Deliverable D2.3-Moving Block Verification and Validation

R. Saddem-Yagoubi, Julie Beugin, Mohamed Ghazel, S. Marrone, B. Janssen, F. Flammini, C. Seceleanu, U. Sanwal, M. Benerecetti, S. Libutti, et al.

Deliverable D2.3
Moving Block Verification and Validation

Project acronym:	PERFORMINGRAIL
Starting date:	01/12/2020
Duration (in months):	31
Call (part) identifier:	S2R-OC-IP2-01-2020
Grant agreement no:	101015416
Due date of deliverable:	Month 20
Actual submission date:	19/09/2023
Responsible/Author:	Mohamed Ghazel
Dissemination level:	PU
Status:	Resubmitted

Reviewed: no

Document history		
Revision	*Date*	*Description*
0.1	22/11/2022	First issue for internal review
1.0	30/11/2022	Submission to EU
1.1	19/09/2023	Revised version due to EU comments

Report contributors		
Name	Beneficiary Short Name	Details of contribution
Stefano Marrone	CINI	Contribution in structuring the deliverable; Section 1.3; Chapter 2
Bob Janssen	EULYNX	Contribution in refining operational scenarios.
Julie Beugin Mohamed Ghazel	UNI EIFFEL	Contribution in structuring the deliverable, Executive Summary, Section 1.1, Section 3.1, Section 4.3
Rim Saddem (ex UNI EIFFEL)	UNI Aix Marseille	Contribution in structuring the deliverable, Section 3.1, Section 4.3, Section 5.1
Francesco Flammini Cristina Seceleanu Usman Sanwal	MDH	Section 1.2, Section 3.2, Section 4.2 , Section 4.4, Section 5.1.3
Massimo Benerecetti Simone Libutti Elena Napolitano Fabio Mogavero Roberto Nardone Adriano Peron Valeria Vittorini	CINI	Section 3.3, Section 3.4, Section 4.1, Section 4.5, Section 4.6, Contribution in Section 5.1, Section 5.2
Joelle Aoun	TUD	Contribution in analysing SAN model

Reviewers	
Name	*Company or Institution*
Nina Versluis	Technical University of Delft
Achila Mazini	University of Birmingham

Funding

This project has received funding from the Shift2Rail Joint Undertaking (JU) under grant agreement No 101015416. The JU receives support from the European Union's Horizon 2020 research and innovation programme and the Shift2Rail JU members other than the Union.

Disclaimer

Contents

Executive Summary

The present document constitutes the D2.3 - Moving Block Verification and Validation (D2.3), which is part of WP2 - Modelling and Analysis of Moving Block Specifications (WP2) of the project PERformance-based Formal modelling and Optimal tRaffic Management for movINGblock RAILway signalling (PERFORMINGRAIL).

D2.3 elaborates on the formal modelling activities that have been conducted based on three selected Operational Scenarios (OPSs) related to the Moving Block System (MBS). It also describes the analysis performed to investigate a number of properties relevant to these OPSs. The selection and relevance evaluation of Moving Block (MB)-related OPSs were achieved in Task 2.2 - Moving block system and scenarios characterization (T2.2) of PERFORMINGRAIL, and detailed in D2.1 - Modelling guidelines and Moving Block Use Cases characterization (D2.1). Particularly, the significance of the OPSs for the railway sector and the safety challenges involved were investigated.

The formal modelling work detailed in the present deliverable is based on the System Modelling Language (SysML) semiformal specifications of the MB system functional and behavioural aspects. Such specifications were derived in light of the elaborated modelling framework described in D2.2 - Moving Block Specification Development (D2.2), along with the preparatory activities for analysing formal properties and preliminary formal models. The formal modelling activities are the results of Task 2.4 - Formal Development for moving-block and virtual coupling train operations (T2.4). The semiformal specifications are related to eight selected ETCS Use Case (EUC)s and twelve Functional Component (FC)s, whose nine are on the Trackside side and three are On-board-related. These EUCs and FCs are relevant for modelling the global MB signalling system according to the following three selected OPSs: “Loss of Train Integrity”, “Points Control” and “Loss/Restore of Communications”. The Verification and Validation (V&V) activities described in the present deliverable report on the activities conducted in Task 2.5 - Verification and Validation of moving block systems (T2.5), which are related to the analysis of safety and functional features.

The formal models obtained in T2.4 allow us to depict the MB system behavioural aspects. The modelling activities were conducted using the modelling guideline defined in the project and the modelling approach defined in D2.2. The selected formalisms, namely extended Timed Automata (TA) and Stochastic Activity Networks (SANs), can cope with the analysis of quantitative and qualitative properties, especially those pertaining to safety.

This deliverable makes the following contributions:

- elaboration of modelling methodology to help elaborate the formal models, integrate them and implement the V&V process;
- development of formal models for different functional blocks;
- integration of the various models developed in a multi-partnership way
- development of formal models for performance evaluation;
- definition of a number of relevant functional, safety properties;
- carrying out V&V for the defined properties;
- carrying out a performance study through sensitivity analysis.

Abbreviations and acronyms

Abbreviation / Acronym	Description
4SECURAIL	FORmal Methods and CSIRT for the RAILway sector
AD	Activity Diagram
ALSP	Axle Load Speed Profile
ASTRAIL	SAtellite-based Signalling and Automation SysTems on Railways along with Formal Method and Moving Block validation
ATO	Automatic Train Operation
ATP	Automatic Train Protection
BDD	Block Definition Diagram
CCS	Control-Command and Signalling
CD	Class Diagram
CRE	Confirmed Rear End
CTL	Computation Tree Logic
D1.1	D1.1 - Baseline system specification and definition for Moving Block Systems
D1.2	D1.2 - Best practice, recommendations and standardisation to definition of the Railway Minimum Operational Performance Standards
D2.1	D2.1 - Modelling guidelines and Moving Block Use Cases characterization
D2.2	D2.2 - Moving Block Specification Development
D2.3	D2.3 - Moving Block Verification and Validation
D4.2	D4.2 - Guidelines for a safe and optimised moving-block traffic management system architecture
DoW	Description of Work
EBNF	Extended Backus–Naur Form
EC	European Commission
EGNSS	European Global Navigation Satellite System (Galileo & EGNOS)
ERTMS	European Railway Traffic Management System
ETCS	European Train Control System
ETCS-L2	European Train Control System - Level 2
ETCS-L3	European Train Control System - Level 3
EU	European Union
EUC	ETCS Use Case
EULYNX DP	EULYNX Data Preparation
EVC	European Vital Computer
EoA	End of Authority
EoM	End of Mission
FC	Functional Component
FM	formal methods
FS	Full Supervision
GNSS	Global Navigation Satellite System
HMI	Human-Machine Interface
HW	Hardware

IBD	Internal Block Diagram
IP2	Innovation Programme 2
LOS	Line Of Sight
LTI	Loss of Train Integrity
LTL	Linear Temporal Logic
MA	Movement Authority
MARTE	Modeling and Analysis of Real-Time and Embedded Systems
MARTE-DAM	Modeling and Analysis of Real-Time and Embedded Systems - Dependability Analysis and Modeling
MB	Moving Block
MBS	Moving Block System
MDE	Model-Driven Engineering
MOVINGRAIL	Moving Block and Virtual Coupling Next Generations of Rail Signalling
MaxSFE	Maximum Safe Front End
minSFE	Minimum Safe Front End
OBU	On-Board Unit
OCRA	Othello Contracts Refinement Analysis
OMG	Object Management Group
OPS	Operational Scenario
OS	On-Sight
PD	Package Diagram
PERFORMINGRAIL	PERformance-based Formal modelling and Optimal tRaffic Management for movINGblock RAILway signalling
PNs	Petri Nets
PVT	Position Velocity Time
RAT	Requirement Allocation Table
RBC	Radio Block Center
RD	Requirement Diagram
RSM	RailSystemModel
S2R	Shift2Rail
SAN	Stochastic Activity Network
SD	Sequence Diagram
SLR	Systematic Literature Review
SM	State Machine
SMC	Stochastic Model Checking
SMD	State Machine Diagram
SR	Staff Responsible
SSP	Static Speed Profile
STA	Stochastic Timed Automata
STPN	Stochastic Timed Petri Nets
SoM	Start of Mission
SysML	System Modelling Language
T1.3	Task 1.3 – Recommendations and standardisation to definition of the Railway Minimum Operational Performance Standards for moving block systems
T2.1	Task 2.1 - Modelling approach and guidelines
T2.2	Task 2.2 - Moving block system and scenarios characterization

T2.3	Task 2.3 - Specifications for safe and reliable moving-block signalling
T2.4	Task 2.4 - Formal Development for moving-block and virtual coupling train operations
T2.5	Task 2.5 - Verification and Validation of moving block systems
TA	Timed Automata
TIMS	Train Integrity Monitoring System
TLU	Train Localisation Unit
TMS	Traffic Management System
TPR	Train Position Report
TSA	Track Status Area
TSR	Temporary Speed Restriction
TTD	Trackside Train Detection
UC	Use Case
UCD	Use Case Diagram
UES	Unconditional Emergency Stop
UML	Unified Modelling Language
UTA	UPPAAL Timed Automata
VBD	Virtual Block Detector
VBF	Virtual Block Function
VC	Virtual Coupling
VSSs	Virtual Sub-Sections
VTD	Validated Train Data
V&V	Verification and Validation
WP	Work Package
WP1	WP1 - Specification for minimum MB performance
WP2	WP2 - Modelling and Analysis of Moving Block Specifications
WP3	WP3 - Fail Safe Train Locationing
WP4	WP4 - Integrated Moving Block architecture for safe and optimised traffic operations
X2Rail-1	Start-up activities for Advanced Signalling and Automation Systems
X2Rail-2	Enhancing railway signalling systems based on train satellite positioning, on-board safe train integrity, formal methods approach and standard interfaces, enhancing Traffic Management System functions
X2Rail-3	Advanced Signalling, Automation and Communication System (IP2 and IP5) – Prototyping the future by means of capacity increase, autonomy and flexible communication
X2Rail-5	Completion of activities for Adaptable Communication, Moving Block, Fail safe Train Localisation (including satellite), Zero on site Testing, Formal Methods and Cyber Security
XML	eXstensible Markup Language
XSD	XML Schema Definition

1. Introduction

1.1. Scope and Objectives

The main objective of the present deliverable (D2.3) is to report on the activities carried out within the framework of T2.5. Namely, a thorough description of the formal modelling activities will be provided. Then, the process of verifying of safety and functional properties will be exposed.

From the activities undertaken in WP1 - Specification for minimum MB performance (WP1) and Task 2.1 - Modelling approach and guidelines (T2.1) and T2.2 of WP2, it is plain to say that the specifications of the MBS show how much railway Control-Command and Signallings (CCSs) are complex, involving numerous interacting and communicating systems. Developing formal models for the whole system is obviously infeasible within the scope of the project. The main objective of the work undertaken within tasks T2.4 and T2.5 is to provide a proof of concept on how formal models can be derived for some parts of the behaviour of the MB system, and how formal methods can be advantageously brought into play to explore a number of behavioural aspects, namely safety, functional and performance features.

It is important to highlight that, although the scope of the formal modelling activities undertaken by the consortium within tasks T2.4 and T2.5 was limited to a number of functions and operational scenarios, this work showed to be very complex, being given the intensive interleaving between the various actors involved in the targeted behaviours in the modelling activities. Moreover, modelling strictly the behaviour of the involved actors would not be relevant, since they have to react to the stimulus they receive from their environment (other components, or the external system environment). Therefore, besides the models of the components that are included in the modelling scope, stricto sensu, it is necessary to elaborate further models that allow the emulation of these interactions. Another aspect that is worth mentioning is related to the fact that the formal modelling work was a multi-partnership activity. In fact, each partner that is involved in this activity has his own modelling practices and choices. Moreover, the various models have to exchange data and communicate via different signals, which required to find out some consensus in terms of notations, and modelling logic. All the aforementioned aspects made it even more challenging to integrate the different models that are developed by the various partners. Overall, the effort that was required to fulfil the work of tasks T2.4 and T2.5 exceeded by far the estimated effort during the project setting up phase.

The modelling and V&V activities are also characterized by a focus on the FCs, taken from the developed functional architecture (see details in the following), and these specific OPSs involving different FCs which interact with each others and with the environment. This will be further detailed in the various relevant sections:

- Trains Manager (FC)
- Communication Manager (FC)
- Points Control (FC)
- Track Status Manager (FC)
- Route Manager (FC)

- Points Management (OPS)
- Loss of train integrity (OPS)
- Loss/Restore of Communications (OPS) - considered twice according to two different modelling and analysis purposes.

1.2. Background

In a multi-modelling-based approach, the system under development is described by several models that represent various perspectives and concerns. Obviously, these partial representations are less complex than the global model, but they need to be composed to address verification and synchronization tasks. The *model composition* is a crucial model-driven development operation, but it remains a tedious and error-prone activity.

Assuming a component-based system design perspective, there are several approaches that address composition and reasoning of formal models of components, most being centred on the so-called *contract-based* reasoning [1, 2], also known as *assume-guarantee* verification. Traditional assume-guarantee reasoning methods rely on the notion of *contract*. In its basic form, contracts were introduced within the Eiffel programming language as a set of pre- and post-conditions between a caller and a method, and class invariants [3].

This idea is further extended in the form of *assume-guarantee* contracts, such that a component makes assumptions regarding its context, required to hold for the guarantees to be provided [1]. A contract is a pair of assertions $C =< A, G >$, where the component's assumptions on its environment are denoted by A, and the component's offered guarantees by G. Contract semantics are defined in terms of environments and implementations. It is said that an environment satisfies a contract $C =< A, G >$ if it provides all the contract assumptions A. An implementation satisfies a contract C, if provided that assumptions A hold, it satisfies the guarantees G.

Cimatti and Tonetta [2] propose a framework backed by tool support in Othello Contracts Refinement Analysis (OCRA), for the synchronous or asynchronous decomposition of a component into subcomponents, complemented with the corresponding refinement of its contracts. The properties to verify can be interpreted as Linear Temporal Logic (LTL) formulas, for instance, the response to a certain event.

Interface theories are an interesting alternative to assume-guarantee contracts, to be used in model composition. They aim at providing a merged specification of the implementations and environments associated with a contract via the description of a single entity, called an interface. Interface theories generally use (a mild variation of) Lynch Input/Output Automata [4] as their framework for components and environments. The first interface theory able to capture the timing aspects of components is *Timed Interfaces* [5]. Timed Interfaces allow specifying both the timing of the inputs a component expects from its environment and the timing of the outputs it can produce. Compatibility of two timed interfaces is then defined and refers to the existence of an environment such that timing expectations can be met.

Since in this deliverable the underlying assumption is that timed models describe the behaviour of components that have to exchange data and communicate via different signals, the employed composition approach relies exactly on identifying and defining the data and signals at the interface of the communicating components that form a subsystem, based on the existing Moving Block functional architecture. The respective subsystems need to meet functional but also timing requirements, which are then checked by verification by simulation

or model checking. The overall model composition approach is described in Section 4.2.

1.3. Relationships with other PERFORMINGRAIL deliverables

Fig. 1.1 represents the dependency between D2.3 and other PERFORMINGRAIL deliverables, highlighting deliverables that are input and others that are output for D2.3.

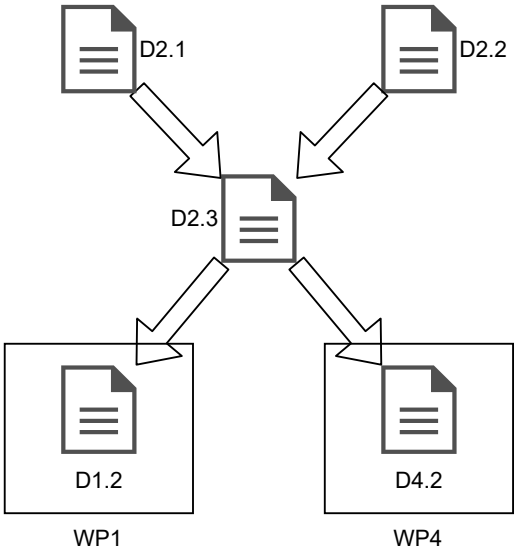


Fig. 1.1. Relationship between D2.3 and other deliverables.

Specifically:

- from D2.1 [6], the description of the OPSs are considered, to choose the slice of the European Train Control System - Level 3 (ETCS-L3) to specify, model and analyse in this deliverable;
- from D2.2 [7], this deliverable inherits the whole structure of the ETCS-L3 specification in SysML as well as the formal models that are here reviewed and expanded.

On the other hand, this deliverable will *directly* influence the following:

- the experiences and lesson learnt in modelling ETCS-L3 will be collected in D1.2 - Best practice, recommendations and standardisation to definition of the Railway Minimum Operational Performance Standards (D1.2);
- the specific results of analysis oriented to understand the impact of the different parameters on the capacity of an ETCS-L3 line will be useful for D4.2 - Guidelines for a safe and optimised moving-block traffic management system architecture (D4.2).

1.4. Structure of the deliverable

This deliverable is structured as the following. Chapter 2 describes the specification-modelling-analysis approach proposed in the deliverable. Chapter 3 describes the high-level specifications of the OPSs. Chapter 4 describes the results of the formal modelling effort and introduces the model library of formal models. Chapter 5 reports the results of the analysis of the described scenarios. Chapter 6 ends the deliverable, summarizing the contributions and discussing current limitations.

2. The Modelling Methodology

This chapter describes the approach related to the SysML specification of the OPSs. The chapter is structured as follows: Section 2.1 reports the key elements of the OPSs to model; Section 2.3 describes how to model OPSs according to the SysML model reported in [7] while Section 2.3.1 makes the discussion concrete by illustrating presented concepts by a small running example.

2.1. Key elements of an OPS

It could be useful to begin from the high-level description of a generic OPS, as reported in [6] and represented in Figure 2.1.

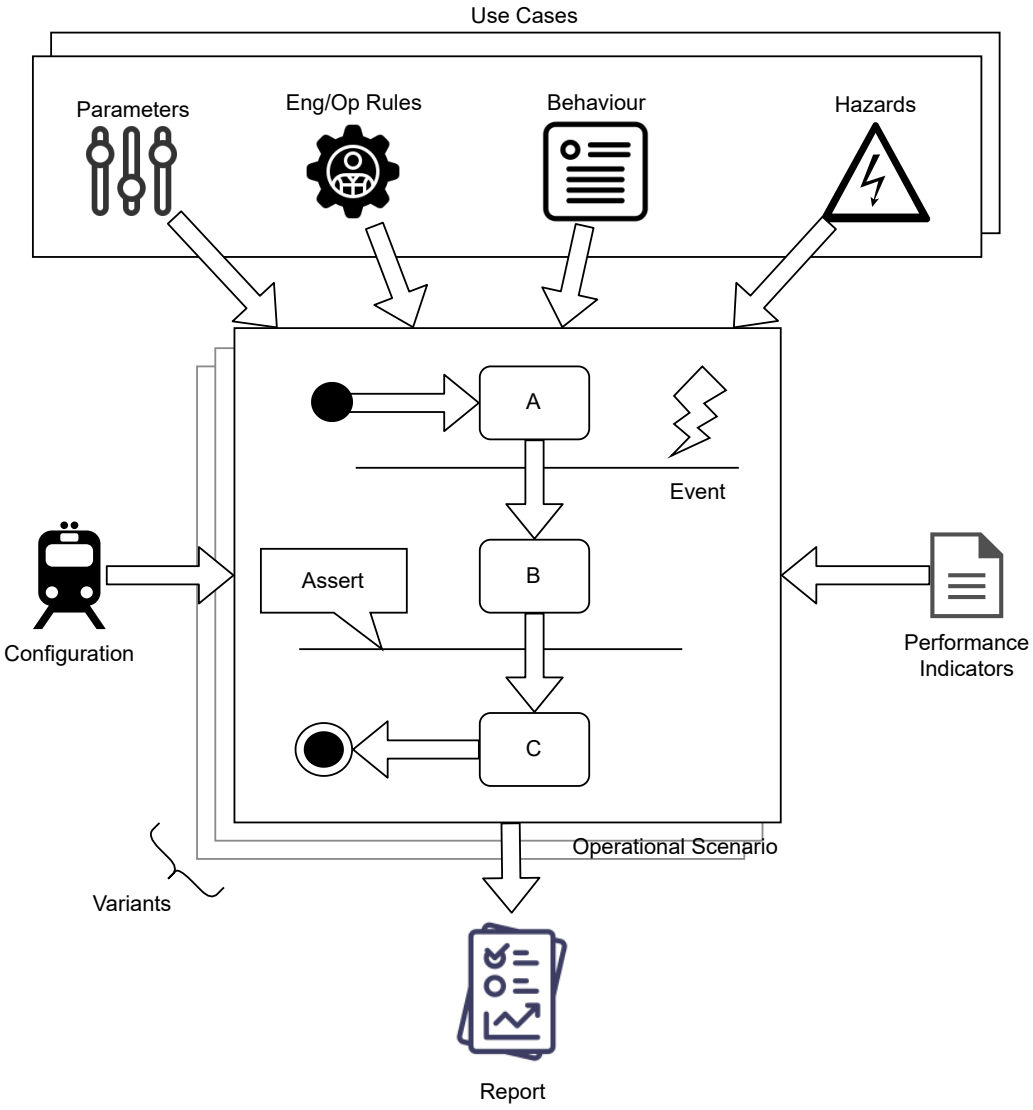


Fig. 2.1. High-level view of an OPS

The full introduction of the concept of OPS is reported in D2.1: in such a deliverable, all

the concepts presented in the figure are fully described. Here, the key elements needed to represent an OPS are summarized:

- *parameters*: representing variables whose value can be, in general, initially tuned, the different values a parameter can assume can affect the behaviour of the system;
- *behaviour*: describing the behaviour of the participating Hardware (HW) and functional components of the ETCS-L3;
- *steps*: they represent the single elementary acts an OPS performs (e.g., an interaction between functional components, a sending/receiving of a message, a change of the internal state of a functional components);
- *events*: they are internal or external acts that happen during the OPS (e.g., a breaking of a physical component, a receiving of a message from an external actor). They may induce the system to change its internal state;
- *branches*: since the system may usually have different evolutions, according to the specification of the use cases on which the OPS is based, the inclusion of alternative branches becomes necessary. A branch is one of the possible sequences of steps the system may follow, depending on one or more conditions that might occur;
- *asserts*: describing predicates that are expected to be true in certain points of the scenario. they may be: *pre- / post-conditions* (that are expected to be true before/after OPS's steps) or *invariants* (that are expected to be true in a portion/the whole of the OPS);
- *variants*: alternative versions of the scenario, where possible small changes do not affect the main behaviour of the scenario itself;
- *performance indicators*: are observable quantities that allow for evaluating whether operational scenarios reach the intended threshold. They may be logical, functional, availability/reliability, safety, performance.

