OETCS/WP4/DXX

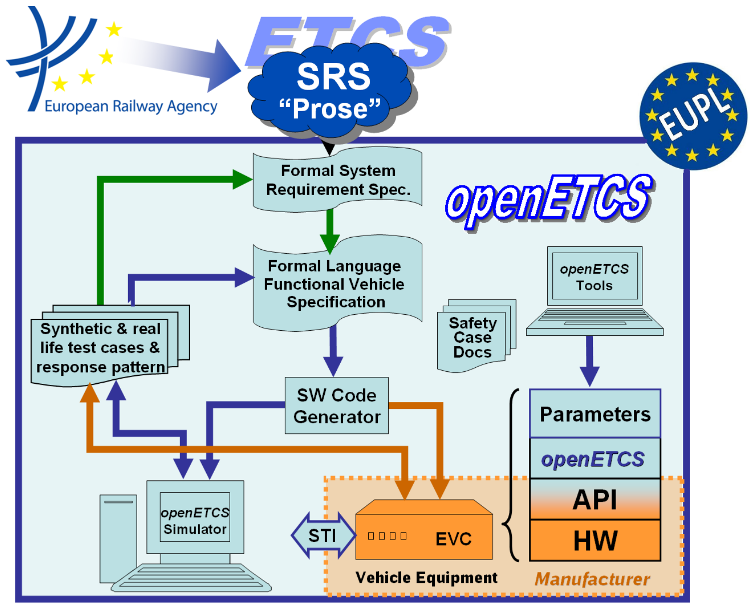
ITEA2 Project
Call 6 11025
2012 – 2015

Work-Package 4: V&V

Colored Petri Net Approach

Documentation of the V&V approach with colored Petri nets in openETCS by TWT

Christian Stahl, Stefan Rieger and Stephan Haas July 2015



Funded by:

Federal Ministry of Education and Research
Liberté • Égalité • Fraternité République Française Ministère de l'Enseignement Supérieur et de la Recherche
Investissements d'Avenir
Région de Bruxelles-Capitale
Gobierno de España Ministerio de Industria, Energía y Turismo

openETCS

This page is intentionally left blank

Work-Package 4: V&V

OETCS/WP4/DXX
July 2015

Colored Petri Net Approach

Documentation of the V&V approach with colored Petri nets in openETCS by TWT

Document approbation

Lead author:	Technical assessor:	Quality assessor:	Project lead:
location / date	location / date	location / date	location / date
signature [creator name] ([affiliation])	signature [assessor name] ([affiliation])	signature [assessor name] ([affiliation])	signature Klaus-Rüdiger Hase (DB Netz)

Christian Stahl, Stefan Rieger and Stephan Haas

TWT GmbH Science & Innovation
Ernsthaldenstrasse 17
70565 Stuttgart

Documentation of Work

Prepared for openETCS@ITEA2 Project

Abstract:

Modification History

Version	Section	Modification / Description	Author

Table of Contents

Figures and Tables

Figures

Tables

1 Introduction

We report on the modeling of the procedures described in Subset 026, Chapter 5—that is, the behavioral part of the ETCS. The goal of the activity is to validate the specification and to support the modeling using SCADE and the verification of SCADE models on a higher[1] level of abstraction.

The activity is described in the Verification and Validation Plan (see Sect. 6.1.2.5). In short, we provide feedback regarding ambiguities, inconsistencies and errors in the current ETCS standard based on our formalization of the specification using mathematical modeling languages.

The goal is to model the the procedures described in Subset 026-5, thereby focusing on modeling the *system behavior*—that is, the control flow of the on-board unit and the interplay with its environment (e.g., the driver and the RBC). The model is then used to validate the specification.

As a formal model, we use *colored Petri nets* (CPNs) [**?**], an extension of classical Petri nets [**?**] with data, time, and hierarchy. CPNs are well-established and have been proven successful in numerous industrial projects. They have a formal semantics and with CPN Tools [**?**], there exists an open source tool for modeling CPNs. Moreover, CPN Tools also comes with a simulation tool and a model checker, thereby enabling formal analysis of CPN models. We focus on modeling the *system behavior*—that is, the control flow of the on-board unit and the interplay with its environment (e.g., the driver and the RBC).

ToDo: explain the tool chain used

As the state space of our models is too large for the integrated CPN-Tools model checker, we transform the CPN-Tools output format into the LoLA format, in order to benefit from its high-performance model checker.

We continue by briefly introducing the procedures, describing their interplay and formalizing their data types in Section 2. Next, in Section 3, we introduce a CPN model for each individual procedure. We present modeling decisions, formalize the data abstraction and show what properties will be preserved in our models. Section 4 presents the properties we want to verify on the model and the analysis results.

2 Procedures of Chapt. 5

Chapter 5 of the ETCS specification describes the stateful behavioral description of the ETCS system. Such a behavioral description is a procedure. Most procedures deal with the behavior of the ETCS system when the mode has been changed.

2.1 Brief Introduction to the Procedures

There are 16 procedures specified in Chapter 5. We will briefly introduce them in the following.

