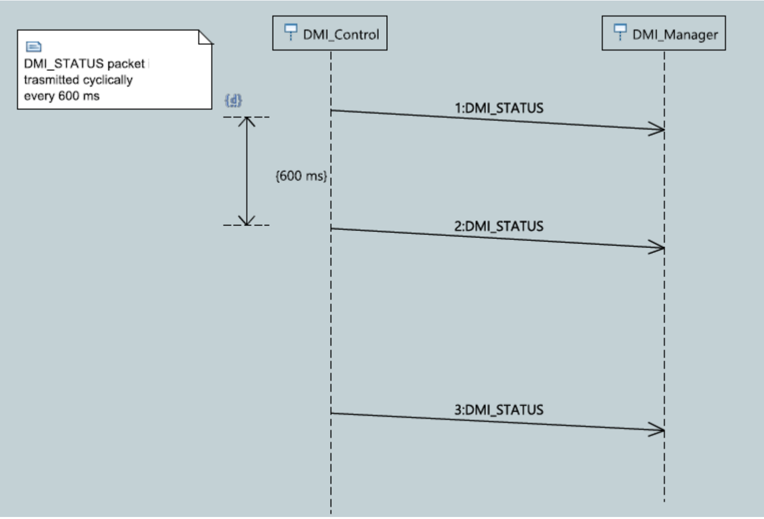


Figure 19. Sequence Diagram of DMI status.

References

[1] ERA. *System Requirements Specification, SUBSET-026*, v3.3.0 edition, March 2012.

[2] ERA. *FFFIS for Eurobalise, SUBSET-036*, v3.0.0 edition, February 2012.

[3] ERA. *Performance Requirements for Interoperability, SUBSET-041*, v3.1.0 edition, March 2012.