2.2. The Overall Approach

The approach of the modelling and the analysis of the OPS can be summarised in Fig. 2.2.

The approach follows the one already defined in [7], related to the specification and of the modelling of the ETCS-L3. Similarly to that process model, in this approach there are both the phases of *specification* and *modelling*. The first clarifies the scope and the general aim of the OPS while the second is related to the formal modelling of the OPS.

Starting from this general similarity, some differences are instead present in both the activities, i.e., in the conducted sub-activities. Both the specification and modelling of the OPSs, in fact, start from existing SysML and formal models developed during the tasks Task 2.3 - Specifications for safe and reliable moving-block signalling (T2.3) and T2.4.

From the perspective of the specification, constructing a SysML model for an OPS means to simply specify a scenario in a model where behaviours, actors and functional components are already specified. In this context, a SysML model of an OPS can be seen as a different rearrangement of elements already present in the model.

On the other hand, the modelling approach aims at generating integrated formal models for each of the OPSs. Such integration does not start from scratch; a meaningful background is already present, considering the models developed during T2.4. Such models are in this task integrated among them.

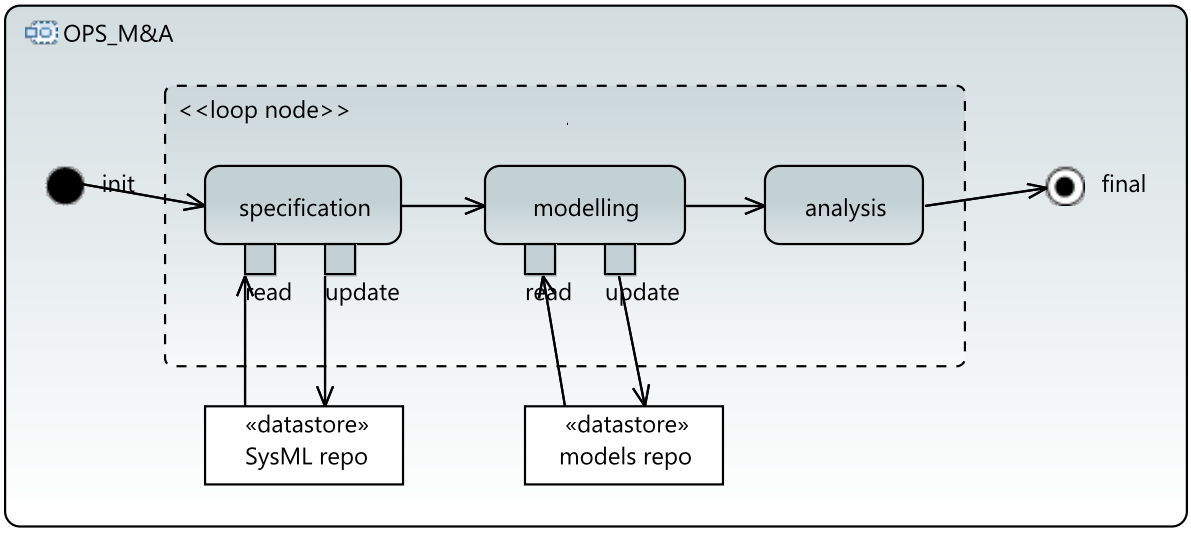


Fig. 2.2. OPS modelling and analysis: the followed approach

To conclude the approach followed in this deliverable, an analysis phase is conducted, taking the model developed during the modelling phase and verifying it against a property derived from the OPS specification.

It is important to underline that the scope of the modelling and analysis approach is not oriented to the construction of a global model claiming to represent all the possible behaviours of the ETCS-L3 systems. Such “one-above-all” formal model would be unfeasible not only during the analysis phase but also during its construction since the variability of such kinds of systems. One realistic objective, that constitutes the aim of the entire WP2, is instead to develop as many formal models as the number of considered OPS.

The entire approach would guarantee concreteness (due to its “scenario-driven” nature) without being too “custom” (due to the reuse of general specifications and models developed in the previous tasks).

The last similarity between the two general processes — the one presented in this deliverable and the one of [7] — is constituted by the iterative nature of the approach. Generally speaking, in a model-driven approach, models are prime citizens whose development never ends, but each iteration improves the results of the previous ones.

The details on the specification, modelling and analysis phases are reported respectively in Section 2.3, Section 4.2 and Chapter 5.

2.3. OPSs specification approach

According to the specification methodology described in [7], this section wants to provide some elements in the definition of a SysML model for the OPS. Fig. 2.3 reports the extension of the figure reported in [7]. That figure reports the main ETCS-L3 concepts considered in WP2 as well as the SysML concepts and diagrams used to depict such concepts.

Constructed on top of the previous one, Fig. 2.3 focuses on integrating the elements that are relevant for the OPS specification, that are:

- Entity: an OPS may involve ETCS-L3 external actors as Train Integrity Monitoring

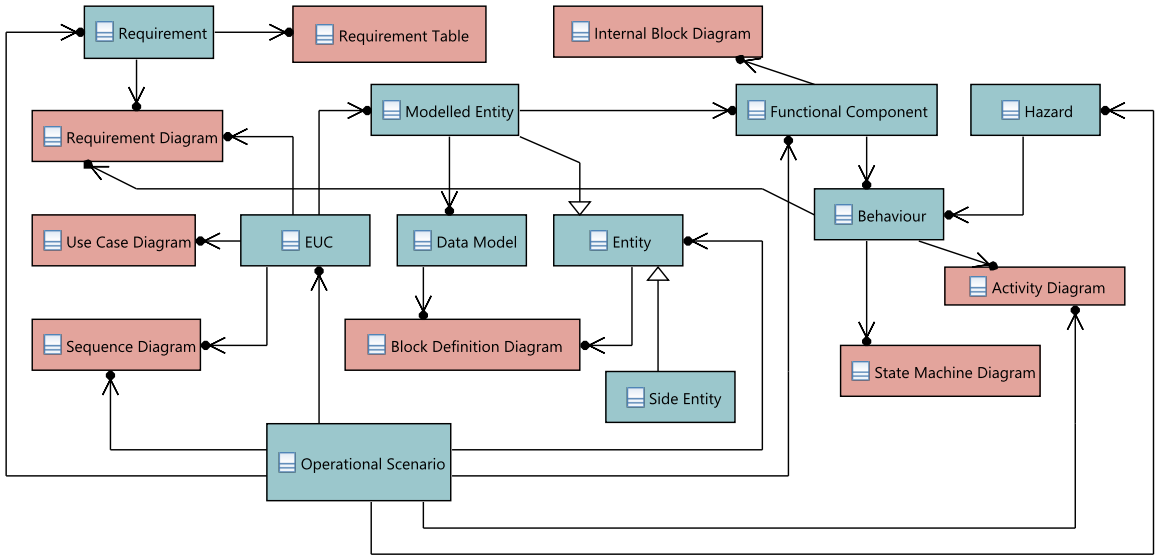


Fig. 2.3. OPS specification: an overview

System (TIMS) or the driver;

- Functional Component: an operational scenario mainly involves internal software functional components of both the trackside and/or the on-board;
- EUC: some EUCs can be partially covered by the sequence of the actions reported into an EUC;
- Hazard: some of the steps of the OPS can bring the system into one or more hazards;
- Requirements: the OPS is built considering one or more system requirements.

To this aim, both SysML's Sequence Diagrams (SDs) and SysML's Activity Diagrams (ADs) are candidate diagrams, able to represent an OPS; these SysML diagrams can be considered two of the best ones to highlight interactions between entities. The guidelines on how to construct both SDs and ADs are reported in the following paragraphs. No details on SysML are here reported: D2.1 describes all the basic elements of the language needed to understand this deliverable also reporting literature references [6].

Sequence Diagrams: Fig. 2.4 represents a generic SD, informally annotated with the concepts of ETCS-L3 of Fig. 2.3.

The figure clearly explains which of the language elements of the SD can be used. No formal specification approach is used for SysML, involving meta-modelling techniques as Unified Modelling Language (UML) profiles and/or Extended Backus–Naur Form (EBNF) grammars. Under this perspective, the parameters of the OPS are listed in a SysML's note as well as the assertions: both are written in natural language. Each lifeline is a FC or an HW element of ETCS-L3. The exchanges of messages are related to the call of a proper signal of the receivers. Self messages are internal calls of a FC while external events are reported in the model as UML's found messages. Branches in the evolution of the OPSs are represented by

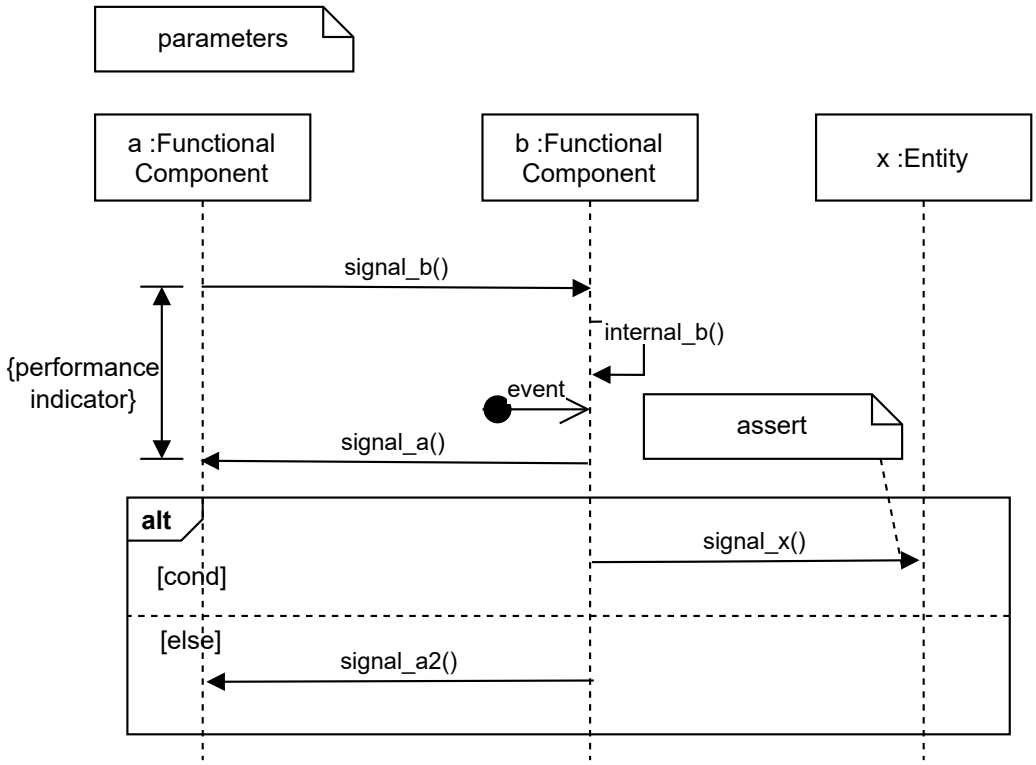


Fig. 2.4. OPS specification by means of SysML SD

alternative (`alt`) compound box, respectively reporting the condition and the default case.

Activity Diagrams: Fig. 2.4 represents a generic AD, informally annotated with the concepts of ETCS-L3 of Fig. 2.3. The figure clearly explains which of the language elements of the AD can be used. More in the details, the same considerations of the previous paragraph for parameters, asserts and performance indicator constraints. Since there is not an easy way to specify sending of messages in ADs, they are represented by a couple of sending/receiving activities whose sender element is tagged with the triggered signal. On the contrary, ADs fit better to model branches as there is a proper construct. Involved HW parties and/or FCs are represented by swimlanes.

In general, to understand with are the FCs, the HW components, the signals and the internal changes of state of FCs, please refer to D2.2 — in particular, the functional architecture.

2.3.1. A small example

The OPS described in these few lines, highlights the application example reported in [7]. In this document, a small example is reported to show the modelling methodology of ETCS-L3. Starting from this example, let us imagine the case of a non-successful sending of the acknowledgements of the train to the trackside in response to the sending of a Unconditional Emergency Stop (UES) message.

The scenario wants to depict the case of a hazard related to the non application of a braking action to the engine when the sending of any message from the train to the trackside is not accomplished. Of course, this scenario is not feasible in practice: the case here presented just wants to be an example of the specification approach described in this deliverable.

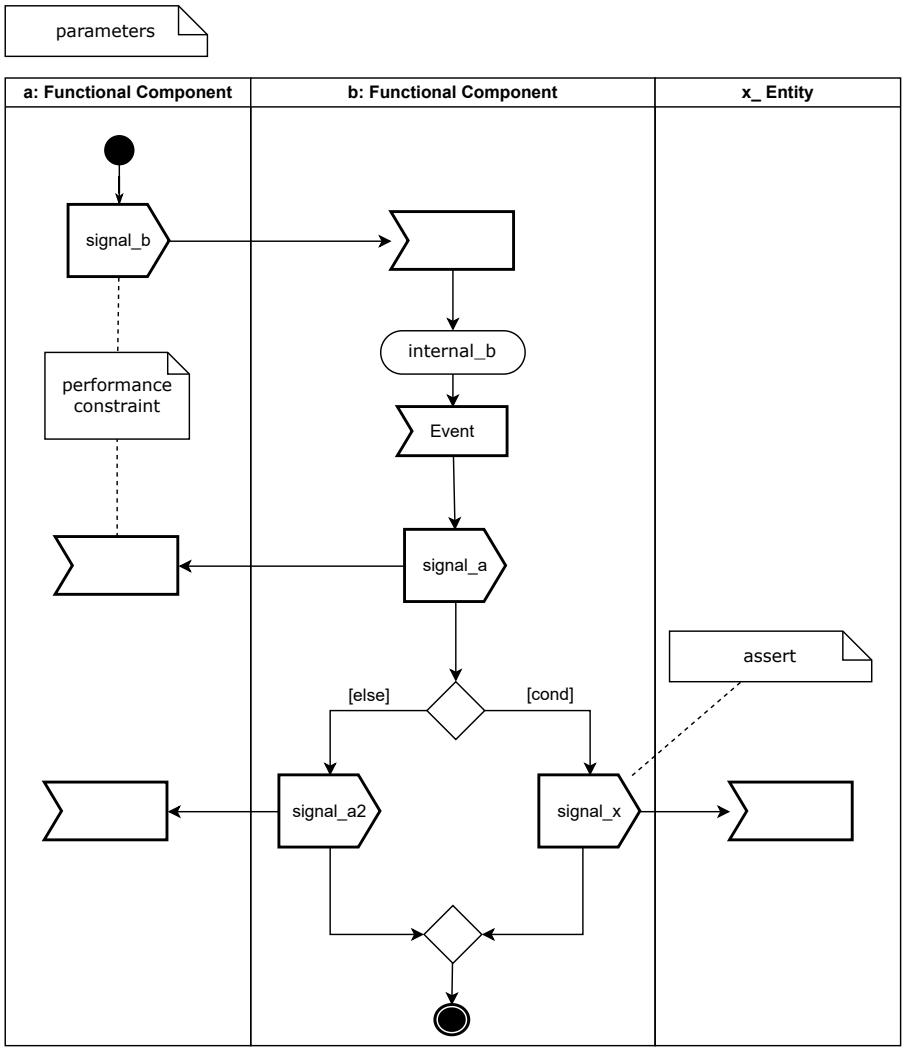


Fig. 2.5. OPS specification by means of SysML AD

Tables 2.1 to 2.10 report a description of the example according to the description template used in [8].

Table 2.1: Running example: General Description

Operational Scenario - small example	
Title:	Communication failure in emergency messages
Abstract:	The scenario wants to verify the presence of a hazard related to the locking of the command to the braking mechanism in case of blocked emergency message acknowledgements.
Description:	The case is of the train controller receiving an emergency message, but the acknowledgement is not sent. If the Braking Supervision function does not schedule a braking command before sending the acknowledgement, the train never starts to brake until the message is not sent.

Table 2.2: Running example: Applicable Use Cases

Applicable Use Case(s)	
1	Braking Supervision

Table 2.3: Running example: Performance Indicators

Performance Indicators				
Name	Type	Property	Threshold/Range	Description
Non Braking Train	Qualitative	Safety	—	In case of unidirectional network partition, the train never brakes.

Table 2.4: Running example: Configuration

Signalling Type	System Type	Track Information
High Speed Line	Any	Linear segment

Table 2.5: Running example: Involved HW Components

HW components
Controller
Brake

Table 2.6: Running example: Involved Functions

Trackside Function(s)	ETCS On-Board Function(s)
—	Braking Supervision

Table 2.7: Running example: Parameters

Parameters				
	Name	Value/Range	Description	Reference
Timer(s)	Timeout	[10-30] secs	Timeout between two attempts of communication from train to track-side	—
Speed	Tspeed	[30-300] km/h	Speed of the train when receiving the UES	—
Communication	Nretry	unbounded	Number of (re-)tries in sending the UES messages	—
	Pfailure	1	Probability the train-to-trackside communication fails	—

Table 2.8: Running example: Behaviour

Behaviour				
Branch	Pre-conditions	Post-conditions	Trigger	Invariants/Assertions/...
	The train is running in Full Supervision (FS) at $Tspeed$ km/h		The trackside sends a UES message to the train.	The network is failing, allowing the delivery of the messages from trackside to the train but not vice versa.
Desc.				
#1	The trackside sends a UES to the train.			
#2	The train sends an acknowledgement before activating the emergency brake.			
#3	The acknowledge message is lost with probability $Pfailure$.			
#4	After $Timeout$ seconds, the train repeats step *#2*.			
#5	After communication success or when the number of trials is greater than $Nretry$, the train activate the emergency brake.			

Table 2.9: Running example: Hazards

Hazards		
ID	Description	Reference
KERNEL-8	Emergency Message Acknowledgement Failure	[9]
TI-1	Service brake / emergency brake not commanded when required	[9]

Table 2.10: Running example: Applicable Requirements

ID	Requirement	Reference
REQ-1	Given a running train, when the Braking Supervision receives an emergency stop message, then it activates the emergency brake.	—
REQ-3	Given the Controller has sent an emergency stop message, when the Braking Supervision function receives such a message, then it sends back to the Controller an acknowledgment message.	—

Once the OPS is described in an extensive way, a compact and clear view can be depicted by a SysML SD: the one representing the chosen example OPS is in Fig. 2.6.

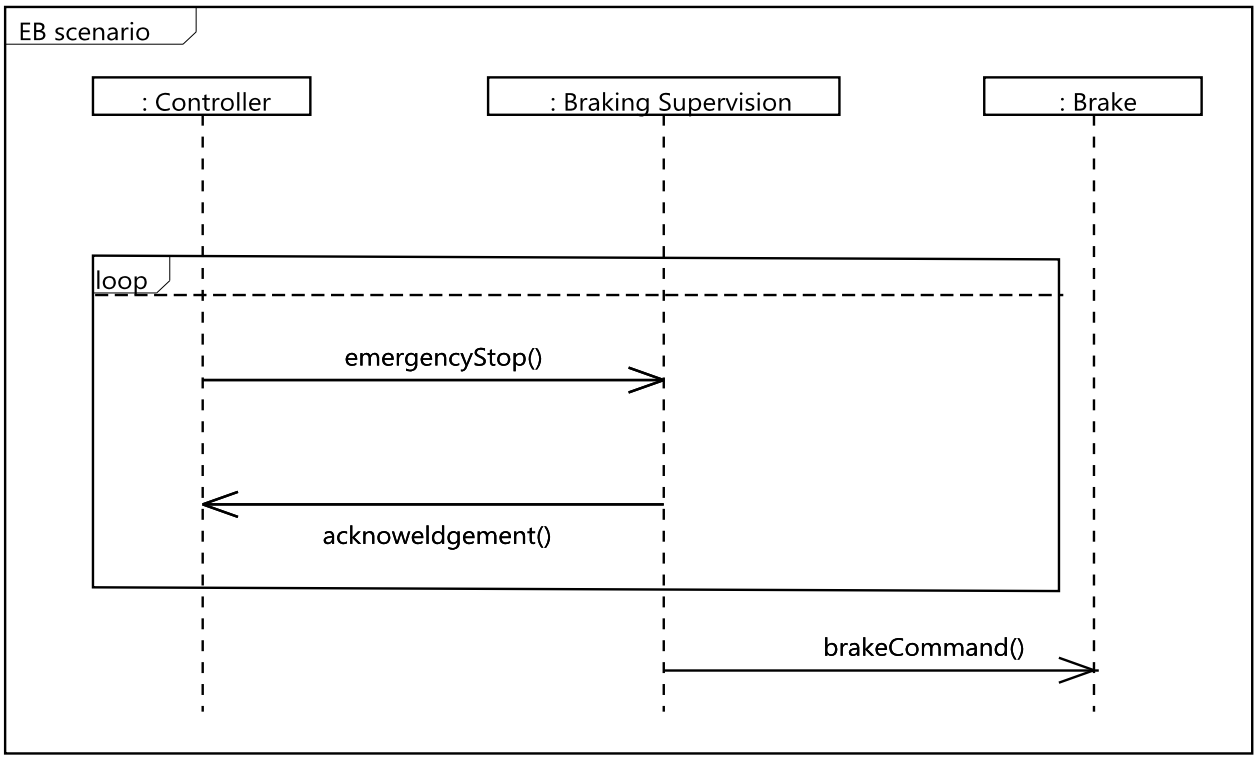


Fig. 2.6. Running example OPS - SysML SD

3. Specifying Operational Scenarios

This chapter is devoted to the description of the OPS considered in this deliverable. There are three different OPS: Loss of Train Integrity, Points Control and Loss/Restore of Communication. The last one has been analysed twice due to the different perspective from which it is specified, modelled and analysed. This deliverable is aligned with the results of D2.1 [6]; the chosen scenarios are the first three in terms of the industrial relevance. For each scenario, a specification is described in a proper section of this chapter, reporting: (1) the tabular description, (2) a proper SysML diagram[1].