Start of Mission This procedure starts the train. At the beginning the driver is asked to enter all necessary information for the train to run. Afterwards the operation mode of the train is determined. This procedure may directly lead to other procedures, such as "Shunting initiated by Driver" or Override.

[1] in comparison to SCADE models

End of Mission If there are active RBC-sessions or RIU-sessions the procedure End of Mission terminates them. Afterwards the train is probably at standstill.

Shunting Initiated by Driver

Entry in Shunting with Order from Trackside

Override This procedure checks whether the transition to "Staff responsible" mode is allowed or not. In this case the mode changes to "SR". This means, that the driver is responsible for any movement of the train and the ETCS solely controls, that the maximum speed is not surpassed.

On-Sight This procedure realizes the transition to the OS-mode of a moving train.

Level Transitions

Train Trip This procedure trips the train, meaning that it activates the emergency brake.

Change of Train Orientation

Train Reversing

Joining / Splitting

RBC/RBC Handover

Procedure Passing a Non-protected Level Crossing

Changing Train Data from Sources Different from the Driver This procedure is called, when trackside devices need to change train data.

Indication of Track Conditions

Limited Supervision This procedure realizes the transition to the LS-mode of a moving train.

2.2 Interplay of the Procedures

The interplay of the procedures is best illustrated using a behavioral model. Figure 1 depicts the interplay as a colored Petri net.

In Fig. 1, a procedure is modeled as a place[2] which is depicted as an ellipse. The name of the procedure is depicted inside the place. A transition from one procedure to another one is modeled as a transition, which is depicted as an rectangle. Again, the description of the transition is depicted inside the rectangle. In addition to the procedures there is one more place, named *Driving*, which is used to model an intermediate state from which other procedures can be called.

ToDo: What do we see in Fig. 1?

2.3 Data Used in the Procedures

In this section, we formalize the relevant data types for Chapter 5.

The OBU can be in 17 possible modes, see Chapter 4.3.2.1. This can be defined as a set *Mode* with $Mode = \{FS, LS, OS, SR, SH, UN, PS, SL, SB, TR, PT, SF, IS, NP, NL, SN, RV\}$.

[2] Better use a substitution transition.

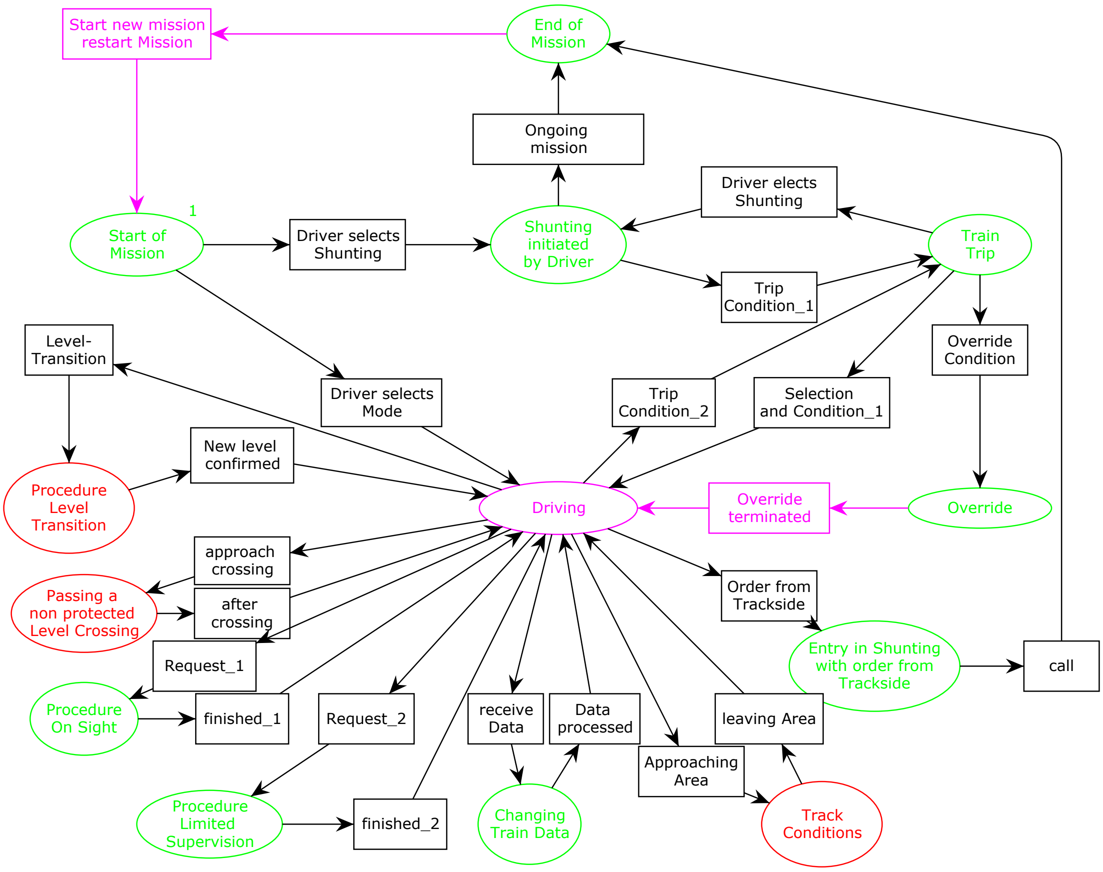


Figure 1. CPN modeling the interplay of the procedures.

There are five possible ETCS levels, i.e., $ETCS\_Level = \{0, 1, 2, 3, NTC\}$.

To communicate with its environment, the OBU sends messages to the trackside and receives messages from trackside. A message is of a particular type whereby the type is identified by a number and a description, i.e., a String. Each message type is a complex data type. We abstract from the concrete data type and define the type *MessageType*. We define messages from and to the driver as well as messages from and to the RBC: $MessageToDriver = MessageType$, $MessageFromDriver = MessageType$, $MessageToRBC = MessageType$, and $MessageFromRBC = MessageType$.

The Train Position Data defines the position of the train front in relation to a balise group. Train Data contains information about the train, such as its length and braking parameters. The Train Running Number and the Driver ID are identifiers. The RBC ID is the phone number of a RBC. The Virtual Balise Cover (VBC) contains a list of balises and a period of validity. However, for most data stored in the OBU we only need the status rather than the concrete value. We formalize the status using the data type $DATASTATE = \{valid, invalid, unknown\}$. To this end, the specification defines the following types, each of them of type *DATASTATE*:

DriverID, *Level*, *RBC_ID*, *TrainData*, *TrainRunningNumber*, *TrainPosition*, *VirtualBaliseCover*, *RadioNetworkID*.

2.3.1 Start of Mission

We list the relevant data types for the procedure.

While executing the procedure, the OBU communicates with the driver and the RBC. In addition to the mode and the ETCS level, the status of the following data is relevant: *DriverID*, *ETCS_Level*, *RBC_ID*, *TrainData*, *TrainRunningNumber*, *TrainPosition*, *VirtualBaliseCover*, and *RadioNetworkID*.

Furthermore, the procedure checks whether the desk of the driver is open or closed, accesses and modifies the set of active RBC sessions as well as the mobile terminals that are registered to the radio network.

To start a mission, the OBU has to be in mode *SB*.

2.3.2 End of Mission

We list the relevant data types for the procedure.

While executing the procedure, the OBU communicates with the RBC. In addition to the ETCS level, the status of the following data is relevant: *ETCS_Level*, *TrainData*, *RIU_Session_Active* and *RBC_Session_Active*

2.3.3 Shunting Initiated by Driver

While executing the procedure, the OBU communicates with the RBC. In addition to the ETCS level and the mode, the status of the following data is relevant: *STMtripprocedure*, *TrainPositionData*.

2.3.4 Entry in Shunting with Order from Trackside

$Mode_{SOT} = \{FS, LS, OS, SR, SB, PT, UN, SN\} \subset Mode$

2.3.5 Override

We list the relevant data types for the procedure.

While executing the procedure, the OBU communicates with the driver. In addition to the mode and the ETCS level, the status of the following data is relevant: *TrainData*

The mode is in $Mode_{Override} = \{FS, LS, OS, SR, SH, SB, PT, UN, SN\} \subset Mode$

2.3.6 On-Sight

While executing the procedure, the OBU communicates with the driver.

$Mode_{OS} = \{FS, LS, OS, SR, SB, PT, UN, SN\} \subset Mode$

2.3.7 Level Transitions

2.3.8 Train Trip

While executing the procedure, the OBU communicates with the driver and the RBC. In addition to the mode and the ETCS level, the status of the following data is relevant: *TrainData*

2.3.9 Change of Train Orientation

2.3.10 Train Reversing

2.3.11 Joining / Splitting

2.3.12 RBC/RBC Handover

2.3.13 Procedure Passing a Non-protected Level Crossing

2.3.14 Changing Train Data from Sources Different from the Driver

$Mode_{CTD} = \{FS, LS, OS, SR, SB, PT, UN, SN, SH\} \subset Mode$

2.3.15 Indication of Track Conditions

2.3.16 Limited Supervision

While executing the procedure, the OBU communicates with the driver.

$Mode_{LS} = \{FS, LS, OS, SR, SB, PT, UN, SN\} \subset Mode$

3 CPN Models

3.1 Design decisions

We do not model unidirectional messages, as this would result in a possible unbounded model or would cause modeling overhead to delete those messages immediately.

3.2 Data abstraction

Instead of considering all possible active RBC session, we only consider whether there exists at least one active RBC session or not. Thus, we model this as a Boolean $ActiveRBCSession = \{true, false\}$. Similar we abstract from the information of the mobile terminal being registered to a radio network and model it as a Boolean $MobileTerminalRegisteredToRadioNetwork = \{true, false\}$.