3.1. Loss of Train Integrity

3.1.1. OPS short description

Table 3.1: Loss of Train Integrity: General Description

Operational Scenario - Loss of Train Integrity	
Title:	Loss of Train Integrity
Abstract:	In this operational scenario, a connected train moving under the supervision of an ETCS-L3 train loses its integrity.
Description:	The final aim of this operational scenario is to protect the rear end of the train and other trains from collision in the case when a train has lost its integrity. This may occur for different reasons, but if a train splits unintentionally, the Dispatcher needs to take relevant steps to prevent potentially hazardous scenarios. It is worth mentioning here that the lack of Train Integrity information has a significant impact on the performance of the line.

Table 3.2: Loss of Train Integrity: Applicable Use Cases

Applicable Use Case(s)	
1	Loss of Train Integrity
2	Normal Train Movement
3	Staff Responsible (SR) movement
4	On Sight(OS) movement
5	Loss/restore of communication

[1] Some tables and/or diagram may be missing. In this case, the description is very close to the one reported in [6] or the model is very close to some EUC diagrams in [7].

Table 3.3: Loss of Train Integrity: Performance Indicators

Performance Indicators				
Name	Type	Property	Threshold/Range	Description
Loss of integrity duration	Quantitative	Performance		Duration ~~MGH that the train had lost its integrity~~ needed to detect the loss of integrity.
Probability of train integrity loss	Quantitative	Safety		Probability that the train integrity is lost.

Table 3.4: Loss of Train Integrity: Configuration

Signalling Type	System Type	Track Information
General	Full MB (FMB)	

Table 3.5: Loss of Train Integrity: Involved Hw Components

Hw components
Train
Train Integrity Management System (TIMS)
Traffic Management System
Trackside Train Detection

Table 3.6: Loss of Train Integrity: Involved Functions

Trackside Function(s)	ETCS On-Board Function(s)
Trains Management	Integrity Information Management
Communication Management	Train Position Reporting
Track Status Management	Speed and Distance Supervision
Reserved Status Management	Dynamic Speed Profile Management
Route Management	
TTD Management	
MA Management	

3.1.2. Identification of missing formal models

To model the *Loss of Train Integrity (LTI)* OPS, and from previous designed models described in [7], two formal models are missing: the first is related to the movement of the train and the second is related to the function Movement Authority (MA) management. Indeed, if a train loses its integrity, the MA of the following train is updated; thus MA management function is required. In addition, considering the initial condition related to LTI EUC[2] implies the need

[2] "*The train is moving under full supervision mode, and it is situated in the middle of the track*" as described in Table 7.6 of [7]

Table 3.7: Loss of Train Integrity: Hazards

Hazards		
ID	Description	Reference/new possible hazard
[X2R3 D4.2 H-Movements-005]	Undetected movement of a part of the train after loss of integrity, leading to collision.	[X2R3 D4.2]
[X2R3 D4.2 H-Movements-003]	Undetected movement entering the L3 area, leading to collision	PERFORMINGRAIL WP1
[X2R3 D4.2 H-Clearing-003]	Track Status Area erroneously cleared after deactivation of a shunting area, leading to collision.	PERFORMINGRAIL WP1
[X2R3 D4.2 H-TTDfailure-001]	TTD erroneously indicates a Clear Track Status Area, leading to collision or derailment	PERFORMINGRAIL WP1
[X2R3 D4.2 H-Level2-003]	Derailment after loss of train integrity causes adjacent tracks to become occupied, leading to collision.	[X2R3 D4.2]

of a model representing the movement of the train. This model is also required to consider the switching conditions related to the computation of train integrity information (refer to the conditions 3 and 9 in Table 3.10, which enumerates the switching conditions). The newly designed formal models are described in Section 4.3.

Table 3.8: Loss of Train Integrity: Applicable Operational Rules

ID	Operational Rule	Reference
[X2R3 D4.2 OPE-LossTI-2]	When advised of loss of train integrity, the Dispatcher shall, in accordance with non-harmonised rules, protect the area in which a train division may have occurred.	[X2R3 D4.2]
[X2R3 D4.2 OPE-LossTI-2]	When advised of loss of train integrity through an in-cab indication, the Driver shall follow non-harmonised rules.	[X2R3 D4.2]
[X2R3 D4.2 OPE-LevelTrans-2]	When TIMS is not working or the train is not reporting train integrity confirmed, and the Level 3 trackside is engineered not to authorise such trains to enter, the Dispatcher shall apply non-harmonised rules whether to authorise a train to enter a Level 3 Only area.	[X2R3 D4.2]
[X2R3 D4.2 OPE-Generic-2]	The Driver shall only confirm train integrity in accordance with non-harmonised Operational Rules.	[X2R3 D4.2]
[X2R3 D4.2 OPE-StartTrain-2]	Non-harmonised Operational Rules shall define under which circumstances the Driver is allowed to move a train which can not report integrity confirmed.	[X2R3 D4.2]

Table 3.9: Loss of Train Integrity: Applicable Requirements

ID	Requirement	Reference
[X2R3 D4.2 REQ-LossTI-1]	When receiving a position report from a train with the information 'Train integrity lost', the L3 Trackside shall change the Track Status Area associated with this train to Unknown.	[X2R3 D4.2]
[X2R3 D4.2 REQ-LossTI-2]	When the L3 Trackside considers that the integrity is lost for a train, the L3 Trackside shall change the Track Status Area associated with this train to Unknown.	[X2R3 D4.2]
[X2R3 D4.2 REQ-LossTI-3]	When the L3 Trackside considers that the Train Integrity is lost for a train, the L3 Trackside shall react as configured.	[X2R3 D4.2]
[X2R3 D4.2 REQ-LossTI-4]	The L3 Trackside shall consider the Train Integrity as lost when 'No train integrity information' is reported longer than a configurable time (Integrity Wait Timer).	[X2R3 D4.2]
[X2R3 D4.2 REQ-LossTI-5]	When receiving a message from a train with the information 'Train integrity confirmed by external device', the L3 Trackside shall start/restart the Integrity Wait Timer.	[X2R3 D4.2]
[X2R3 D4.2 REQ-LossTI-6]	When receiving a message from a train with the information 'Train integrity confirmed by Driver', if the L3 Trackside is configured to accept confirmation by Driver and the Integrity Wait Timer is running, then the L3 Trackside shall stop the Integrity Wait Timer.	[X2R3 D4.2]
[X2R3 D4.2 REQ-LossTI-7]	After a loss of integrity, the driver shall be made aware of the situation via an indication in the cab.	[X2R3 D4.2]
[X2R3 D4.2 REQ-LossTI-8]	The L3 Trackside shall be able to be configured whether to accept Train Integrity confirmation by the driver.	[X2R3 D4.2]
[X2R3 D4.2 REQ-LossTI-9]	The L3 Trackside shall be configurable as to whether it authorises a Movement Authority for a train that has lost Integrity.	[X2R3 D4.2]
[X2R3 D4.2 REQ-LossTI-10]	If the L3 Trackside receives Validated Train Data for a train with a train length different from previously reported within the same communication session, then the L3 Trackside shall consider the train as having lost Integrity.	[X2R3 D4.2]

Table 3.10: Switching Conditions

SWITCH_ID	Content of the conditions
[1]	No valid Train data is available
[2]	(Train is at standstill) AND (valid Train Data is available and has been acknowledged by the RBC) AND (the train integrity is confirmed by the driver)
[3]	(The information “Train integrity confirmed” is received from an external device) AND (valid Train Data is available and has been acknowledged by the RBC) AND (Train Data regarding train length has not changed since the time the train was last known to be integer) AND (the train position is valid and is referred to an LRBG) AND (the train position was valid and was referred to an LRBG at the time the train was last known to be integer) AND (no reverse movement is currently performed nor has been performed since the time the train was last known to be integer) AND (the distance between the min safe rear end at the time the train was last known to be integer and the current estimated train position does not exceed the range of the safe train length information)
[4]	(The information ”Train integrity lost” is received from an external device) AND (valid Train Data is available since the time the train integrity was last known to be lost)
[5]	A position report indicating that the train integrity is confirmed is sent to the RBC
[6]	The information ”Train integrity status unknown” is received from an external device
[7]	Train Data regarding train length is changed
[8]	A reverse movement is performed
[9]	The distance between the min safe rear end at the time the train was last known to be integer and the current estimated train position exceeds the range of the safe train length information

3.2. Points Control

3.2.1. OPS short description

Table 3.11: Points Control: General Description

Operational Scenario - Points Control	
Title:	Points Control
Abstract:	Points control operational concerns the moving, locking and releasing of points related to two subsequent trains requesting to pass over different points.
Description:	In this scenario, the situation is considered in which two trains running under normal moving block conditions cross a point successively, with the second train requiring the point to move to a different position. Movement of this point is not allowed as long as the first train occupies the associated track area. Also, the point cannot be moved when the point is already reserved for the second train.These point movement timing restrictions apply due to the hazard of moving a point while a train is passing, or about to pass, over it, possibly leading to derailment of the train.

Table 3.12: Points Control: Applicable Use Cases

Applicable Use Case(s)	
1	Points control
2	Normal train movement
3	Sweeping

Table 3.13: Points Control: Performance Indicators

Performance Indicators				
Name	Type	Property	Threshold/Range	Description
Headway time	Quantitative	Performance		Time between train heads over points

Table 3.14: Points Control: Configuration

Signalling Type	System Type	Track Information
General	Full MB (FMB)	

3.2.2. Identification of missing formal models

For the modelling the *Points Control* OPS, some channels miss from the models designed in [7]; such channels are needed to communicate with other automata. The missing details in this automaton are “MainAutomaton”, “CallRoute” and “SetandLock” and have made significant changes to this automaton as compared to the models present in [7]. These updated and new models are described in Section 4.4.

Table 3.15: Points Control: Involved Hw Components

Hw components
Train
Train Integrity Management System (TIMS)
Traffic Management System
Trackside Train Detection

Table 3.16: Points Control: Involved Functions

Trackside Function(s)	ETCS On-Board Function(s)
Points Management	Train Position Reporting
Track Status Management	Integrity Information Management
Reserved Status Management	Speed and Distance Supervision
Trains Management	
Route Management	
MA Management	

Table 3.17: Points Control: Parameters

Parameters				
	Name	Value/Range	Description	Reference
Timer(s)	Track section timer	To be defined		
Train	L_TRAIN	To be defined	Length of the train	
	M_MODE	0: FS 1: OS 2: SR 3: SH 6: SB 12: LS	On-board operating mode	
Speed	V_TRAIN	To be defined	Train speed	
Track	Track configuration: location of points	To be defined		
	Point control processing time (Interlocking)	To be defined		

Table 3.18: Points Control: Hazards

Hazards		
ID	Description	Reference/new possible hazard
H-Points-001	A point is moved in an Unknown / Occupied / Reserved Track Status Area with a train over it, or when it is about to pass over it, leading to derailments.	[X2R3 D4.2]

Table 3.19: Points Control: Applicable Operational Rules

ID	Operational Rule	Reference
OPE-Generic-1	Where the system permits, the Dispatcher shall, in accordance with non-harmonised rules, remove an area with track status Unknown.	[X2R3 D4.2] [MR D1.1]
OPE-Generic-6	The Dispatcher shall, in accordance with non-harmonised rules, create or extend an Unknown Area flagged as “Sweepable” or “Non-sweepable”.	[MR D1.1]
OPE-Generic-7	The Dispatcher shall, in accordance with non-harmonised rules, be able to move a set of points partially or completely located in an Occupied or Unknown Area.	[X2R3 D4.2]
OPE-OS-1	When sweeping an area in ETCS Level 3 On Sight mode, the Driver shall, in accordance with non-harmonised rules, follow operational procedures.	[MR D1.1]
OPE-OS-2	When asked to confirm the line is Clear, the Driver shall, in accordance with non-harmonised rules, observe the track and confirm the status of sections of track joining/diverging from the line over which the train is passing.	[MR D1.1]
OPE-OS-3	When advised by the Driver that a section of line has been examined and observed clear, the operator shall, in accordance with non-harmonised rules, clear the status of sections of track joining/diverging from the line over which the train passed where the system allows.	[MR D1.1]
OPE-OS-4	The operator shall, in accordance with non-harmonised rules, advise the Driver of any specific checks before authorising a move in ETCS Level 3 On Sight mode.	[MR D1.1]

Table 3.20: Points Control: Applicable Requirements

ID	Requirement	Reference
REQ-PTS-1	The L3 Trackside shall prevent movement of points within an Unknown or Occupied Track Status Area, or within a Reserved Status Area, unless using an operational procedure.	[X2R3 D4.2]
REQ-PTS-2	The L3 Trackside shall be configured with Release Points to enable Points to be moved when the area of track containing the Points has Consolidated Track Status Clear and does not contain any part of a Reserved Status Area.	[X2R3 D4.2]
REQ-PTS-3	On request from the TMS, the L3 Trackside shall be able to move points for which all or parts of it is in an area with Track Status Unknown or Occupied, or both.	[X2R3 D4.2]
REQ-PTS-4	When a train is sweeping a set of points, the L3 Trackside shall remove or reduce a Sweepable Unknown Track Status Area from the alternate leg of the points as far as the Fouling Point, in addition to the path that the train takes.	[X2R3 D4.2]

3.3. Loss/Restore of Communications - Scenario (a)

3.3.1. OPS short description

Table 3.21: Loss/Restore of Communication: General Description

Operational Scenario	
Title:	Loss/Restore of Communication — case A
Abstract:	In this Operational Scenario, a connected train moving under the supervision of ETCS L3 train may lose communication with the Trackside and may restore it before the session expires.
Description:	The aim of this operational scenario is to monitor the communication session with a train and in case of loss to guarantee appropriate management of the reconnection. In case communication with the train is lost, three possible cases can occur: A) connection is re-established before session timeout; B) connection is re-established before session timeout, with changes in train id/length; C) the train fails to reconnect before session timeout.

Table 3.22: Loss/Restore of Communication: Applicable Use Cases

Applicable Use Case(s)	
1	Loss/Restore of Communication
2	Normal Train Movement
3	Staff Responsible (SR) movement
4	Sweeping
5	Loss of Train Integrity

Table 3.23: Loss/Restore of Communication: Configuration

Signalling Type	System Type	Track Information
General	Full MB (FMB)	

3.3.2. Identification of missing formal models

In order to model the operation scenario Loss & Restore of Communication (a) (OS LRC), a formal model is missing among the ones described in a previous deliverable [7]. This model is a stub for the Train Management System that needs to synchronise with the Train Manager upon reception of a new position of the train.

Table 3.24: Loss/Restore of Communication: Involved Hw Components

Hw components
Train
Train Integrity Management System (TIMS)
Traffic Management System
Trackside Train Detection

Table 3.25: Loss/Restore of Communication: Involved Functions

Trackside Function(s)	ETCS On-Board Function(s)
Communication Manager	TPR Manager
Trains Manager	Speed Distance Supervisor
Track Status Manager	Integrity Information Manager
Route Manager	

Table 3.26: Loss/Restore of Communication: Parameters

Parameters				
	Name	Value/Range	Description	Reference
Timer(s)	MUTE_TIMER	2	Defines a timeout for a given train with which L3 Trackside has an active communication session. When this timer expires, the communication with this train is considered lost.	[X2R3 D4.2] [10]
	SESSION _EXPIRED _TIMER	10	Timer of the train after which a communication session is considered terminated.	[X2R3 D4.2] [10]
Train	NID_ENGINE	To be specified	Identifies a train. Used to identify a train when restoring communications with a new session after entering the Radio Hole area.	[X2R3 D4.2] [10]
	L_TRAIN	To be specified	Length of the train.	[X2R3 D4.2] [10]

Table 3.27: Loss/Restore of Communication: Hazards

Hazards		
ID	Description	Reference/new possible hazard
H-Clearing-001c	Track Status Area erroneously cleared during L3 Trackside initialisation by dispatcher leading to a collision. The hazard applicable to this use case is mainly related to the incorrect clearing of tracks after the recovery of communication (after the Mute Timer timeout).	[X2R3 D4.2] [10]

Table 3.28: Loss/Restore of Communication: Applicable Operational Rules

ID	Operational Rule	Reference
[X2R3 D4.2 OPE-LossComms-1]	The Dispatcher shall, in accordance with non-harmonised rules, protect the movement of a non-communicating train. The movement of a non-communicating train must be safe and controlled by the Driver and the Dispatcher working together.	[X2R3 D4.2] [10]

Table 3.29: Loss/Restore of Communication: Applicable Requirements

ID	Requirement	Reference
[X2R3 D4.2 REQ-LossComms-1]	To timely react to the potential loss of communications with ETCS On-board, the L3 Trackside, if configured to do so, shall be able to supervise a defined timeout (a MUTE_TIMER) for each train with which it has an active communication session.	[X2R3 D4.2] [10]
[X2R3 D4.2 REQ-LossComms-2]	L3 Trackside shall reset the MUTE_TIMER for a train upon receiving of a message from said train.	[X2R3 D4.2] [10]
[X2R3 D4.2 REQ-LossComms-3]	The L3 Trackside shall maintain the communication session with ETCS On-board as active even when the MUTE_TIMER has expired until also the maximum time (SESSION_EXPIRED_TIMER) to maintain a communication session has expired.	[X2R3 D4.2] [10]
[X2R3 D4.2 REQ-LossComms-4]	When the Mute timer expires for a train which has not been sent Reversing Area Information, nor entered an announced Radio Hole, then the L3 Trackside shall change the Track Status Area associated with the train to Unknown and extend this Area until the end of the Reserved Status Area for the train.	[X2R3 D4.2] [10]
[X2R3 D4.2 REQ-LossComms-5]	If the Mute timer is not considered for use on a particular application, the L3 Trackside shall react when the session timer expires by setting the Track Status Area associated with the train to Unknown and extend this Area until the end of the Reserved Status Area for that train.	[X2R3 D4.2] [10]
[X2R3 D4.2 REQ-LossComms-6]	When the L3 Trackside considers the communication session with a train is terminated, then the L3 Trackside shall remove any Reserved Status Area associated with that train.	[X2R3 D4.2] [10]
[X2R3 D4.2 REQ-RecoveryMgmt-1]	L3 Trackside shall consider a train which starts communicating with the L3 Trackside within the same communications session as previously used for the train as the same train, so long as no change in train data has occurred. This happens when a train leaves a Radio Hole area before the Radio Hole timer expires.	[X2R3 D4.2] [10]

Table 3.30: Loss/Restore of Communication: Applicable Requirements

ID	Requirement	Reference
[X2R3 D4.2 REQ-RecoveryMgmt-2]	L3 Trackside shall consider a train reconnecting with a new communication session as the same train of a previous communication session if (a) the two trains have the same ID (NID_ENGINE) and (b) the two trains have the same length (L_TRAIN).	[X2R3 D4.2] [10]
[X2R3 D4.2 REQ-RecoveryMgmt-3]	If the L3 Trackside determines that the same train has reconnected and confirmed Integrity, the L3 Trackside shall update the Unknown Track Status Area associated with this train, resulting from the Loss of Communications due to the expiry of the Radio Hole timer to an Occupied Track Status Area with an extent corresponding to the new train location.	[X2R3 D4.2] [10]

3.4. Loss/Restore of Communication - Scenario (b)

3.4.1. OPS short description

This scenario has the aim of investigating the trade-off between performance (e.g., capacity, line throughput) and quality of service (e.g., number of acceleration, number of brake activations) when the communication between the onboard unit and the trackside is lost and then restored. The scenario is developed to perform a different kind of analysis, with other objectives, with respect to the Loss/Restore of Communication scenario introduced in the previous subsection. The simulation approach taken for this scenario is the same initially planned for the Virtual Coupling scenario, deleted according to the JU indication.

Table 3.31: Loss/Restore of Communication -b- : General Description

Operational Scenario	
Title:	Loss/Restore of Communication -b-
Abstract:	In this Operational Scenario, one train belonging to a fleet moving under the supervision of the ETCS-L3 trackside loses communication with the trackside at a specific instant of time and restores it before the session expires. Different from the previous scenario, the connection is re-established before the session timeout.
Description:	The aim of this operational scenario is to study the effects of a loss and a subsequent restoration of the communication on both the railway network throughput, also taking as parameters train mechanical features due to different market segments. In addition, it would be possible to investigate the effects on the passengers' comfort due to continuous accelerations and brakes.