We further abstract from concrete messages—that is, the content of a message. To identify which message type has been sent by the OBU to the driver or the RBC, we abstract those messages to a String, i.e., $MessageToDriver = MessageToRBC = String$. A message from the RBC is abstracted to an undistinguishable black token, i.e., $MessageFromRBC = UNIT$. From a message sent by the driver, we are only interested whether or not the driver conforms a request; thus, the abstracted data type is of type Boolean, i.e., $MessageFromDriver = Bool$. Decisions based on on received data are modeled using a nondeterministic choice. As result, our model is a safe over-approximation of the concrete system.

3.3 Property preservation

If there exists a path in the concrete system, then there also exists a path in the model (but not necessarily the other way around).

3.4 Start of Mission

3.4.1 Assumptions

3.4.2 Interface

Figure 2 shows the top level model of Start of Mission and its environment.

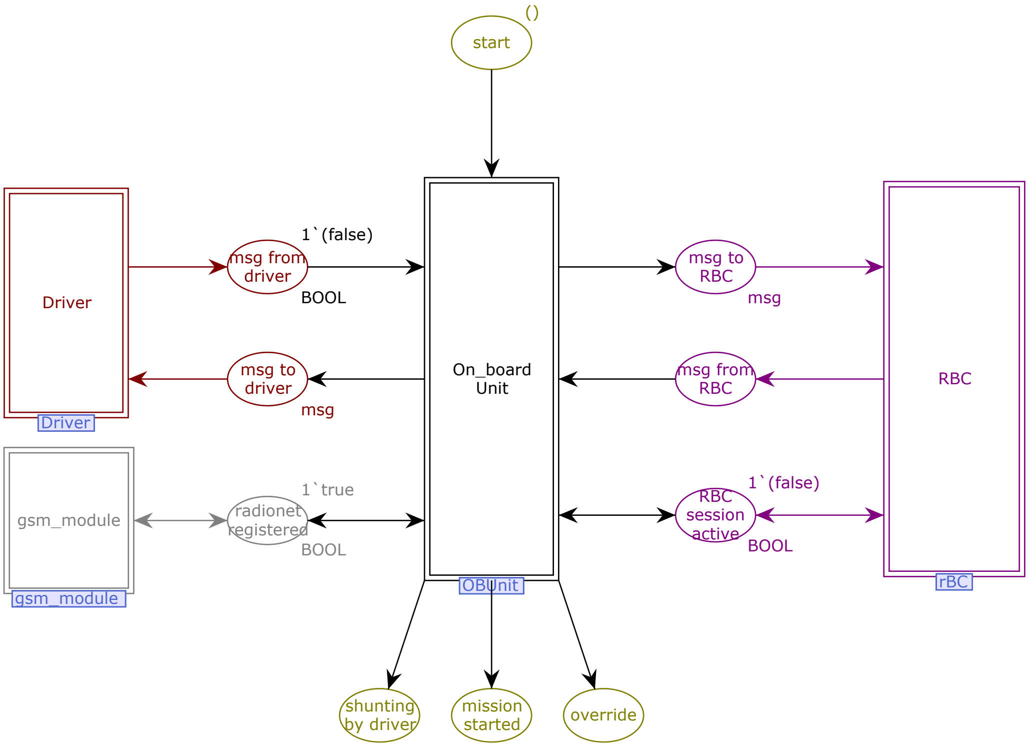


Figure 2. Top level model of Start of Mission and its environment

We model an interface to the driver and to the RBC. In addition, *MobileTerminalRegisteredToRadioNetwork* serves as an interface to the GSM module. The following places model the interface:

place "msg to driver" *MessageToDriver*

place "msg from driver" *MessageFromDriver*

place "msg to RBC" *MessageToRBC*

place "msg from RBC" *MessageFromRBC*

place "RBC session active" *ActiveRBCSession*

place "radionet registered" *MobileTerminalRegisteredToRadioNetwork*; interface to GSM module

The interface to other components (i.e., Petri nets modeling procedures other than Start of Mission) consists of four places, one input and three output places:

place "start" input place of the component

place “shunting by driver” output place to invoke procedure Shunting initiated by driver

place “mission started” output place signaling that the mission has started

place “override” output place to invoke procedure Override

3.4.3 Data used

The following data is modeled using a data place:

place “virtual balise cover” *VirtualBaliseCover*

place “driver_id” *DriverID*

place “mode” *Mode*

place “train_rno” *TrainRunningNumber*

place “ETCS State” *Level*

place “train position data” *trainPositionData*

place “ETCS level” *ETCS_Level*

place “rbc_id” *RBC_ID*

place “train data” *TrainData*

place “Radio Network ID” *RadioNetworkID*

place “desk(s) open” *Boolean*, models whether the desk is open

place “RBC session active” *ActiveRBCSession*

place “radionet registered” *MobileTerminalRegisteredToRadioNetwork*

Figure 3 shows the first level of the model Start of Mission. The places colored green are the data places.

3.4.4 Initial marking

The current mode is *SB*.

3.4.5 State space analysis

First of all, the net is bounded and there are no deadlocks, except the desired exit-states.

3.5 End of Mission

3.5.1 State Space Analysis

In order to analyze the subnet in all possible scenarios, we randomly choose the initial marking.