Table 3.32: Loss/Restore of Communication -b-: Applicable Use Cases

Applicable Use Case(s)	
1	Loss/Restore of Communication
2	Normal Train Movement
3	Loss of Train Integrity

Table 3.33: Loss/Restore of Communication -b-: Performance Indicators

Performance Indicators				
Name	Type	Property	Threshold/Range	
$\#\_Trains$	Quantitative	Capacity	[0-num_vehicles]	number of trains exiting the line in the given time interval
$\#\_brakings$	Quantitative	Quality of service	>0	number of activation of a braking in the given time interval

Table 3.34: Loss/Restore of Communication -b-: Configuration

Signalling Type	System Type	Track Information
General	Full MB (FMB)	Straight line

Table 3.35: Loss/Restore of Communication -b-: Involved Hw Components

Hw components
Communication Network
Track
Trackside
Traffic Management System
Train
Train Integrity Management System (TIMS)

Table 3.36: Loss/Restore of Communication -b-: Involved Functions

Trackside Function(s)	ETCS On-Board Function(s)
Communication Management	Train Position Reporting
Trains Management	Speed and Distance Supervision
Track Status Management	Integrity Information Management
MA Management	Dynamic Speed Profile Management

3.4.2. Identification of missing formal models

The model described in [7] is a general performability model whose extension is required to fit this OPS; this model also provides proper inputs to WP4 - Integrated Moving Block architecture for safe and optimised traffic operations (WP4). Regarding that model, the following modifications have been made:

- Braking curves and related braking events have been introduced. The current model considers the following braking events: emergency, indication, service and warning.
- The driver has been explicitly modelled to introduce his/her reaction time. The driver is the actor in charge of activating the indication, service and warning braking. The emergency braking is up to the onboard system and stops the train with the maximum deceleration.
- The function of extending Track Status Areas performed by the trackside has been modelled in a more accurate way, defining the maximum possible extent.
- A specific function has been added to simulate the loss of communication of a single train for a determined time interval and the subsequent restoration.

It is worth noting that this scenario requires a number of data to be simulated, which allows us to set the scenario for different system configurations and perform sensitivity analyses. These data are partially reported in deliverable [7], and further data have been considered due to the above-mentioned modifications. They are summarized in Section 4.6.

Finally, here the parameters NID_ENGINE, L_TRAIN and P_COMM are not considered be-

Table 3.37: Loss/Restore of Communication -b-: Parameters

Parameters				
	Name	Value/Range	Description	Reference
Timer	MUTE_TIMER	2 seconds	Defines a timeout for a given train with which L3 Trackside has an active communication session. When this timer expires, the communication with this train is considered lost.	[X2R3 D4.2]
	SESSION _EXPIRED _TIMER Timer	10 seconds	Timer of the train after which a communication session is considered terminated.	[X2R3 D4.2]
Line	Headway (Scheduling)	According to the market segment	Temporal distance between trains entering the line.	N.A.

Table 3.38: Loss/Restore of Communication -b-: Variants

Variant	Description	Alternatives	Main case	Impact/Affected Steps
Market segment	Passenger-related market segments.	high-speed, mainline, regional, urban and freight railways	high-speed	The market segments are characterized by different values of some parameters, there is no impact of the steps of the scenario

Table 3.39: Loss/Restore of Communication -2-: Applicable Operational Rules

ID	Operational Rule	Reference
[X2R3 D4.2 OPE-LossComms-1]	The Dispatcher shall, in accordance with non-harmonised rules, protect the movement of a non-communicating train. The movement of a non-communicating train must be safe and controlled by the Driver and the Dispatcher working together.	[X2R3 D4.2]

cause the first two are model data (the fleet has more than one train; this number may vary in different simulations) whereas P_COMM is related to the Radio Hole Use Case that is not considered by the developed scenario.

4. Formal Modelling of Operational Scenarios

4.1. Overview on Moving Block Formal Models

This section reports on the formal models of the MB system developed in the context of WP2. Regarding what described in [7], more work is needed by the modelling needs of the considered OPSs: new formal models have been developed, or existing models are updated to better capture specific behaviours. The need for new models and/or updates is reported in Chapter 3 for each scenario.

The work described in this section is the outcome of a joint work performed by the working group. In this work, a long internal analysis of the scenarios and of the existing formal models is performed, with the objective of model integration and with the side effect of a joint cross-review of the models developed in isolation.

Specifically, UPPAAL models are possibly integrated according to the composition approach described in the next section. Of course, it is not possible to integrate UPPAAL and SAN models[1]. The SAN model consists of more subcomponents that have been already identified and integrated in the first version described in D2.2 [7]. Hence, in the following, the updates of these models are described.

Composition of formal models is never a trivial task, in particular two main aspects deserve great attention: properties and complexity. The integration produces a new model whose properties must be checked carefully, and whose complexity could make the analysis unfeasible.

In addition, the proposed composition approach includes the development of so-called *stub models*. As in software development, a stub is used to stand in for some not modelled functionality: a stub simply provides the implemented models with their needed inputs or receive/consume their outputs if necessary. Nonetheless, stubs are models too, even simple, they must be composed with the fully developed models as well.

Table 4.1 summarizes the developed Moving Block models available for composition and discussed in the rest of the chapter. In the column ”type”, the word ”Full” means that the model is not a stub, but still, the modelled behaviour is at the (high) level of abstraction allowed by this modelling activity.

[1] For brevity, details about UPPAAL TA and SANs and their solver Möbius are reported in D2.1 [6].

Table 4.1: PERFORMINGRAIL Formal Model Library (cont.)

Model	Form. Lang.	Tool	Type	MB Component
TIMS	TA	UPPAAL	Full	TIMS
Driver	TA	UPPAAL	Stub	Driver
Localization Unit	TA	UPPAAL	Full	Train
Data management	TA	UPPAAL	Stub	Train
TimeStep	TA	UPPAAL	Full	System
Train Movement	TA	UPPAAL	Full	Train
Train Movement Main	TA	UPPAAL	Full	Train
On-Board Init	TA	UPPAAL	Stub	Train On-Board
IIM Status Management	TA	UPPAAL	Full	Train On-Board (Int. Inf. Management)
IIM Updating	TA	UPPAAL	Full	Train On-Board (Int. Inf. Management)
TPR Management	TA	UPPAAL	Full	Train On-Board (Train Position Management)
Speed Distance Supervision	TA	UPPAAL	Full	Train On-Board (Speed Distance Supervision)
MA Management	TA	UPPAAL	Full	Trackside (MA Management)
Points Control (main)	TA	UPPAAL	Full	Trackside (Points Management)
Call Route	TA	UPPAAL	Full	Trackside (Points Management)
SetandLock	TA	UPPAAL	Full	Trackside (Points Management)
Report Point Status	TA	UPPAAL	Stub	Trackside (Points Management)
Signals	TA	UPPAAL	Full	Trackside (Points Management)
Left-Right Point Position	TA	UPPAAL	Full	Trackside (Points Management)
Track Status Clear	TA	UPPAAL	Full	Trackside (Points Management)

Table 4.2: PERFORMINGRAIL Formal Model Library (end)

Model	Form. Lang.	Tool	Type	MB Component
Communication Manager	TA	UPPAAL	Full	Trackside (Communication Management)
Train Manager (Main)	TA	UPPAAL	Full	Trackside (Communication Management)
Train Manager (ReqTPR)	TA	UPPAAL	Full	Trackside (Communication Management)
Train Manager (AckVTD)	TA	UPPAAL	Full	Trackside (Communication Management)
Train Manager (IntChk)	TA	UPPAAL	Full	Trackside (Trains Management)
Track Status Manager	TA	UPPAAL	Full	Trackside (Track Status Management)
Route Manager	TA	UPPAAL	Stub	Trackside (Route Management)
Traffic Management System	TA	UPPAAL	Stub	Trackside (Traffic Management)
Trackside	SAN	Möbius	Full	Trackside
On-Board and Comm. Net.	SAN	Möbius	Full	Train On-Board, Comm. Network

4.2. Composition approach and artefacts

In this section, the Moving Block formal models composition approach is detailed, and next, the models are specialised by using the syntactic and semantic constructs of the UPPAAL timed automata formalism.

The composition approach follows the following steps:

- **Step 1**. Based on the functional architecture of the European Train Control System (ETCS)-L3 system, identify dependency relations between blocks, i.e., blocks that communicate with each other directly, via signals;
- **Step 2**. Create "sub-systems for verification", by grouping the blocks identified above, and considering the defined OS;
- **Step 3**. For each sub-system, replace the constituent architectural blocks with their respective formal-model-based behavioural descriptions;
- **Step 4**. Define a common set of variables and data structures used by all formal models, across the entire functional architecture;
- **Step 5**. Identify the composition operator and apply it onto the formal models involved in a sub-system, to create its corresponding composed formal model;
- **Step 6**. Identify the communication mechanism between the corresponding formal models within a sub-system, e.g., shared variables, message passing etc.
- **Step 7**. Specify formally the sub-systems' properties for verification.

Since the formal models consider for integration are all defined as UPPAAL Timed Automata (UTA), UTA mechanisms are used to instantiate Steps 5 and 6.

To model concurrent timed systems, several UTAs can be composed in an automata system. UPPAAL uses the CCS parallel composition operator [11] to build a system or a *network* of UTA. CCS allows individual components to carry out internal actions (i.e., interleaving), as well as pairs of components to perform hand-shake synchronization. For a network of UTA, particular automata in the network synchronize using *channels* and values can be passed between them using *shared (global) variables*.

A state of a network of UTA is defined by the locations of all automata in the network and the values of clocks and discrete variables. Let S be a set of channel names, then the set of synchronization actions of the network of UTA is defined as $\Sigma = \{a? \,|\, a \in S\} \cup \{a! \,|\, a \in S\}$. An edge with a synchronization action $a? \in \Sigma$ is only enabled if another automaton in the network simultaneously can perform a complementary action $a! \in \Sigma$. The symbol *name(a)* denotes the channel name of a, defined by $name(a?) = name(a!) = a$. Binary channels are used to synchronize one sender with a single receiver, and broadcast channels are used to synchronize one sender with an arbitrary number of receivers.

The above-mentioned UTA parallel composition operator is used, per each identified sub-system, with respect to the operational scenarios defined in section 3, to create the respective formal model of each sub-system, on which verification can be carried out. To give an example, formal models of functional blocks *Route Manager*, *Track Status Manager*, *Reserved Status Manager*, and *Points Manager* are composed in parallel, and form the network of UTA on which verification of the requirements related to the *Points Control* OPS can be performed. Similarly, a network of UTA, corresponding to all relevant sub-systems of the functional architecture, is defined.

Step 4 of the above methodology becomes concrete as a defined *Shared Declarations.xml* file, which is stored in the repository referenced in this deliverable. The file contains the definitions of all global variables, including shared channels, of the moving block formal models enumerated in Section 4.1, and represents the integration-communication mechanism of each sub-system.

4.3. Loss of Train Integrity

In this section, both the outcomes of internal and cross review, and the new designed models are discuessed.

4.3.1. Internal and cross Review

The internal and cross review allowed to:

- fix some missing updates of variables, for example, the local variable $LOC\_CurrentIntegrityStatus$ which contains the current integrity status of the train.
- add some improvements in the formal models in order to provide a closer match on state names between the UPPAAL timed automata and the SysML state machines.

4.3.2. New designed and updated models

The structure of the formal models representing the Operation Scenario Loss of Train Integrity is composed of fourteen automata. The behaviour of each automaton is described as below

- "Ext_TIMS" automaton which emulates the behaviour of the *TIMS* block by sending three signals: $'TIIreceived\_Unknown'$, $'TIIreceived\_Confirmed'$ and $'TIIreceived\_Lost'$. For this model, only the names of channels are updated from the previous version. This automaton is represented in Fig. 4.1.

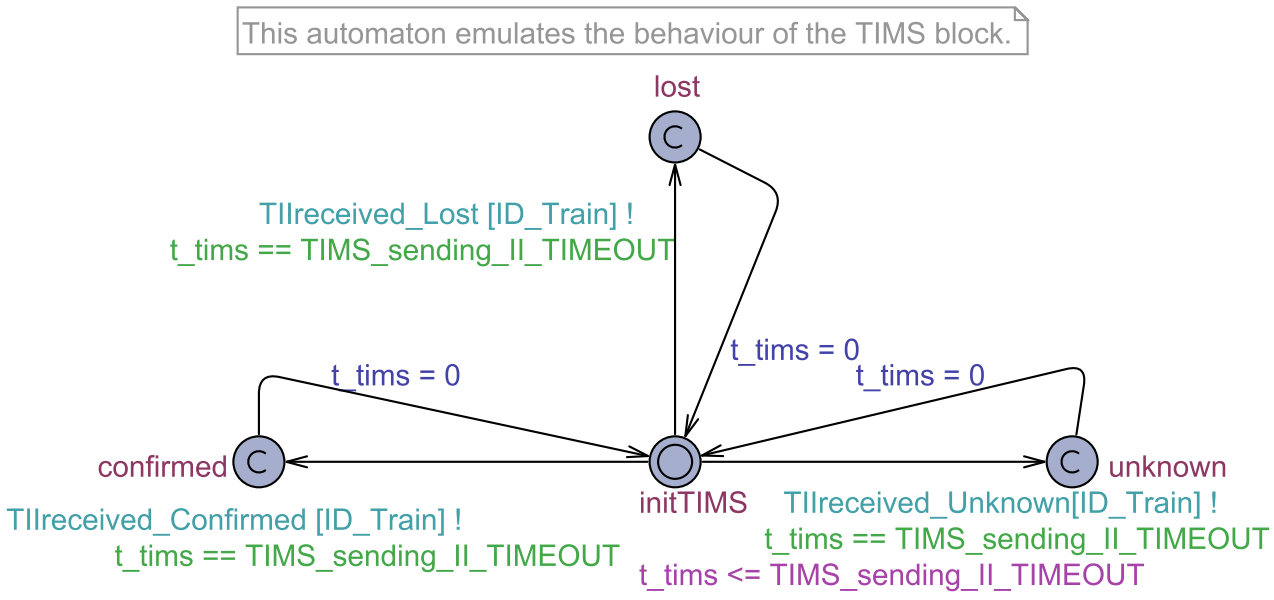


Fig. 4.1. "Ext_TIMS" automaton

- "Ext_Driver" automaton which emulates the behaviour of the *Driver* block by sending signal $'integrityDriver"$. For this model, only the name of the channel is updated from

the previous version. This automaton is represented in Fig. 4.2.

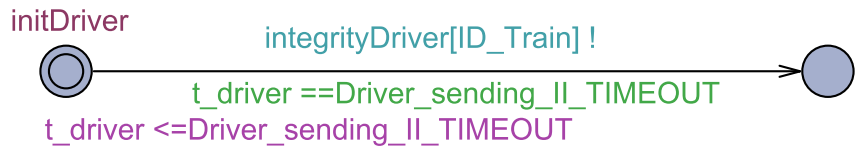


Fig. 4.2. "Ext_Driver" automaton

- "Ext_TrainLocalizationUnit" automaton, which emulates the behaviour of the external actor *Train Localization Unit*. It regularly receives from the automaton "Ext_TrainMovement_MAIN" the location and speed of the train ($'LOC\_sendLocationSpeed'$). When, it receives from the automaton "OB_TPRManagement" requests for location $'positionRequest'$, it records the last received train information related to speed and location and sends it back to "OB_TPRManagement" automaton ($'positionReceived'$). This automaton is updated from the previous version, and it represented in Fig. 4.3.

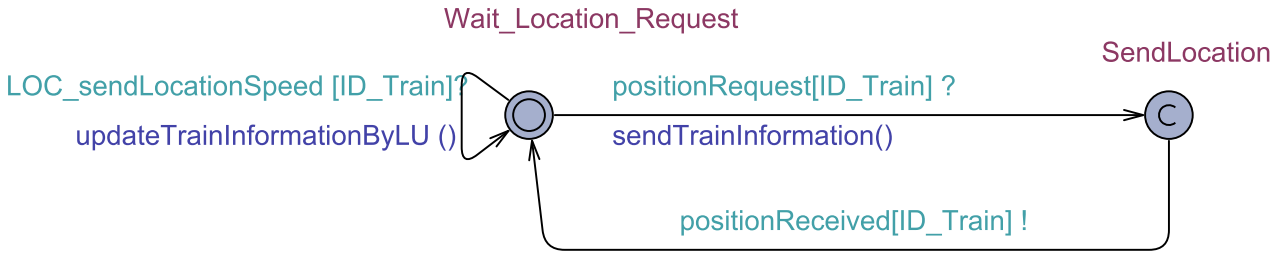


Fig. 4.3. "Ext_TrainLocalizationUnit" automaton

- "Ext_TrainDataManagement" automaton represents a part of the behaviour of train which is related to train data management. Train data is required to compute integrity status (please refer to conditions 1, 2, 3, 4, and 7 in the Table 3.10). "Ext_TrainDataManagement" automaton receives a request ($'LOC\_RequestTrainData'$) from "OB_SpeedDistanceSupervision" automaton to send train data. It fills train data structure and sends it back ($'trainData'$). This automaton is updated from the previous version where Train data was re-sent each timeout. This does not depict the real behaviour. To have a profound understanding of train data management, one can refer to [7] (figure 6.6 page 51), to [12] (part 7 page 41, part 3 page 178 and part 5 page 19). "Ext_TrainDataManagement" automaton is represented in Fig. 4.4.
- "Ext_TimeStep" automaton is a simple model to manage the incrementation of the integer variables. If the clock LOC_C_FREQ is equal to time step (LOC_FREQ), the broadcast channel $'LOC\_NextTimeStep'$ is issued. This channel ensures the synchronization with "Ext_TrainMovement" and "Ext_TrainMovement_MAIN" automata which update the integer variables according to the defined execution frequency. This automaton is a new model. It is inspired from [13] and it is represented in Fig. 4.5.
- "Ext_TrainMovement_MAIN" automaton represents a part of the behaviour of train movement. In this automaton, the function initialise() initializes the value of all train

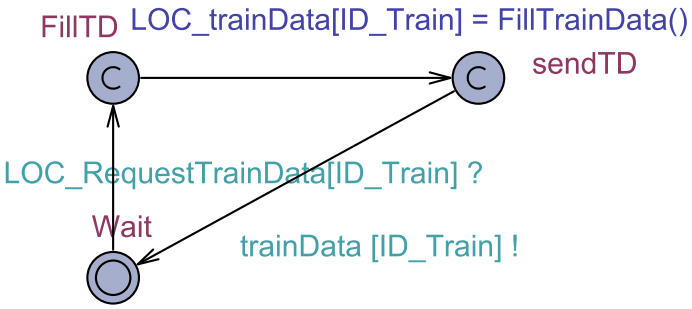


Fig. 4.4. “Ext_TrainDataManagement” automaton

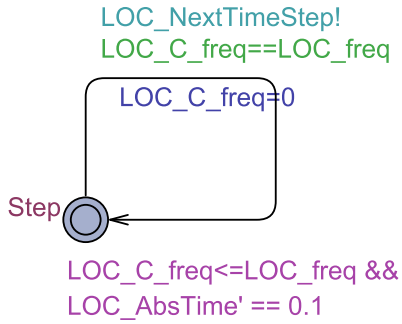


Fig. 4.5. ‘Ext_TimeStep automaton

movement variables: current speed ($V\_int[id\_T]$), current position ($P\_int[id\_T]$, current acceleration ($A\_int[id\_T]$), destination ($destination[id\_T]$), the initial reaction of the train movement through the boolean variables $Brake[id\_T]$ and $Accelerate[id\_T]$. Boolean variables are also required to switch from one state to another in “Ext_TrainMovement” automaton. The function updateTrainInfo() prepares train information to be sent to “Ext_TrainLocalizationUnit” automaton using the channel $'LOC\_sendLocationSpeed'$. Train information includes the value of current speed ($V\_int[id\_T]$), the value of current position ($P\_int[id\_T]$) and the identifier of last referenced balise group. For the first position report, train information, is sent, just after their initialization (function initialise()) in order to send the start location and the initial speed of the train. Then, they are sent each $freqLU$ (frequency of sending train information). The function computeAcceleration_int() computes the value of train acceleration represented by the variable $A\_int[id\_T]$ and the reaction of the train movement in the next time step represented by the value of the boolean variables $Brake[id\_T]$ and $Accelerate[id\_T]$). The value of variables ($A\_int[id\_T]$, $Brake[id\_T]$ and $Accelerate[id\_T]$) depend on the value of movement authority ma_int, the value of current speed ($V\_int[id\_T]$), the value of current position ($P\_int[id\_T]$) and the distance to break. “Ext_TrainMovement_MAIN” automaton receives the channel $'MAupdate'$ and updates the value of movement authority ma_int and the train mode $mode[id\_T]$ accordingly. This automaton is a new model. It is inspired from [14] and it is represented in Fig. 4.6.

- “Ext_TrainMovement” automaton represents a part of the behaviour of train movement. In this automaton, functions $ComputeA()$, $ComputeV()$ and $ComputeP()$ allow updating the train acceleration, velocity and position values. The initial state is *'Wait_First_MA'* state. The train waits for its first MA. In the state *'Start_Move'*, depending on the value

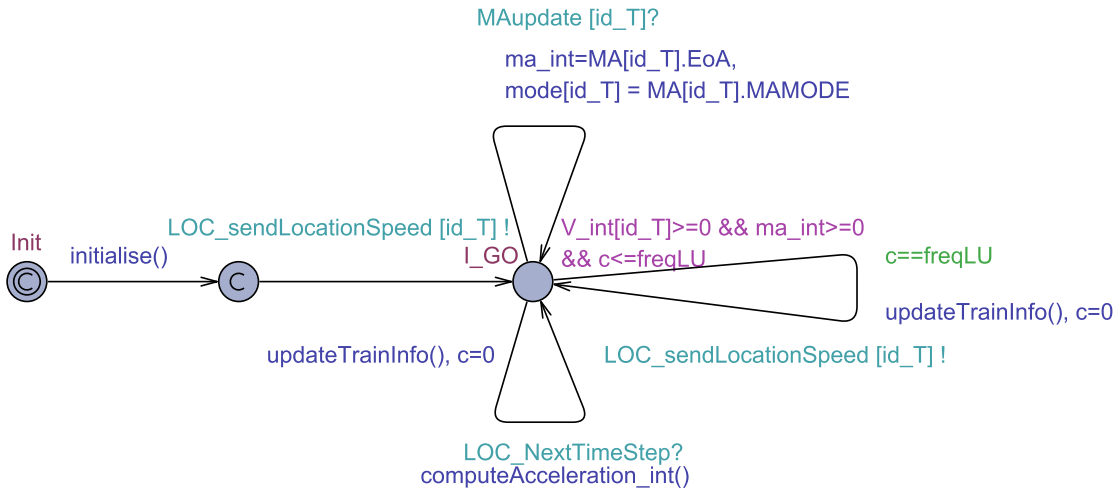


Fig. 4.6. "Ext_TrainMovement_MAIN" automaton

of train acceleration ($A\_int[id]$), the train will accelerate if the acceleration value is positive ($A\_int[id] > 0$) and in this case, the next state is *'AcceleratingTrain'*. If the acceleration value is negative ($A\_int[id] < 0$), the train will brake and the next state is *'BrakingTrain'*. Finally, if the acceleration value is zero ($A\_int[id] == 0$), the train will run at a constant speed and the next state is *'ConstantSpeed'*. The switching from one state to another is then performed each time step ('$LOC\_NextTimeStep$') following the values of boolean variables $Brake[id_T]$ and $Accelerate[id_T]$. This automaton is a new model. It is inspired from [13] and it is represented in Fig. 4.7.

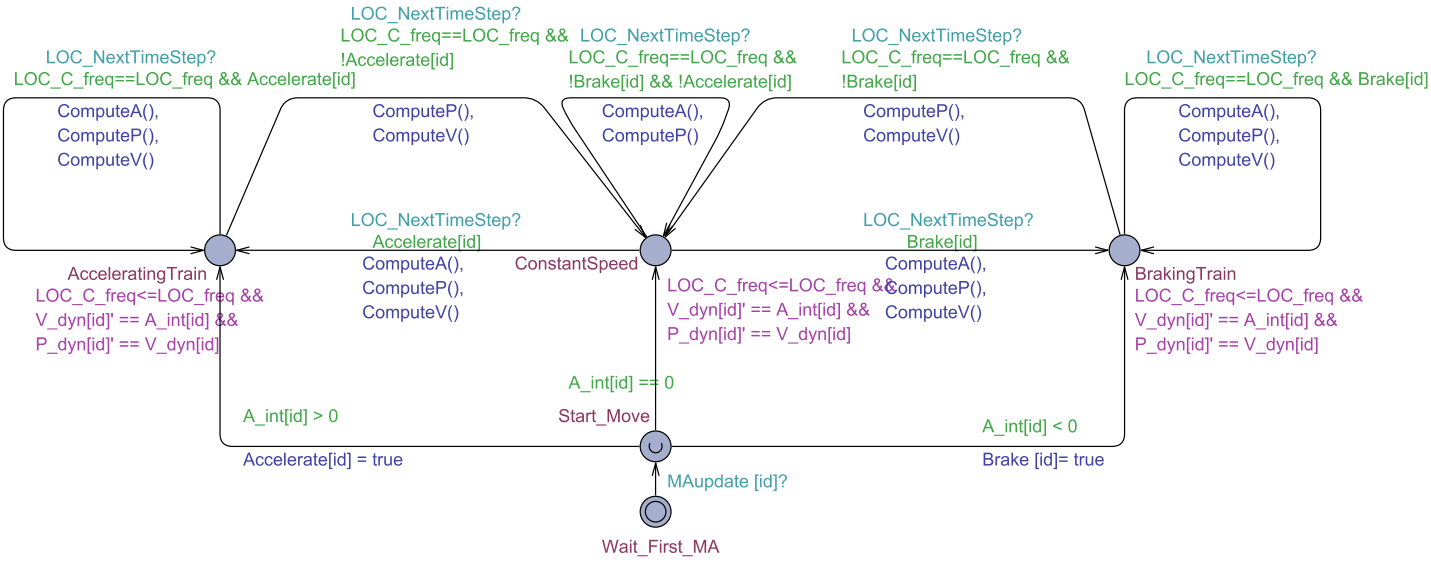


Fig. 4.7. "Ext_TrainMovement" automaton

- "OB_IIMStatusManagement" automaton represents a part of the behaviour of the on-board function "Integrity_Information_Management". It intercepts *TIMS* and *Driver* signals, and according to the current status of integrity and other conditions described in Table 3.10, performs the switching from one state to another. In this version, all

switching conditions are implemented, while in the previous one, conditions 3 and 9 were missing. This automaton is represented in Fig. 4.8.

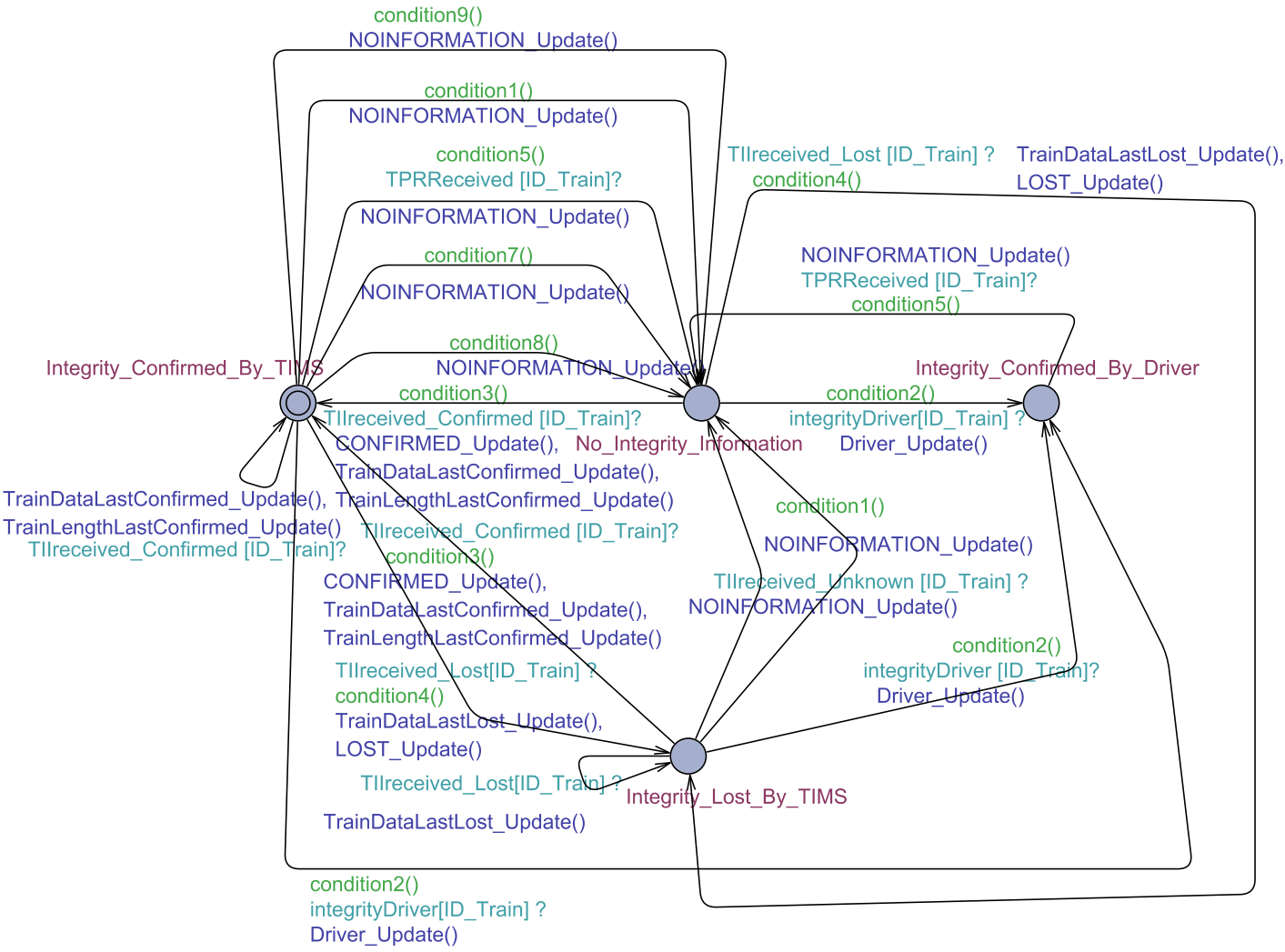


Fig. 4.8. *IIM_StatusManagement* automaton

- "OB_IIMUpdating" automaton represents a part of the behaviour of the on-board function "Integrity_Information_Management". At initial state, it waits for INTEGRITY_CHECK_TIMEOUT. Then, it sets the value of the variable LOC_sentII to the value of the variable $LOC\_CurrentIntegrityStatus$ and sends this information (LOC_sentII) to "OB_TPRManagement" automaton through the channel $'integrityInfoRecv'$. "OB_IIMUpdating" automaton is slightly modified from the previous version and is represented in Fig. 4.9.
- "OB_TPRManagement" automaton represents the behaviour of the on-board function "Train_Position_Management". This automation is responsible for sending to the "TS_TrainManagement" automaton a train position report which includes, mainly, the train position, the train speed and the train integrity information. It receives from "TS_TrainManagement" automaton a request for Position Report $'getTPRRequest'$ and it receives the integrity status from "IIM updating" automaton

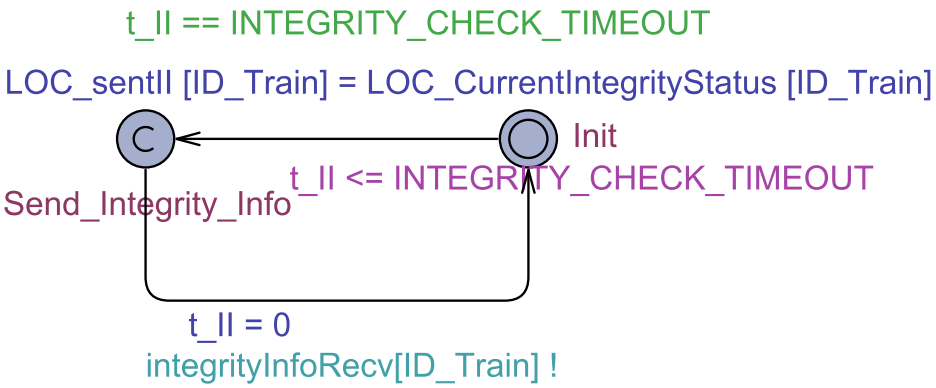


Fig. 4.9. "OB_IIMUpdating" Automaton

$'integrityInfoRecv'$. It sends a request for location to "Ext_TrainLocalizationUnit" automaton. In the state *Wait_Train_Localization_Unit_Position*, it waits for a reply. When receiving $'positionReceived'$ from "Ext_TrainLocalizationUnit" automaton, it fills the variable $msgTPRReceived$ which contains the train position report and sends it to "TS_TrainManagement" automaton using the channel $'TPRReceived'$. "OB_TPRManagement" automaton is deeply modified from the previous version and is represented in Fig. 4.10.

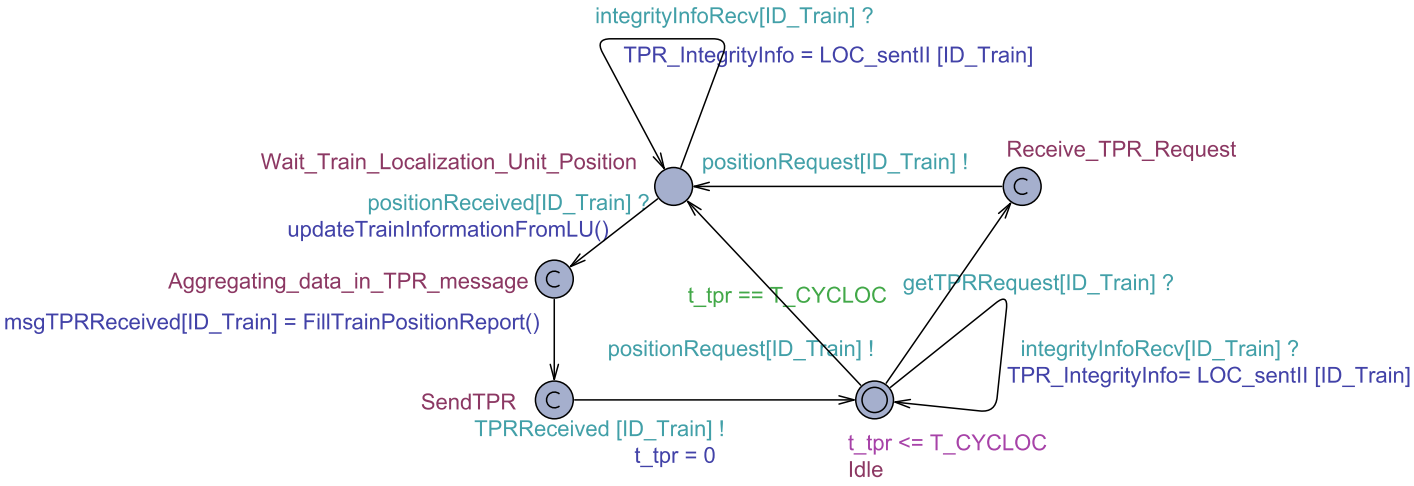


Fig. 4.10. "OB_TPRManagement" Automaton

- "OB_SpeedDistanceSupervision" automaton represents the behaviour of the on-board function "Speed_Distance_Supervision". The boolean variable $LOC\_start[ID\_Train]$ is a local variable allowing to receive train data for the first time by sending a request to "Ext_TrainDataManagement" automaton. If $LOC\_start[ID\_Train]$ is false, train data are already received. The variables $LOC\_Available\_Train\_Data[ID\_Train]$ and $LOC\_Ack\_Train\_Data\_RBC[ID\_Train]$ are initialized by "InitOnBoard" automaton. If no valid train data is available ($!LOC\_Available\_Train\_Data[ID\_Train]$) and the train is at a standstill ($TrainSpeedSentToTPR[ID_Train] == 0$), the "OB_SpeedDistanceSupervision" automaton sends a request to "Ext_TrainDataManagement" automaton to re-sent train

data ($'LOC\_RequestTrainData'$). After reception of train data ($'trainData'$), it sends validated train data ($'VTDReceived'$) to send to "TS_TrainManagement" automaton and waits for an acknowledgement from this latter ($'VTDAck'$). This automaton is deeply improved from the previous version, and it is represented in Fig. 4.11.

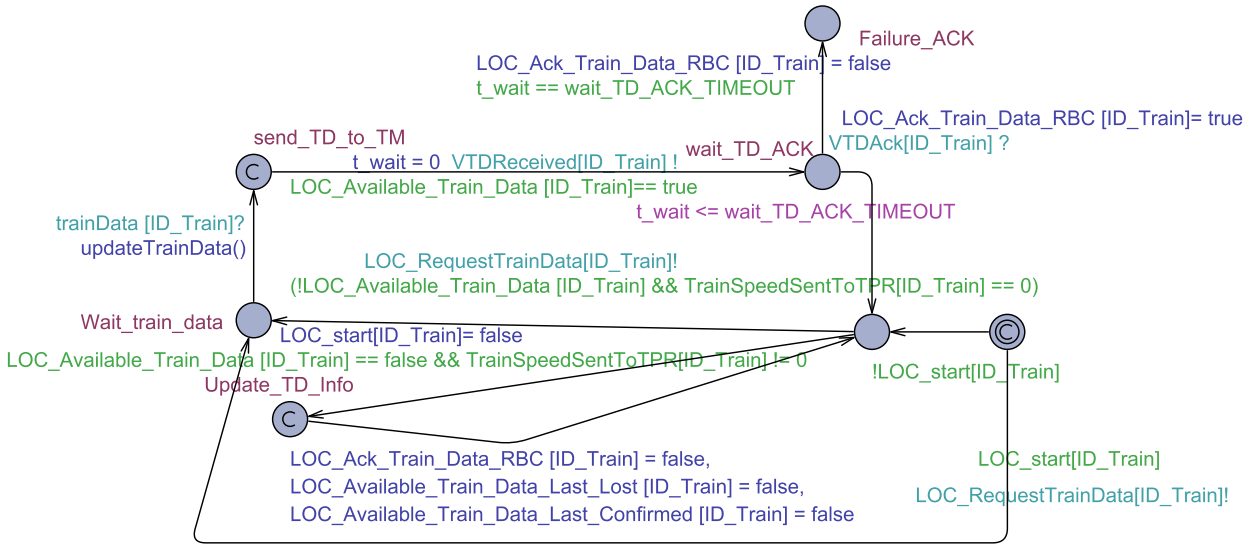


Fig. 4.11. "OB_SpeedDistanceSupervision" automaton

- "InitOnBoard" automaton is designed to initialize onboard variables. The current integrity status is set to confirmed by TIMS using the function CONFIRMED_Update(). Train data are initialized according to the initial conditions defined in Table 7.21 page 138 in [7]: valid train data are always available and valid Train Data have been acknowledged by the RBC. To emulate the generic behaviour of train data and not only the initial conditions, initial values are defined as parameters of the automaton ($int\ ID\_Train$, $bool\ initValueTrainData$, $bool\ initValueACKTrainData$, $bool\ initValueLastLostTrainData$, $bool\ initValueLastConfirmedTrainData$). Then, "InitOnBoard" automaton stores the value of the variable $LOC\_CurrentIntegrityStatus$ representing the current integrity status in the variable LOC_sentII and issues a channel $'integrityInfoRecv'$ to send this value to "OB_TPRManagement" automaton. This automaton is a new model, and it is represented in Fig. 4.12.

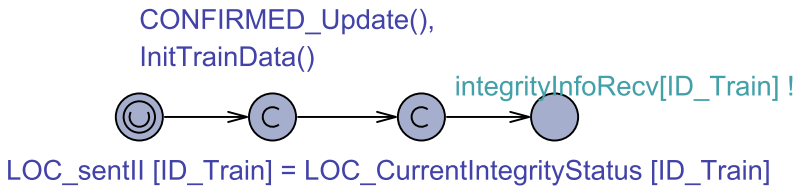


Fig. 4.12. "InitOnBoard" automaton

- "TS_TrainManagement" automaton represents a part of the behaviour of the trackside function "Trains_management". It receives the train position report ($'TPRReceived'$) from "OB_TPRManagement" automaton, receives validated train data ($'VTDReceived'$) from "OB_SpeedDistanceSupervision" automaton, sends an acknowledgement upon reception of train data ($'VTDAck'$) and sends a request for train position report

($'getTPRRequest'$) from "OB_TPRManagement" automaton if the min safe front end of the train exceeds a specific value expressed as the destination of the train $destination[ID\_Train]$. This automaton is updated for previous version and it is represented in Fig. 4.13.

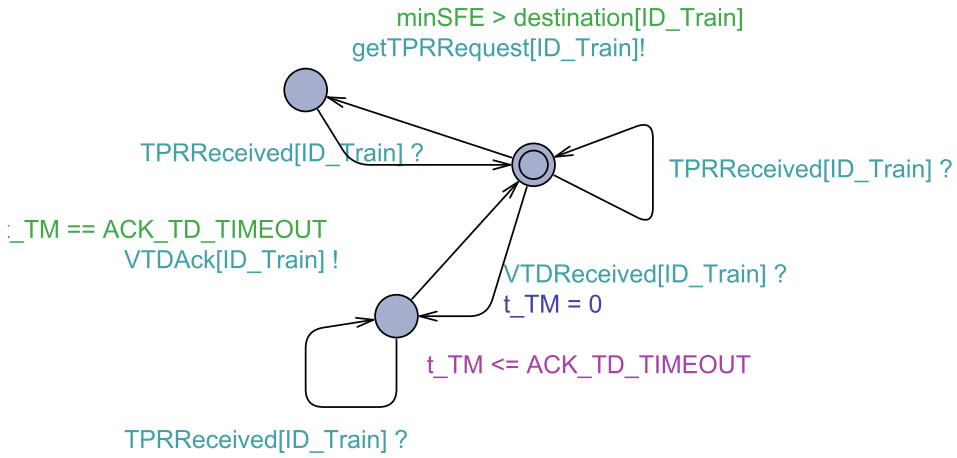


Fig. 4.13. "TS_TrainManagement" automaton

- "TS_MAManagement" automaton is a new model designed to emulate the behaviour of the trackside function "Movement_Authority_Management". It is inspired from[14]. In "TS_MAManagement" automaton, the Movement Authority (MA) is computed after reception of Train Position Report ($'TPRReceived'$) from "OB_TPRManagement" automaton because MA needs the position of all trains to compute future MA. Once MA is computed, a channel $'MAupdate'$ is sent to "Ext_TrainMovement_MAIN" automaton. "TS_MAManagement" automaton is represented in Fig. 4.14.