If we assume, that a RBC-session can only be active if the ETCS-Level is 2 or 3, then the net is deadlock free. The exception of this deadlock freedom is the desired "end" state. The net doesn’t contain any dead transitions. (if we assume, that any initial marking is possible when entering the procedure.)

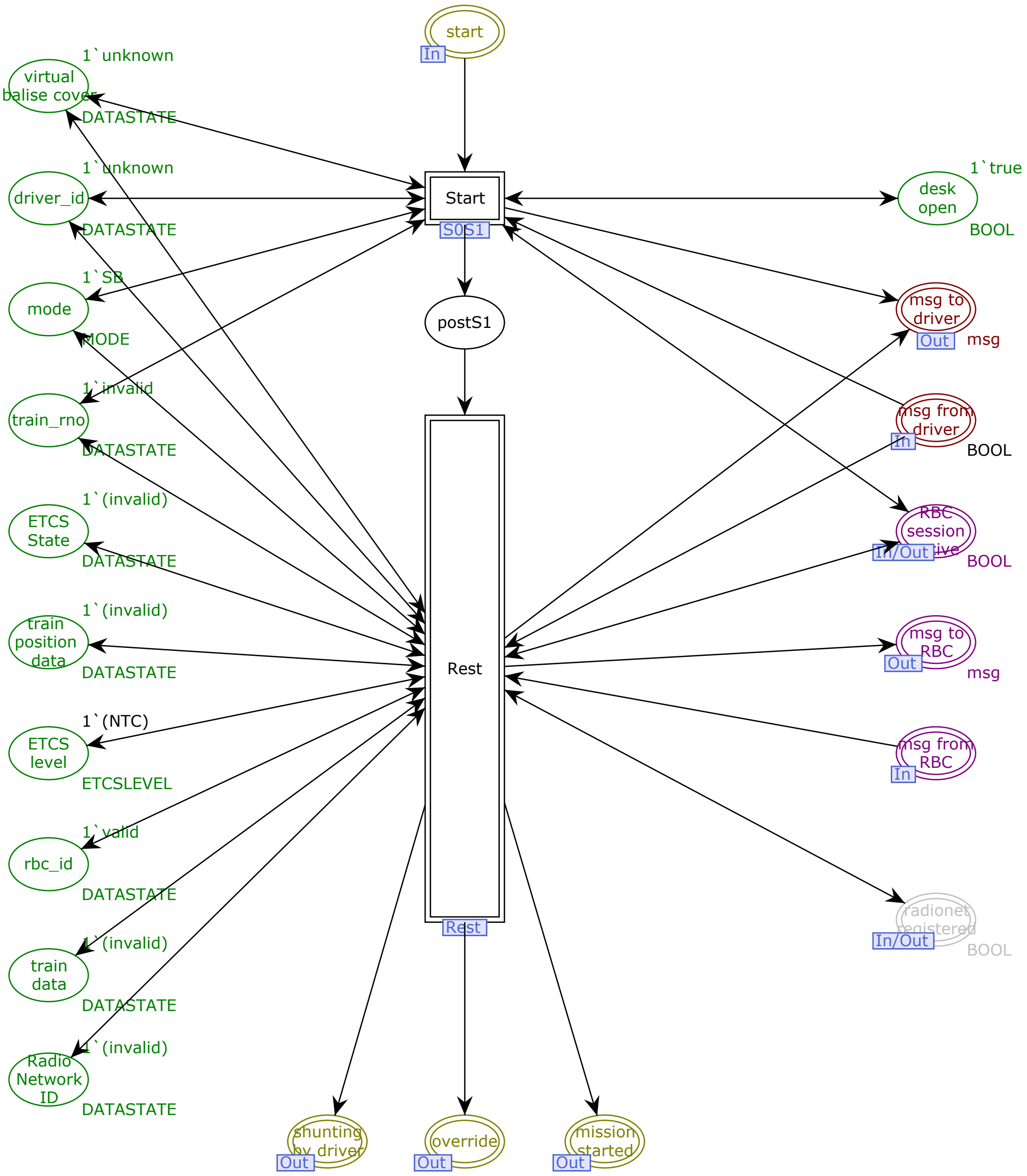


Figure 3. First level model of Start of Mission—interface place are colored green.

3.6 Shunting Initiated by Driver

It is not stored yet whether there is an ongoing mission.

3.7 Entry in Shunting with Order from Trackside

3.8 Override

If we decided to model movement authorities, then, in level 2 and 3, there would be communication with the RBC where messages can be lost.

3.8.1 initial marking

The Train Data is valid. The mode is Full Supervision, Limited Supervision, On Sight, Staff Responsible, Shunting, Unfitted, Post Trip, Stand By (in level 2/3 only) or SN. All other variables are chosen randomly, thus ensuring that every possible scenario is covered.