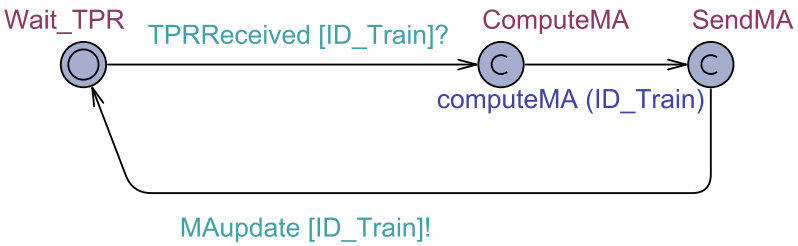


Fig. 4.14. "TS_MAManagement" automaton

4.3.3. Deleted models

From previous version, some automata are deleted: the impact of such a deletion on the overall behaviour is here discussed.

- "Train mode" automaton which emulates the switching between the different ETCS operation modes of the train is deleted. It was designed particularly in order to consider the condition 8 3.10. In the new version, only the $FULLSUPERVISION$ ETCS operation mode is considered. This information is included in the mode profile which is part of the MA structure which is sent by the automaton "MAManagement".

- "Train speed" automaton which emulates the speed of the train by sending the train speed information to the automaton "Speed and Distance supervision" is deleted. It is substituted by "Train Movement automata".

4.4. Points Control

This section describes the internal and cross review of the model and improvements done after the review phase.

4.4.1. Internal and cross Review

The internal and cross review of the models has allowed us to improve existing models. Major changes are made to the "MainAutomaton", "CallRoute" and "SetandLock" automata, while the rest of the automata are new developed. The description of each automaton is provided as below.

4.4.2. New designed and updated models

The current formal model for Points Control consists of eight automata: "MainAutomaton", "CallRoute" and "SetandLock" automata are modified from the ones presented in D2.2; others are newly built and first presented in this deliverable. The description for each of these automata is given below:

- "MainAutomaton" This automaton communicates with the three other automata, shown in Fig. 4.15. This automaton is updated from the one discussed in D2.2 by adding new conditions under which it communicates with 'Nominal', 'Degraded' and 'Sweepable' scenarios (see Fig. 4.15).

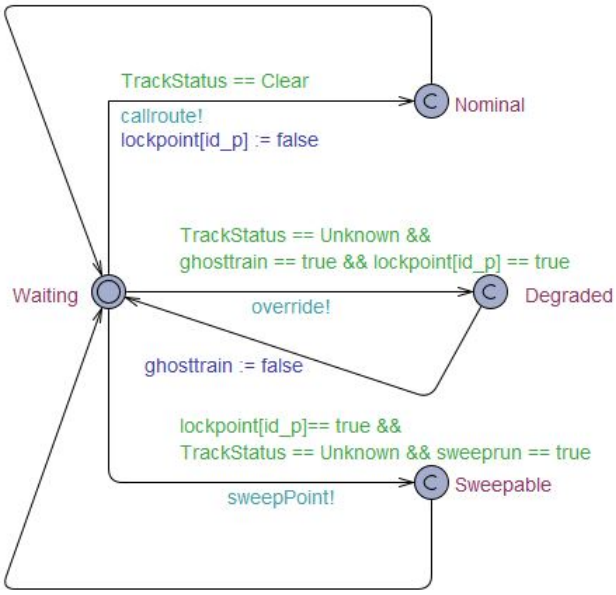


Fig. 4.15. "Points Control Main" automaton

- "CallRoute" is updated from the one presented in [7]; it is also named as "CallRoute" now instead of "CreateRoute". In this automaton, the presence of the involved points in the route to call in an area that is Occupied/Unknown (Track Status) or that is Reserved (Reserved Status), is checked. When the ID of the lock point returns a false value and

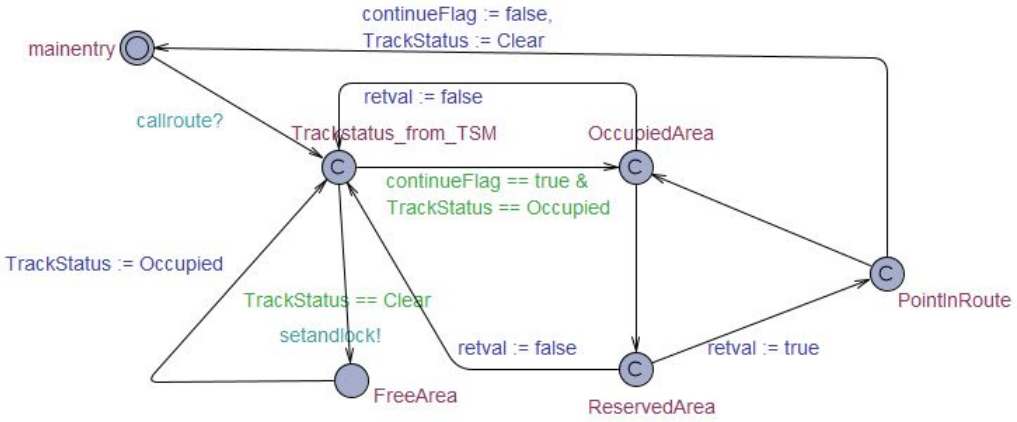


Fig. 4.16. “CallRoute” automaton

track status is clear, the automaton synchronizes with the “SetandLock” automaton, via the synchronization channel *setandlock*. The updated model for “CallRoute” is presented in Fig. 4.16.

- “SetandLock” The “SetandLock” automaton present in [7] has been updated. The variable *EndPosition* captures the point’s position at rest, which can either “Left” or “Right”. After that, the automaton moves to either location *PointPositionLeft* or location *PointPositionRight* by checking the position of point at location *EndPos*. From both locations *PointPositionLeft* or *PointPositionRight* automaton moves to location *NoEndPosition* and stays there until the maximum time is not reached. Once the time is maximum time is reached, then the automaton moves to location *NewPos*. The updated “SetandLock” is shown in Fig. 4.17.

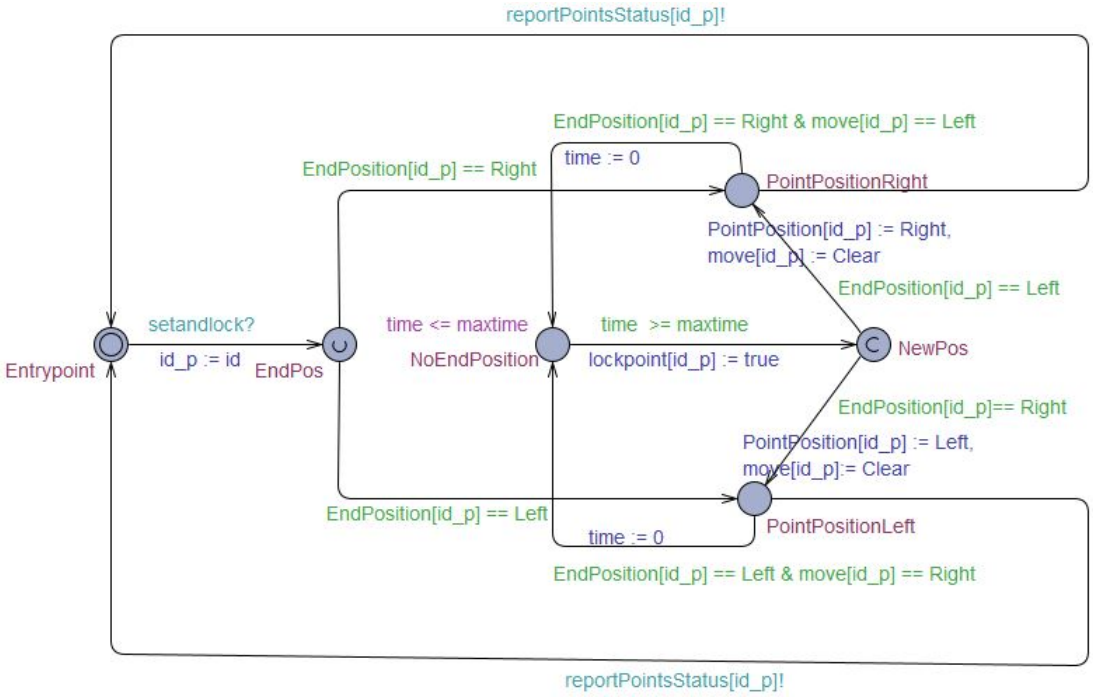


Fig. 4.17. “SetandLock” automaton

- “ReportPointStatus” automaton emulates the Points Manager, sending the set of locked points to the Route Manager automaton as shown in Figure 4.18.

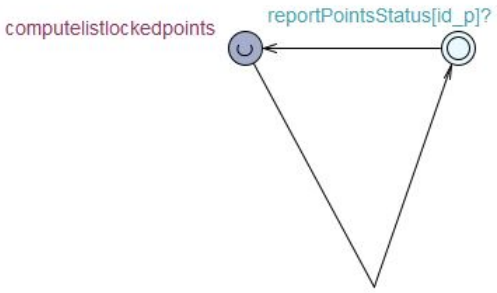


Fig. 4.18. “ReportPointStatus” automaton

- “Signals” This automaton is used to tell the status of the track status area which can be ’Occupied’, ’Free’ or ’Release’ as shown in Fig. 4.19.

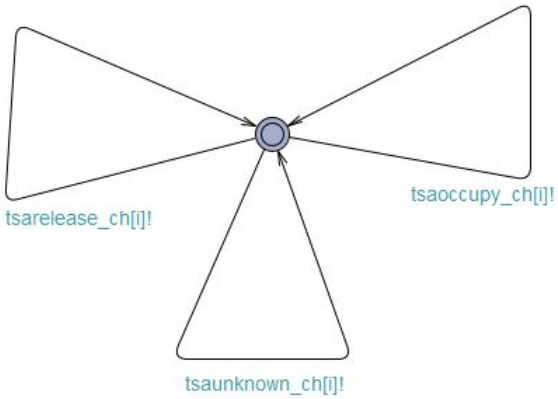


Fig. 4.19. “Signals” automaton

- “LeftRightPP” automaton sets the initial position of a point which can be either ’Left’ or ’Right’ as shown in Fig. 4.20.

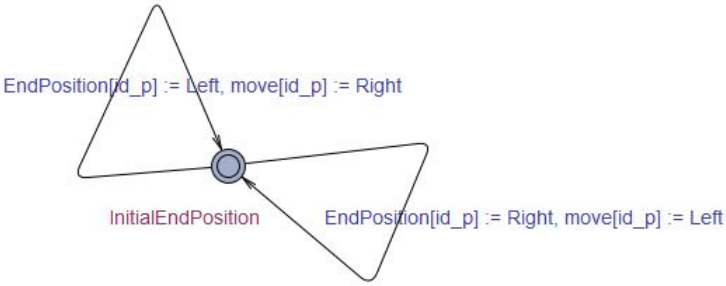


Fig. 4.20. “LeftRightPP” automaton

- “TrackStatusClear” automaton sets the track status area value to clear as shown in Fig. 4.21.

4.4.3. Deleted models

The “Degraded” timed automaton from the Deliverable D2.2 has been deleted, as this scenario has not been considered in the updated UPPAAL model.

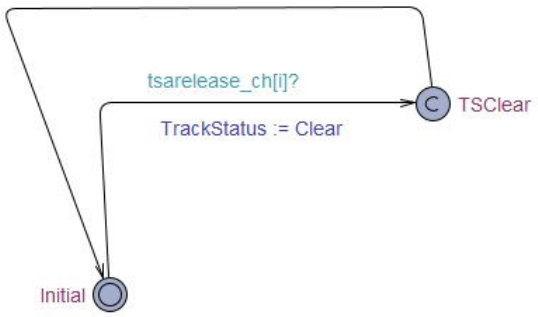


Fig. 4.21. “TrackStatusClear” automaton

4.5. Loss/Restore of Communications

In this section, the revision and integration process of the formal models from D2.2 is reported, focusing on the Loss & Restore of Communication operational scenario (a), made according to the internal and cross review. The main areas of intervention of the review and integration processes are listed below:

- standardisation of data types, variables and communication channels compliant with the data model of D2.2;
- encoding of complex triggers and actions by Boolean functions to improve the readability and maintainability of artefacts;
- concrete implementation of triggers, previously left unspecified, according to a formal specification of the data model;
- revision of the automata, e.g., TrainManager, TrackStatusManager, and CommunicationManager, to solve incompletely specified behaviours and to allow for their integration in the operational scenario;
- modelling of the scenario by replacing some stubs with refined automata.

As a by-product of the above activities, most of the limitations of the formal models reported in Deliverable 2.2 have been overcome.

4.5.1. Updated formal models (OnBoard)

For the description of the formal models of the OnBoard functions for the Operational Scenario Loss & Restore of Communication, refer to Section 4.3.

4.5.2. Updated formal models (TrackSide)

The structure of the formal model of the TrackSide functions for the Operational Scenario Loss & Restore of Communication comprises six automata and two stubs. The description of each automaton is reported below.

The model is intended to have a process instantiation for each train. In other words, each living train will have its own CommunicationManager, TrainsManager, TrackStatusManager, each having the train identifier as a parameter.

The *CommunicationManager* function is required to notify the TrainsManager upon expiration of the connection timers, signalling a loss of communication to the Trackside. Due to the nature of the function, the model was developed taking its interactions with the Trackside into consideration. For this reason, five secondary automatons have been included, with particular attention towards those that represent the track status management and, of course, the train management.

The *TrainsManager* function is responsible for managing a train, more specifically, by communicating with the onboard functions, it determines the location of the train and sends the proper signals (i.e., the Occupied, Unknown, and Release signals) to the Track Status Manager so that the Status Area occupied by the train can be suitably updated.

The *TrackStatusManager* function is responsible for managing the Track Status within its Area of Control. The Track Status represents the information held within the trackside about which parts of the track have an occupied, unknown, or clear status. The collection of all the instances of the TrackStatusManagers, one for each train, provides the Consolidated Track Status of the entire track within a given Area of Control.

4.5.2.1. The CommunicationManager automaton

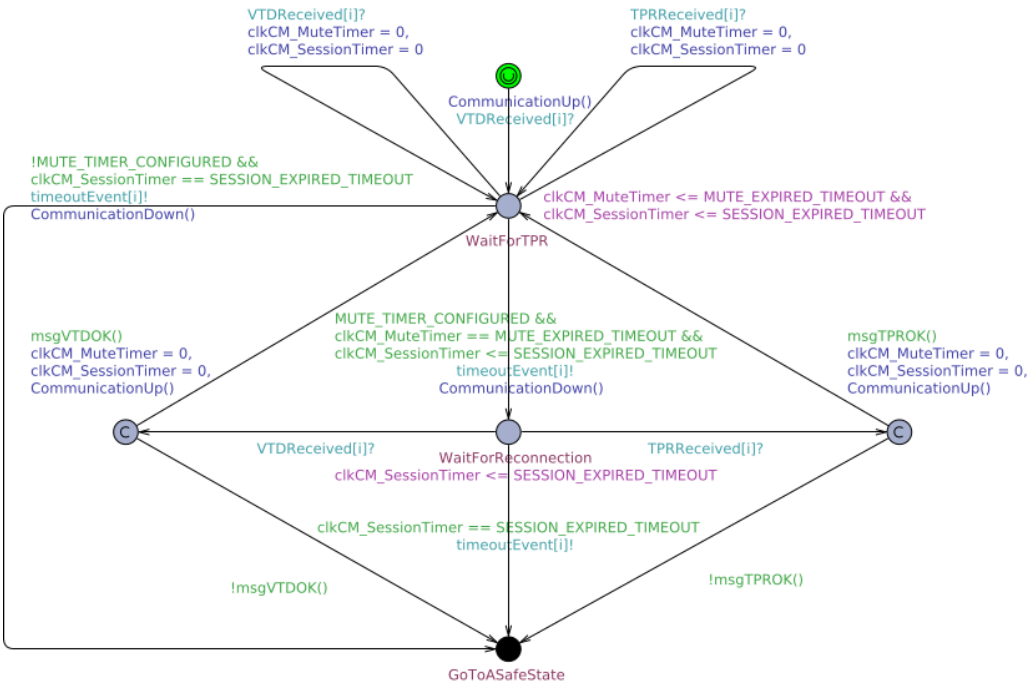


Fig. 4.22. Communication Manager Automaton

The main automaton — depicted in Fig. 4.22 — starts in the initial state where it waits for a validated Train Data Message (see signal `VTDReceived[i]?`) that sets up the communication session and moves to the state `WaitForTPR` where it processes the possible messages from the train, either train position reports (TPRs) or validated train data (VTDs), as long as the *Mute Timer* (and the *Session Timer*) does not expire. Each time a message is received before a time-out, it is considered valid, and the two timers are reset. If the *Mute Timer* is not configured and the *Session Timer* expires, an alert is sent to the `TrainManager`, which manages the situation accordingly, and moves to a safe state (`GoToASafeState`), after setting a flag signalling that the communication went down (`CommunicationDown()`).

If, on the other hand, the *Mute Timer* is configured and its timer expired, it alerts the `TrainManager` that a timeout event has occurred and moves to the state `WaitForReconnection` and sets the flag signalling that the communication went down (`CommunicationDown()`). The automaton, then, stays in that state until either a message from the train is received or the *Session Timer* expires. The invariant associated with that state ensures that, upon expiration of the *Session Timer*, the automaton signals the time-out event to the `TrainManager` and terminates by moving to a safe state. If a valid message from the same train is received (a VTD or a TPR, where the train ID and the train length are those of the original train, `msgVTDOK()` or `msgTPROK()`) before the *Session Timer* expires, then it moves back to the state `WaitForTPR`, after resetting the timers and flag signalling that the communication is up again (`CommunicationUp()`). If the message is not valid, however, it terminates by moving to a safe state.

4.5.2.2. The TrainsManager automaton

The `TrainManager` automaton — depicted in Fig. 4.23 — handles the `ValidatedTrainData` and `TrainPositionReport` messages sent directly by the `Train` together with the `timeoutEvent` message sent by `CommunicationManager` on the connection status of the train. From the initial state, after the reception of a `ValidatedTrainData` message, the local train data are set to the prescribed values and, consequently, the automaton moves to the `Waiting` state. In this state, the automaton is able to perform a series of actions, able to be grouped in three different blocks, those concerning the communication status, those relevant to the update of the `TrackStatusArea` after the reception of a `TrainPositionReport` message, and, finally, the ones associated with the modification of the `Train` length after the reception of a `ValidatedTrainData` message. The different blocks are separately described below.

- If a `timeoutEvent` message is received, depending on whether the *Mute Timer* is configured or not, the automaton either transits to the `WaitForReconnection` state or sends a `clearRSA` message to the `RouteManager` and terminates in the `GoToASafeState` state. In the first case, the reception of a timeout signal represents a mute timer expiration, while, in the second case, is the session timer that is considered expired. At the `WaitForReconnection` state, the automaton is waiting for proper `ValidatedTrainData` and `TrainPositionReport` messages in order to attempt reconnection. If this happens before the expiration of the session, it proceeds to one of the phases described below. If the received messages are considered invalid (see conditions `!msgTRPOK()` and `!msgVTDOK()`) or the session expires, the connection is considered permanently lost and the automaton terminates in the `GoToASafeState` state after sending a `clearRSA` message to the `RouteManager`.
- When a `TrainPositionReport` message is received, the local train data are updated accordingly (including the `MaxSFE`), a `TSAoccupy` message is sent to the `TrackStatusManager`, and a `recvTrainLocation` message is sent to the `TrafficManagementSystem`. if the integrity of the train is confirmed (see condition `IntegrityConfirmed()`), the `CRE` is updated as well and a `TSArelease` message is sent to the `TrackStatusManager`. Otherwise, a `TSAunknown` message is sent to the same machine. At this point, if the position is considered correct, the automaton returns to the `Waiting` state, else, it terminates in the `GoToASafeState` state.
- When a `ValidatedTrainData` message is received, the new `Train` length is eval-

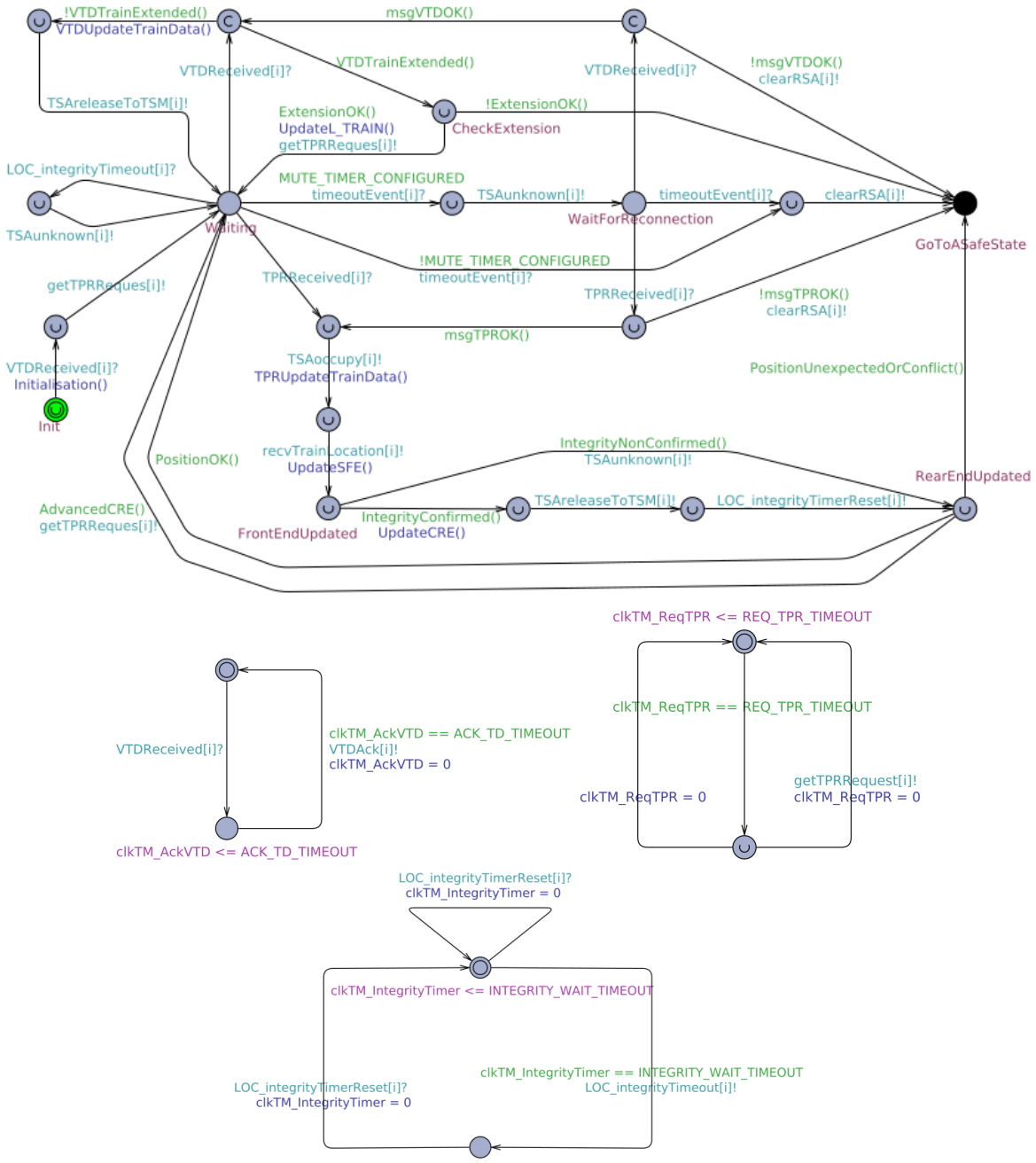


Fig. 4.23. Train Manager Automaton

uated. If it shorter than the previous one, then the train data are updated and a `TSArelease` message is sent to the `TrackStatusManager`. If it is longer, instead, a check of the extensions is performed (see condition `ExtensionOK()`). If it considered correct, the train data is updated accordingly, a request for a new `TrainPositionReport` is sent to the Train, and then the automaton returns to the `Waiting` state. Otherwise, it terminates in the `GoToASafeState` state.

4.5.2.3. The TrackStatusManager automaton

The TrackStatusManager automaton — depicted in Fig. 4.24 — manages the track status information for the track status associated with trains (array variable `varTrackStatus[]`).

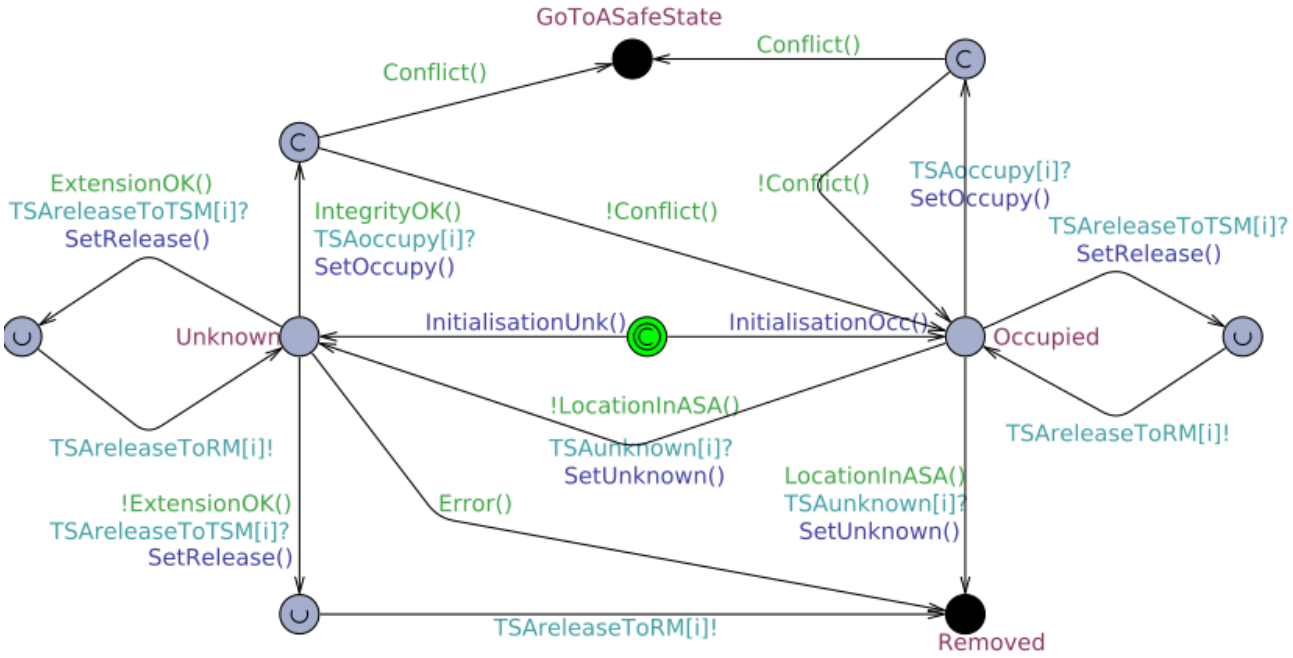
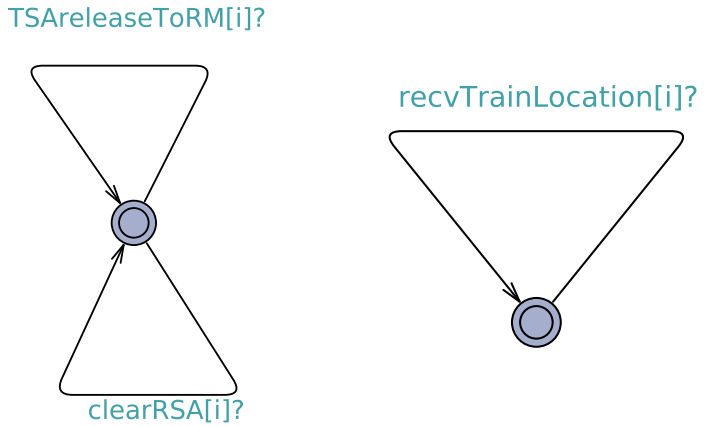


Fig. 4.24. Track Status Automaton

The status is updated in accordance with the interaction with the TrainsManager. Each area can be either in a `Occupied` or `Unknown` state. The automaton can be initialized with an area in one of the two states. In `Occupied` state, when a message is received on the `TSAreleaseToTSM` channel, the starting point of the occupied area is updated to the current CRE of the train. Otherwise, if a message on the `TSAoccupy` channel is received, the ending point of the occupied area is updated to the current `MaxSFE` of the train and a following check on the raising of a possible conflict with areas occupied by other trains is performed. In case of a conflict, the `GoToASafeState` is entered. The `Occupied` status commutes to `Unknown` when a message is sent by the TrainsManager on channel `SetUnknown` and the train is not located in an active shunting area. An additional side effect is the update of the starting point of the area to the current `CRE` and, in case a loss of communication has occurred, the end point of the unknown area is set to `EoA`. If, instead, the train is located in an active shanting area, then the status is commuted to `Removed` and the start and end points of the area are updated as in the previous case. An area in `Unknown` status change to an `Occupied` status because of a TrainsManager request (after a restoring of the lost communication) if an additional check of integrity is fulfilled. A request of release issued by the TrainsManager in the `Unknown` status may lead to the same side effect on the extent of the area as in the `Occupied` status if the extension is admissible, and it leads to a `Removed` status, otherwise.

4.5.2.4. The RouteManager and TrafficManagerSystem automata

RouteManager and TrafficManagerSystem (see Fig. 4.25) are two automata are simple stubs for the sole purpose of keeping the other automatons alive by intercepting messages sent by the automata described above to the corresponding functions.



(a) Route Manager Automaton

(b) Traffic Manager Automaton

Fig. 4.25. Additional Stubs

4.6. Loss/Restore of Communication - Scenario (b)

4.6.1. Internal and cross Review

The internal cross review aimed at reviewing the entire SAN model, including both structural elements (places, transitions, input and output gates) and C++ code, to validate the modelled behaviours regarding realistic situations (the SAN formalism is described in [6]). The cross review mainly consisted of the generation and the analysis of simulation traces of some simple situations that could be easily understandable and handled, such as the service of one and two trains. This allowed us not only to correct some bugs, but also to increase the level of detail of the considered behaviour by mainly updating the C++ code. In addition, SAN model has been extended to represent the Loss and Restore of Communication - Scenario (b), as described in the following. The cross-review of this model involved TUD in joint collaborative work between WP2 and WP4 to revise the modelled behaviour and the values needed to instantiate the model.

4.6.2. Updated model

The SAN introduced in [7] models the movement of the trains along a straight line. The model consists of composed sub-models, including more instances of the train on-board unit and communication model (one per train in the fleet) and just one instance of the trackside model. The changes described in Section 3.4 impact the on-board and communication model. The updated model is shown in Figure 4.26. These changes required extending the model with the elements enclosed in the red box. This portion of the model mainly considers the presence of a driver who is asked to activate a braking, as well as the continuous supervision performed on-board and possible activation of the emergency braking.

The function modelling the loss and restore of communication is associated with the $TPRNetworkDelay$ activity. Let us recall that the extended places (the orange circles) represent global variables and that the composition of the trackside and the on-board sub-models is mainly realized through these shared variables. The trackside sub-model is shown in Figure 4.27 to facilitate the understanding of the modelled behaviour. In this model, the output gate named $updateTSAsAndgiveMAs$ is updated; this gate models the function of extending

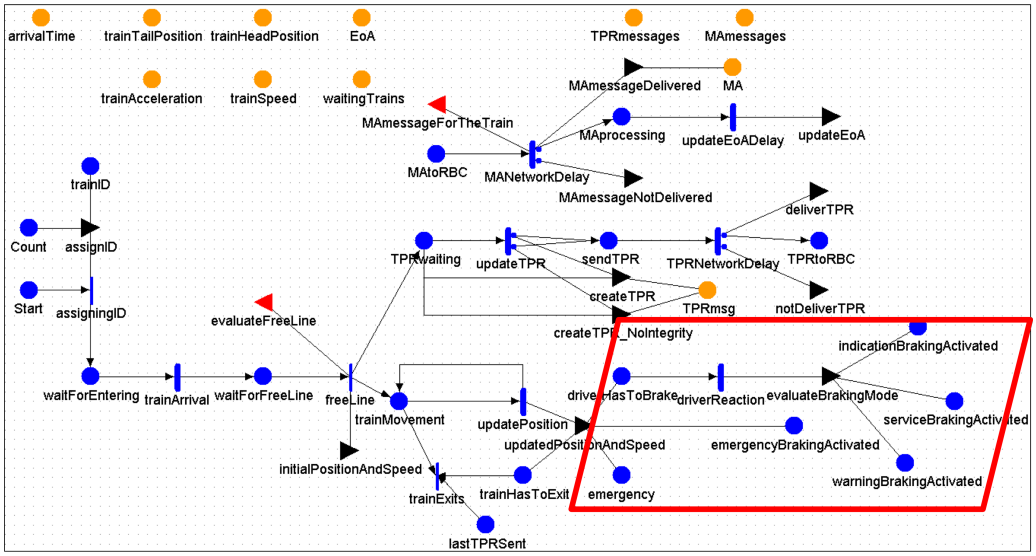


Fig. 4.26. SAN on-board unit + communication atomic updated model

Track Status Areas. As described in Section 3.4, the revised version of the model considers a more accurate behaviour, where the update of Track Status Areas is performed regarding the free area in front of the train, up to a certain maximum extent.

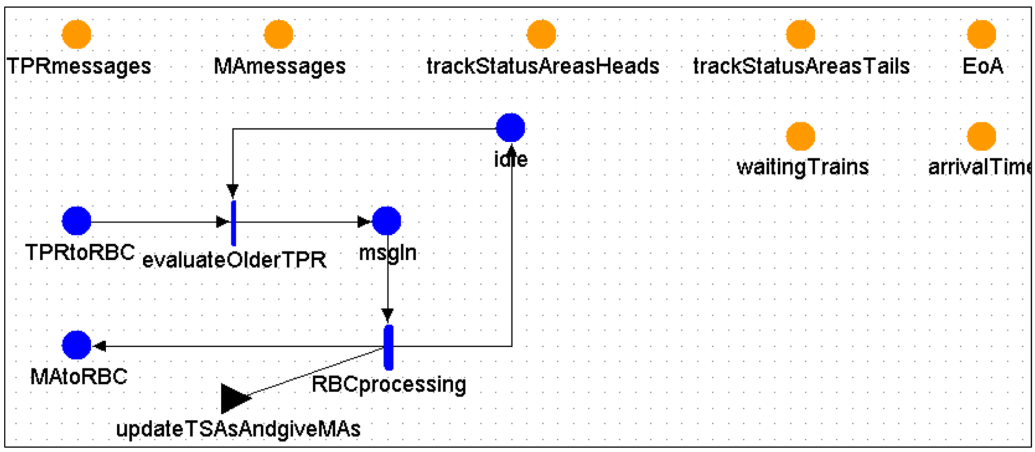


Fig. 4.27. SAN trackside model

In the following, the two sub-models are described in more details.

4.6.2.1. The on-board and communication sub-model

The modelled behaviour is partially presented in [7]. In the left part of the model, a unique ID is assigned to each train (NID_ENGINE). Then two paths go in parallel. Along the lower path of Figure 4.26 models a step-by-step movement of the train on the line. The upper path models the communication with the trackside.

As for the train movement, at each instant of time, the train evaluates its current position. Then it computes its distance to the EoA and the speed that it has to take at the next step. When the train reaches the end of the line, it has to exit it.

The extension of this behaviour here introduced considers some braking curves and the driver's reaction time: periodically, the train evaluates the Emergency Braking and the Indi-

cation Braking distances (the current model simulates a uniformly decelerated motion). If the distance to the EoA is higher than the Indication Braking distance, the train continues its movement. If the train is at the Indication Braking point, the driver is asked to brake (after a certain reaction time). When the driver intervenes, the train follows the related braking curve (indication, service or warning in the current version of the model). If the train is at the Emergency Braking point, the onboard unit enables the emergency brake and the train stops along the line.

The onboard unit continuously generates the Train Position Report message. As for the communication between the onboard and the trackside along the upper path of the model, the communication delay on the messages exchanged between the train and the trackside are considered. The main difference between the updated model and the version discussed in [7] is that the function associated with one of the cases of the transition $TPRNetworkDelay$ models the loss and restore of the communication.

In addition, a Fault Tree approach has been considered with TU-DELFT, aiming at integrating a Fault Tree Analysis (FTA) with the SAN model [15]. At the current state of this research, the following failure probabilities are expected to be evaluated by an FTA: a) the communication network does not deliver a message (TPR or MA), and b) the train does not confirm its integrity.

4.6.2.2. The trackside sub-model

The trackside sub-model is not changed regarding what was presented in [7], except for the introduction of a more accurate function for the update of the Track Status Areas. It does not model the full complexity of the trackside, but it generates the events needed for the onboard sub-models. Briefly, the trackside sub-model is responsible for evaluating the TPR messages received by the train, processing them, updating the Track Status Areas and sending back MA messages to the train.

4.6.2.3. Model data

The model requires the availability of some data beyond the parameters strictly identified for the Loss and Restore of Communication scenario described in Section 3.4 as the proposed model simulates the march of the trains along a *straight* line. Some model data has been already identified in [7]. They are summarized for clarity's sake in Table 4.3. The new ones, introduced according to the changes done to fit the objectives of the scenario, are in bold. The ranges of the model data may be changed as needed, the values used to perform the simulations are reported in the next section, where some studies are described analysing the Loss and Restore of Communication impact on the system capacity and quality of service.

Model data are mainly related to the components identified in Section 3.4: the trains, the track, the trackside and the communication between the trains and the trackside. Of course, the scenario is modelled at the system level but allows studying of the impact of different system configurations at a high level of abstraction. Note that values related to times are in seconds and distances in meters in the values reported in Table 4.3.

The Loss of Communication scenario required the addition of new parameters to the analysis. They are reported in Table 5.1.

4.6.2.4. Variants

As described in Section 3.4, this scenario also requires the analysis of possible variants based on different market segments. These variants have been defined together with TUD

Table 4.3: Loss and Restore of Communication - SAN model data

Component	Model Name	Description
Communication Network	netNotDeliveryProb	Probability that a message is lost
Communication Network	MAnetDelay	Trackside to Train communication time
Communication Network	TPRnetDelay	Train to Trackside communication time
Communication Network	TPRUpdatePeriod	Train Position and Integrity Report update time (may include GNSS)
Track	lineLength	Length of the track
Trackside	RBCprocessingTime	Trackside computation time
Traffic Management System	TrainScheduling	Time between two successive trains
Traffic Management System	safetyMargin	Additional safety distance between subsequent trains
Train (Onboard)	EVCProcessingTime	On-board unit computation time
Train (Rolling stock)	trainLength	Length of the train
Train (Rolling stock)	maxTrainSpeed	Maximum Train Speed
Train (Rolling stock)	maxTrainAcceleration	Maximum train acceleration
Train (Rolling stock)	indicationBraking	Indication train deceleration
Train (Rolling stock)	serviceBraking	Service train deceleration
Train (Rolling stock)	warningBraking	Warning train deceleration
Train (Rolling stock)	emergencyBraking	Emergency train deceleration
Train Integrity Management System	integrityNotConfirmedProb	Probability that the train integrity is not conformed by TIMS
Train Integrity Management System	nTrains	Number of trains moving on the track
Driver	driverReactionTime	Speed visualization time + human time to interpret and react to the indication
Step movement	positionUpdatePeriod	Duration of a step

Table 4.4: Loss and Restore of Communication - additional model data

Component	Model Name	Description
Communication Network	LossComm_trainId	Id if the train affected by the loss of communication, in the interval $[0, ..., nTrains[$
Communication Network	LossComm_startTime	Starting time of the communication loss (the train is muted from this time instant)
Communication Network	LossComm_duration	Duration of the loss of communication (after this this time interval the train sends the TPR to the trackside)

and include high-speed, mainlines, regional, urban and freight. Each scenario considers different features of the convoys and of their service. In the SAN model, the setting up of a scenario is obtained by setting a subset of model data at the values specific for a given market segment. Table 4.5 reports the parameters used for the studied variants. As many different values for these parameters are available from different sources, it is important to underline that the objective here is to perform a sensitivity analysis by varying the values of the parameters of interest, and that *Table 4.5 does not claim to report the conventional signalling and rolling stock values*. The maximum speed and the acceleration values are from [16].

Table 4.5: Market-Segment dependent model data

	HS	Mainlines	Regional	Urban	Freight
trainLength sample $[m]$	327.6	115	96	130	192.8
trainHeadway $[s]$	481.2	182.5	156.0	114.4	350.1
maxTrainSpeed $[m/s]$	83	39	33	22	28
maxTrainAcceleration (maxSpeed) $[m/s^2]$	0.069	0.139	0.450	0.697	0.060

5. Verification & Validation

This chapter is devoted to the Verification & Validation of the designed models considered in this deliverable. Three different OPSs are considered: Loss of Train Integrity, Points Control and Loss/Restore of Communication (both cases (a) and (b)).

For what concerns the analysis of the first three scenarios, some properties have been analysed. The integration process of the models is complex, and the resulting composed models present a high level of complexity, which makes formal analysis hard to accomplish. Notwithstanding such a complexity, some verification activities have been performed also taking into account formal analysis. Results of this activity are reported in Section 5.1.

The case (b) of the last scenario differs from the previous cases since it is modelled by means of the SAN formalism and analysed by creating a reward model as described in the following. For this scenario, a satisfactory level of analysis has been reached. The results are described in Section 5.2.

5.1. Formal verification of Timed Automata

This section presents some preliminary results related to the verification and validation of some designed models following the composition approach defined in Section 4.2. The first step consists in defining for UPPAAL models a global declaration file. In this file, all variables required for designing formal models are included. The file contains data types, message types, trackside data, onboard data, shared data, trackside channels, onboard channels and shared channels. This file is used then for the integration, which is based on shared data and channels.

Local properties check the required behaviour of some automata in just one OPS. Global properties are analysed by putting together many automata from different operation scenarios, also checking if the composition is correct.

For the verification process, using UPPAAL, first, the automata that will be instantiated with the required parameters for each automaton is defined in system composition. Then, the verification process is explained. Finally, using the verification process, the result of the verification of each property and how it is verified (simulation or verification) is described.

5.1.1. Description of global declaration file

The global declaration was defined with respect to the data model defined in [7]. This file is mandatory to facilitate the integration part. Thus, a naming convention is needed for all the model elements that are shared between two or more models. Indeed, it is necessary that a sender and a receiver of a message use the same name of the channels. The following naming convention for channels is needed to avoid conflicts in the case of two signals having the same name but related to different functional components: name_as_in_sysml + ”To”+ acronym_of_the_receiving_functional_component (e.g. TSAreleaseToTSM). For global variables, the followed convention is to use the same name of the SysML model. In the case of channels and variables that are ”internal” to a specific UPPAAL model, the idea is to avoid conflicts with global names and, at the same time, to allow a quick identification of such items. The agreed solution was to prefix such items with the prefix ”LOC”.