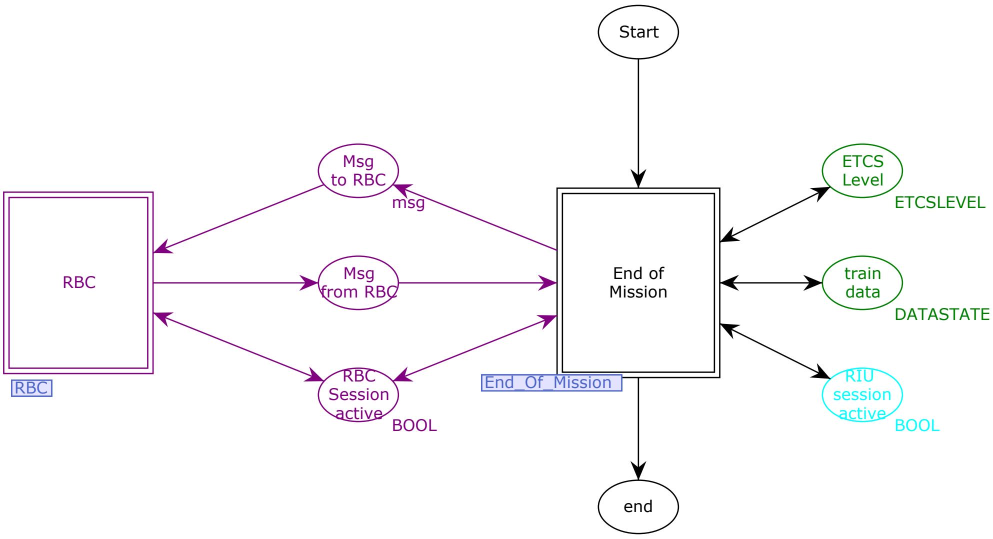


Figure 4. Top level model of End of Mission and its environment

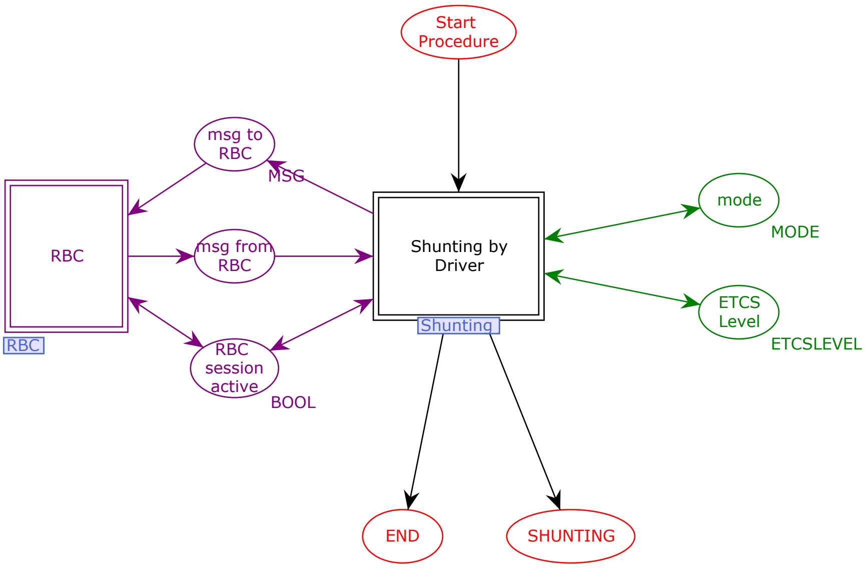


Figure 5. Top level model of Shunting Initiated by Driver and its environment

3.8.2 State space analysis

The state space contains 212 nodes and 418 edges. There are no deadlocks and the net is bounded.

3.9 On-Sight

3.10 Level Transitions

3.11 Train Trip

3.12 Change of Train Orientation

3.13 Train Reversing

3.14 Joining / Splitting

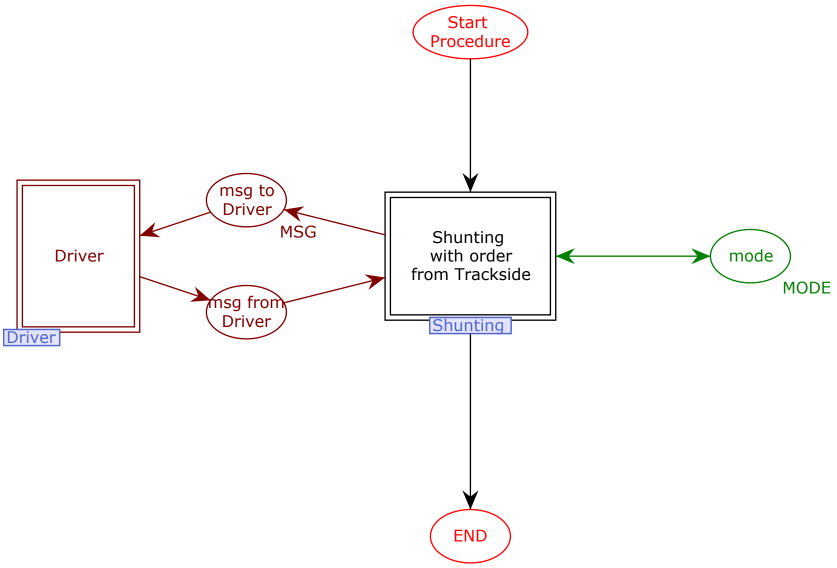


Figure 6. Top level model of Shunting with Order from Trackside and its environment

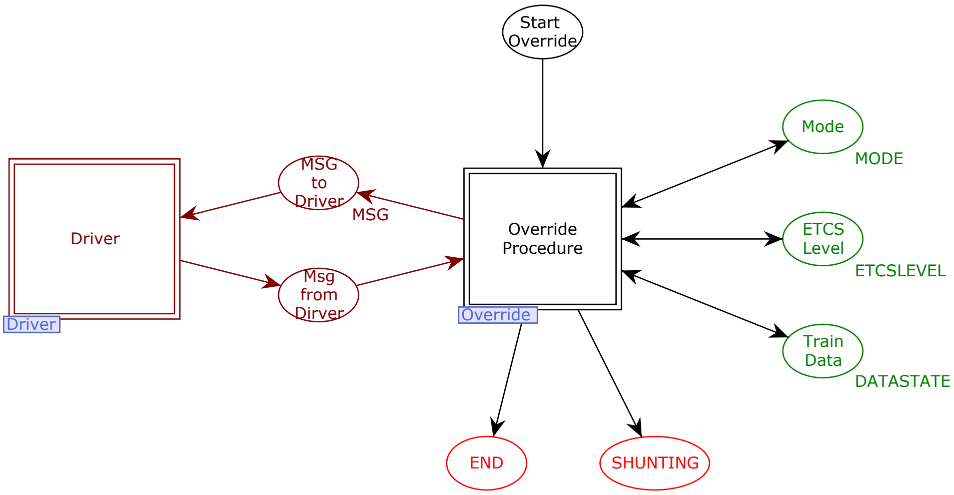


Figure 7. Top level model of Override and its environment

3.15 RBC/RBC Handover

3.16 Procedure Passing a Non-protected Level Crossing

3.17 Changing Train Data from Sources Different from the Driver

3.18 Indication of Track Conditions

3.19 Limited Supervision

4 Validation of the specification

4.1 Specification findings

4.2 Considered scenarios/properties

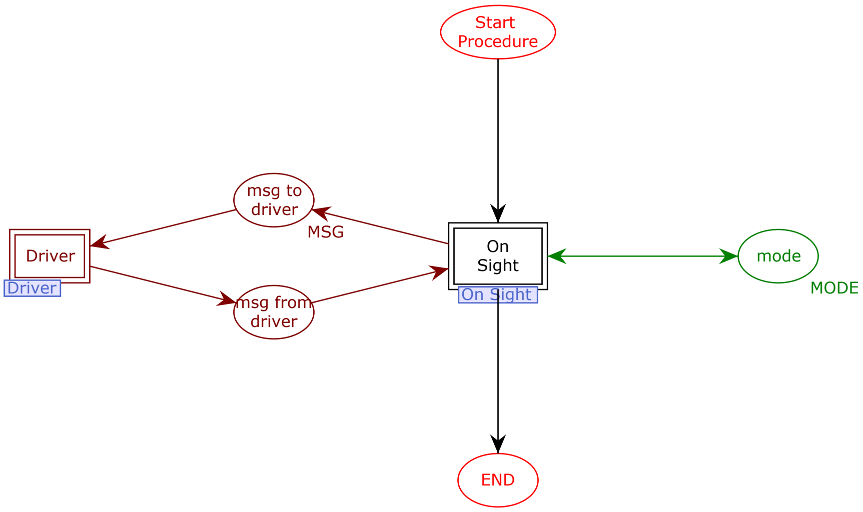


Figure 8. Top level model of On Sight and its environment

5 Detected mistakes by model checking

1. By running the deadlock-test of LoLA, we detected an error in S20. The degraded situation in which the driver selected NL-mode was not handled correctly. Instead of considering the mission as started, once the driver selected NL mode, we continued the procedure Start of Mission, which led to a deadlock, because no change from SB to oher modes could be performed, since the mode had already changed to NL.

2. By comparing our model to the belgian model, we noticed, that we had in D3 the decision based on the ETCS level wrong.

3. By running the deadlock-test of LoLA, we noticed, that we forgot to check both the availability of train position data AND the ETCS-state in D2.

4. By running the deadlock-test of LoLA, we noticed that we forgot to revert the change of mode (from SB to Sh) in S10 and S20, in case, that the RBC rejected the entry in Shunting mode.

References

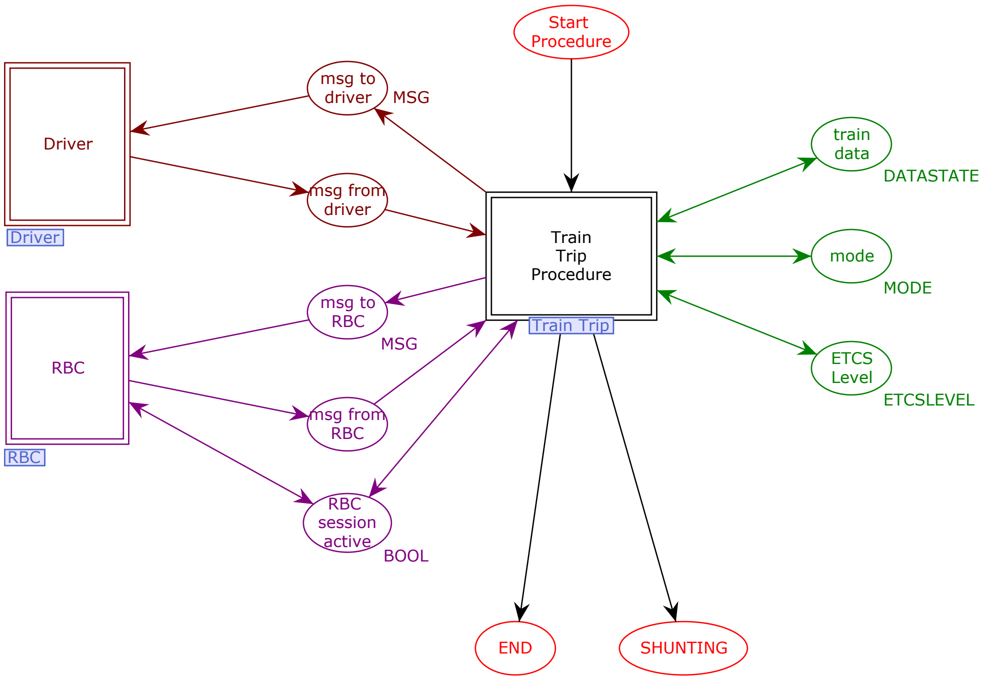


Figure 9. Top level model of Train Trip and its environment

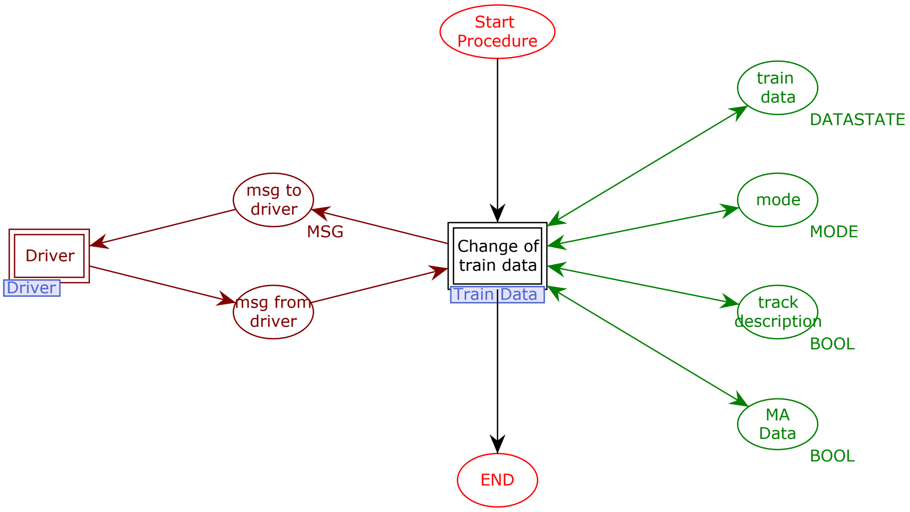


Figure 10. Top level model of Change of Train Data and its environment

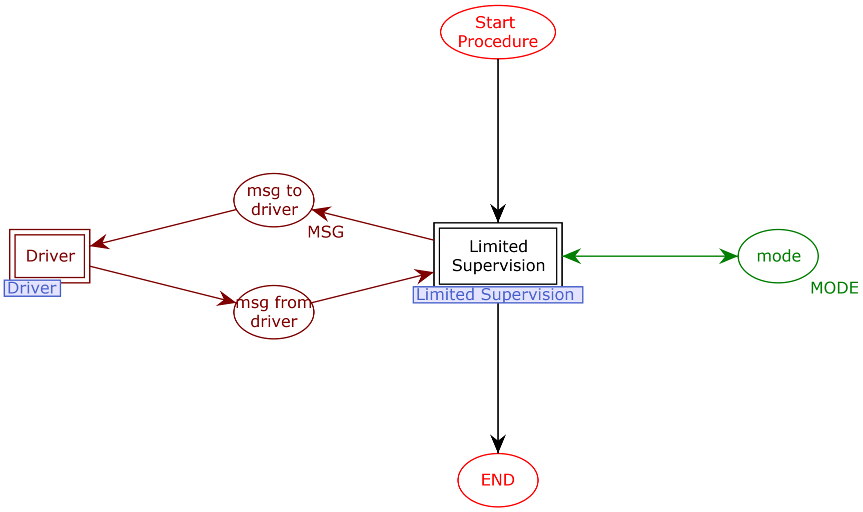


Figure 11. Top level model of Limited Supervision and its environment