The global declaration file includes all variables required for designing formal models. Vari-

ables can be classified into the following categories: data types, message types, trackside data, onboard data, shared data, trackside channels, onboard channels and shared channels. For data types, the structure of data is defined. For example, the data type $TrainIntegrity$ is a structure that contains two attributes: Q_LENGTH which contains integrity status and $L\_TRAININT$ which contains the safe train length. In message types, the structure of message exchanged while sending synchronisation channels is defined. For example, the message $MSGPositionReport$ contains the structure of the message position report. It contains two attributes: msg structure which contains the ID of the train (int NID_ENGINE) and PositionReport structure which contains all variables required in a position report. Trackside data contains data required shared only between trackside automata. For example, the variable $LOC\_communicationTimerExpired$ in trackside variable required to state if the communication timer is expired or not. The same reasoning is followed to define the rest of the variables (onboard data, shared data, trackside channels, onboard channels and shared channels).

Listing 5.1 contains an excerpt of the declaration file. The complete file is stored — as all other artefacts here described — at the referenced repository.

Listing 5.1: Global declaration file (excerpt)

```
typedef struct {
  ...
  M_MODE_Type M_MODE;
  TrainIntegrityStatusQualifier integrityConfSource;
} PositionReport;

typedef struct {
  ...
  int L_TRAININT;
} TrainIntegrity;

typedef struct {
  ...
  TrainIntegrity trainIntegrity;
} Location;

typedef struct {
  ...
  AreaExtent extent;
  bool sweepable;
} TrackStatusArea;

typedef struct {
  int NID_ENGINE;
  ...
  TrackStatusArea trackStatusArea;
  Location location;
  PositionReport positionReport;
} TrainData;
```

5.1.2. Loss of Train Integrity & Loss/Restore of communication

This section is devoted to two OPSs: Loss of Train Integrity & Loss/Restore of communication. It can be noticed that the designed models for Loss of Train Integrity OPS which are deeply described in Section 4.3, represent the moving block onboard part. However,

the designed models for Loss/Restore of communication, OPS which are deeply described in Section 4.5, represent the moving block trackside part. The main properties checked on both operation scenarios are reachability properties, which are described in Section 5.1.2.1 and Section 5.1.2.2

5.1.2.1. Local Properties

Some local properties can be verified only in the operation scenario, Loss of Train Integrity. Verifying local properties in the operation scenario Loss/Restore of communication is not possible. Indeed, this OPS requires information from the on-board system, such as validated train data and train position report. Without such information, it is not possible that the train manager automata computes, for example, the max safe front end of a train.

As highlighted in section 3.1.2, the main new designed models for the operation scenario Loss of Train Integrity are models related to the movement of the train and to the function "MA_Management".

Some local properties are verified on Loss of Train Integrity operation scenarios in order to assess its soundness:

1. Is it possible that the end of authority (EoA) in the movement authority MA for the first train (Id = 0) is different from 0? This means that the "TS_MAManagement" automaton receives from "OB_TPRManagement" automaton a train position report and computes accordingly the MA. If $MA[0].EoA == O$ means that no MA is computed.
   This property is expressed in UPPAAL as $E <> MA[0].EoA! = 0$.
2. Is it possible to reach the state *Integrity_Confirmed_By_Driver* in the automaton "OB_IIMStatusManagement"? This property is expressed in UPPAAL as
   $E <> IIM\_Process\_A.Integrity\_Confirmed\_By\_Driver \; \&\& LOC\_AbsTime > 60$.
3. Is it possible to reach the state *No_Integrity_Information* in the automaton "OB_IIMStatusManagement"? This property is expressed in UPPAAL as
   $E <> IIM\_Process\_A.No\_Integrity\_Information \; \&\& LOC\_AbsTime > 60$.
4. Is it possible to reach the state *Lost_By_TIMS* in the automaton "OB_IIMStatusManagement"? This property is expressed in UPPAAL as '
   $E <> IIM\_Process\_A.Lost\_By\_TIMS \; \&\& LOC\_AbsTime > 60$.
5. Is it possible to reach the state *Confirmed_By_TIMS* in the automaton "OB_IIMStatusManagement"? This property is expressed in UPPAAL as '
   $E <> IIM\_Process\_A.Confirmed\_By\_TIMS \; \&\& LOC\_AbsTime > 60$.
6. Is it possible to reach the state *SendTPR* in the automaton "OB_TPRManagement"? This property is expressed in UPPAAL as
   $E <> TPR\_Process\_A.SendTPR \; \&\& LOC\_AbsTime > 60$.
7. Is it possible to reach the state *SendMA* in the automaton "TS_MAManagement"? This property is expressed in UPPAAL as
   $E <> MA\_Process\_A.SendMA \; \&\& LOC\_AbsTime > 60$.
8. Is it possible to reach the state *Failure_ACK"* in the automaton "OB_SpeedDistanceSupervision"? This property is expressed in UPPAAL as
   $E <> SDS\_Process\_A.Failure\_ACK$.
9. Is it possible that the train reaches its destination? This property is expressed in UPPAAL as $E <> (P\_int[0] == 700000 \; \&\& LOC\_AbsTime > 60)$.
10. How speeds of trains evolve over time? This property is expressed in UPPAAL as

$simulate\ [LOC\_AbsTime <= 300; 1]\{V\_int[0] * 0.1, V\_int[1] * 0.1\}$.

11. How trains speeds received by train management evolve over time? This property is expressed in UPPAAL as $simulate[LOC\_AbsTime <= 300; 1]\{(msgTPRReceived[0].positionReport.V\_TRAIN) * 0.1,\ (msgTPRReceived[1].positionReport.V\_TRAIN) * 0.1\}$
12. How locations of trains evolve over time? This property is expressed in UPPAAL as $simulate\ [LOC\_AbsTime <= 2000; 1]\{P\_int[0] * 0.1, P\_int[1] * 0.1\}$.

In general, the reachability of all states for all the automata composing the LTI OPS is checked.

Focusing on the property: "it possible to reach the state *Integrity_Confirmed_By_Driver* in the automaton "OB_IIMStatusManagement"?", its checking thisrequires that condition 2 in table 3.10, presented as a guard in the automaton "OB_IIMStatusManagement", is true. $LOC\_AbsTime > 60$ is to express that time elapses, and the initial conditions are not valid anymore. Condition 2 (table 3.10) seems simple, but implementing and verifying such a condition, requires many interactions between many designed automata. Indeed, $Train\ is\ at\ standstill$ means that the speed of the train is 0. This requires an interaction between "Ext_TrainMovement_MAIN" automaton which represents a part of the behaviour of train movement, the "Ext_TrainLocalizationUnit" automaton which receives train speed from "Ext_TrainMovement_MAIN" automaton and sends it to "OB_TPRManagement" automaton. $Valid\ Train\ Data\ is\ available\ and\ has\ been\ acknowledged\ by\ the\ RBC$ requires an interaction between 'Ext_TrainDataManagement" automaton which sends train data to "OB_SpeedDistanceSupervision" automaton, this latter sends it back to "TS_TrainManagement" automaton and waits for acknowledgement. $Train\ integrity\ is\ confirmed\ by\ the\ driver$ requires an interaction "Ext_Driver" automaton, which issues the channel $'integrityDriver'$ received by the automaton "OB_IIMStatusManagement". This example is just to show that verifying some reachability properties requires a big interaction between designed models, and it is not a trivial activity.

5.1.2.2. Global properties

Global properties are verified on the integration of all FCs that and interest both the OPSs:

- Is it possible to reach the state *Waiting* in the automaton "TrainManager"? This property is expressed in UPPAAL $E <> TS\_TM\_M0.Waiting$. This means that the trackside receives from onboard system Train data and train position report and the initialisation of the trackside system is succeeded.
- Is it possible to reach the state *tprRequested* in the automaton "TS_Initialiser"? This property is expressed in UPPAAL $E <> TS\_Init0.tprRequested$.
- Is it possible that the track status associated to a train is occupied and the integrity status in the received TPR is integrity confirmed $MONITORINGDEVICE$? This property is expressed in UPPAAL as $E <> ((varTrackStatusArea[0].status == OCCUPIED)\&\&(msgTPRReceived[0].positionReport.trainIntegrity.Q\_LENGTH == MONITORINGDEVICE))$
- Is it possible that the track status associated to a train is occupied and the integrity status in the received TPR is lost $LOST$? This property is expressed in UPPAAL as $E <> ((varTrackStatusArea[0].status ==$

$OCCUPIED) \&\&(msgTPRReceived[0].positionReport.trainIntegrity.Q\_LENGTH == LOST))$

- Is it possible do not reach for all paths the state *GoToASafeState* in the automaton "TrainManager", in the automaton "Communication Manager and in the automaton track status Manager. The system moves to this state if a conflict is detected. This property is expressed in UPPAAL as $E[](!TS\_CM0.GoToASafeState\ !TS\_TM\_M0.GoToASafeState!TS\_TSM0.GoToASafeState)$. CM means Communication Manager, TSM means Track Status Manager and TM means Train Manager;

5.1.2.3. Verification process

The verification can be performed either by simulation or by checking properties with the verifier of UPPAAL tool. Local properties instantiate the models involved in the LTI OPS, and verification is performed using a verifier. For the last three local properties, two trains are instantiated. For simulate properties, all channels are broadcast. For global properties, the automaton "TS_TrainManagement" is deleted and all other automata forming both OPSs are instantiated.

5.1.2.4. V&V results

For the first seven local properties, they are all satisfied, which means that the composition of automata works good. For each satisfied property, a trace was generated. All generated traces are checked and they produce the expected behaviour.

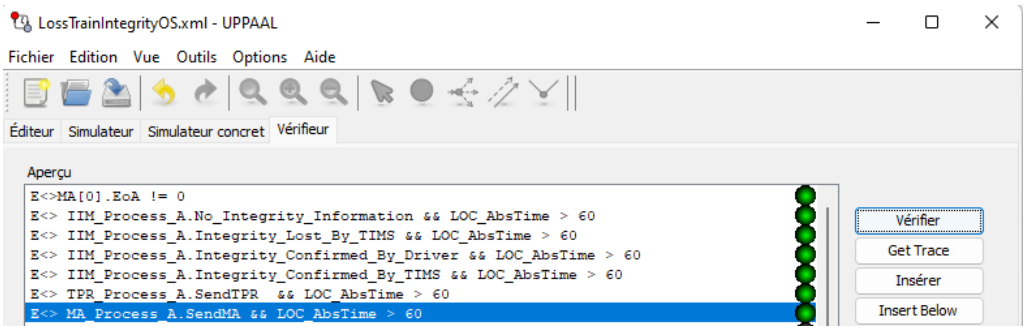


Fig. 5.1. Verification of first 7 local properties using verifier

Property 8 is not satisfied, which means that the system never moves to a failure state. This is the expected behaviour. The "TS_TrainManagement", "OB_SpeedDistanceSupervision" and 'Ext_TrainDataManagement" automata are involved in this analysis and, hence, are instantiated in system composition to verify the property.

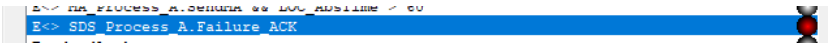


Fig. 5.2. Verification of local property number 8 using verifier

Property 9 can not be verified due to memory exceptions.

For the last three properties, which focus on train movement, it is important to mention the start position and the start speed for all trains. The initial speed is zero for all trains. Initial positions are defined with reference to the first balise. The initial position for the first train (ID=0) is located 8000 m and its destination is set to the position located at 16000 m. The

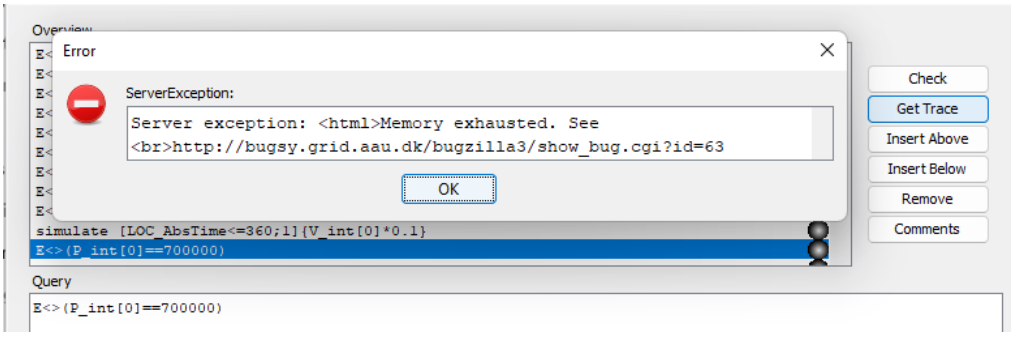


Fig. 5.3. Verification of local property number 9 using verifier

second train, its initial position is set to 1000 m and its destination is set to 13000 m. The braking distance is 5000 m. The last three properties are satisfied (see Fig. 5.4)

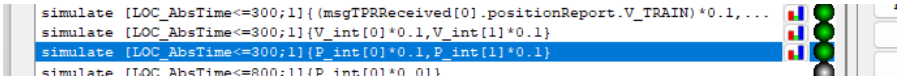


Fig. 5.4. Verification of 3 last local properties using verifier

Fig. 5.5 represents the speed of both trains over time, highlighting that:

- the trains accelerate together as braking distance is respected, and then they brake;
- the first train brakes before the second, considering the travelled distances respectively of 8000 m and 12000 m;
- the curves are continuous curves as the speed is updated each 0.1 s.

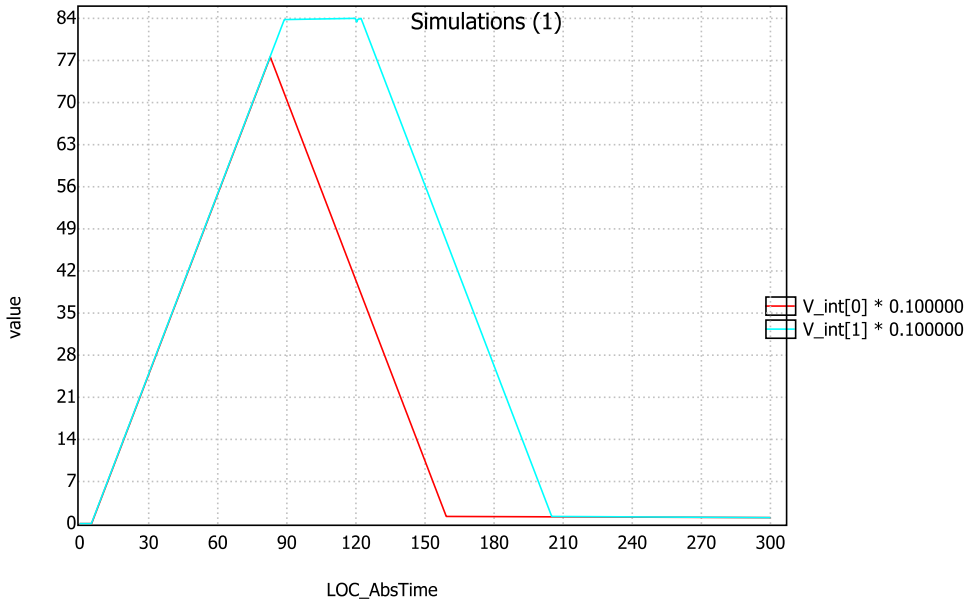


Fig. 5.5. Verification of local property number 10 using verifier

Fig. 5.6 represents the speed of both trains over time received by train management. It can be seen that the speed curves are represented as a step curves. Indeed, the speed value is updated each 5 seconds (interval between two position reports).

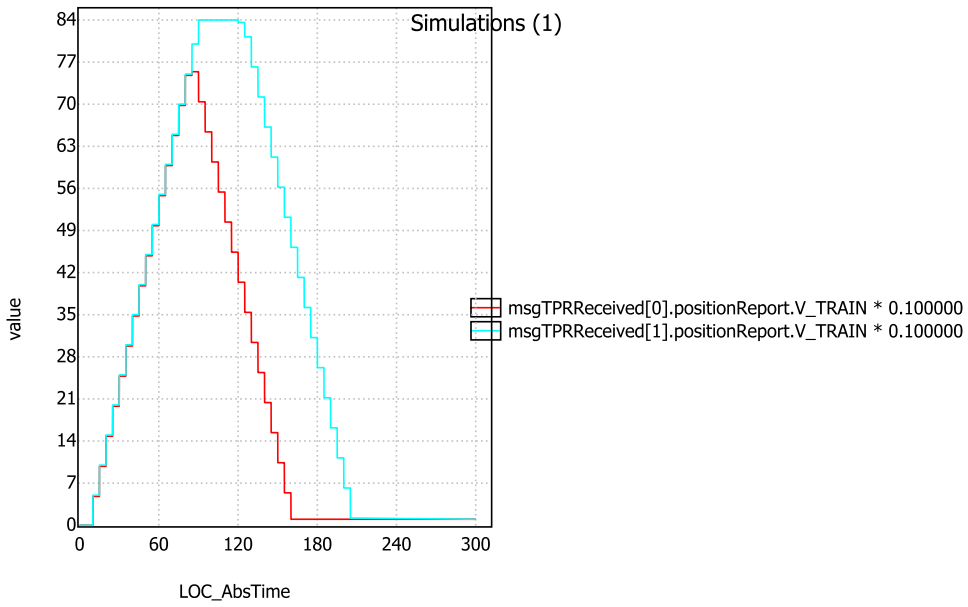


Fig. 5.6. Verification of local property number 11 using verifier

Fig. 5.7 represents the location of both trains over time. The curves are continuous curves as the location is updated each 0.1 s. It can be seen as instantaneous locations. The location of the first train tends to 16000 (its destination) and for the second, it tends to 13000 (also its destination). This is the expected behaviour.

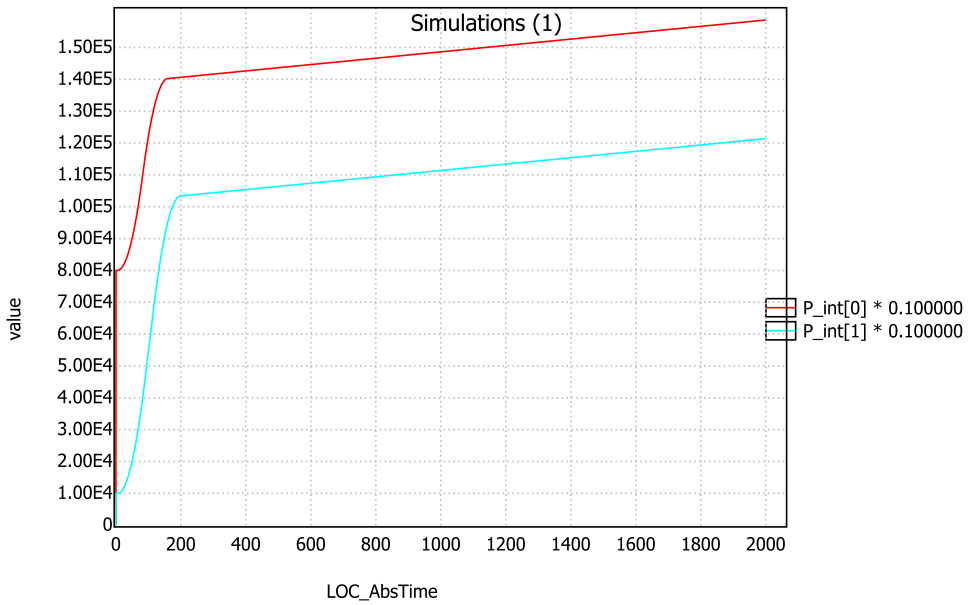


Fig. 5.7. Verification of local property number 12 using verifier

For global properties, they are all satisfied, which means that the integration works good.

Due to lack of time, it was not possible to study verification and validation properties in combined OPSs with many running trains.

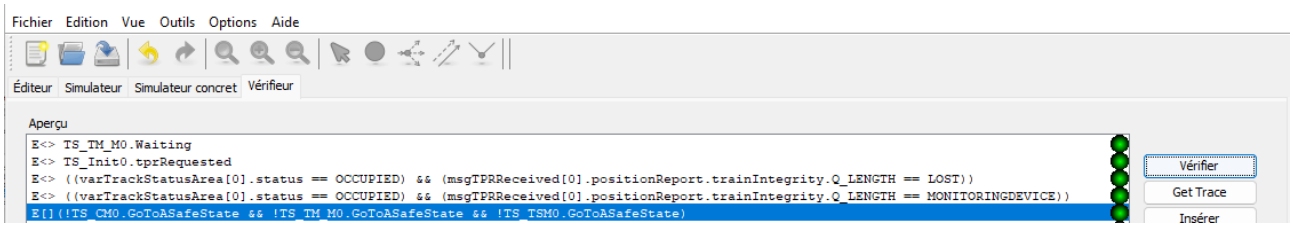


Fig. 5.8. Global properties verification using verifier

5.1.3. Points Control

The Points Management UPPAAL model introduced previously allows the formal verification of important safety-critical properties, by employing the UPPAAL verifier. The first verified property concerns checking that no point belonging to an occupied track is moved to a new position, which is formalized in Computation Tree Logic (CTL) as follows:

$$A[]\, not((TrackStatus == Occupied\, and\, lockpoint1.NewPos) \\ or\, (TrackStatus == Occupied\, and\, lockpoint2.NewPos) \\ or\, (TrackStatus == Occupied\, and\, lockpoint3.NewPos)$$

The second property refers to checking that a clear status of the track will eventually be followed by the corresponding points being locked. In CTL this property is:

$$E <> (TrackStatus == Clear\, imply \\ (lockpoint[1] == true\, and\, lockpoint[2] == true\, and\, lockpoint[3] == true))$$

Both properties are verified by model checking the Points Management network of UPPAAL TA described above, and are satisfied by the model (see Figure 5.9).

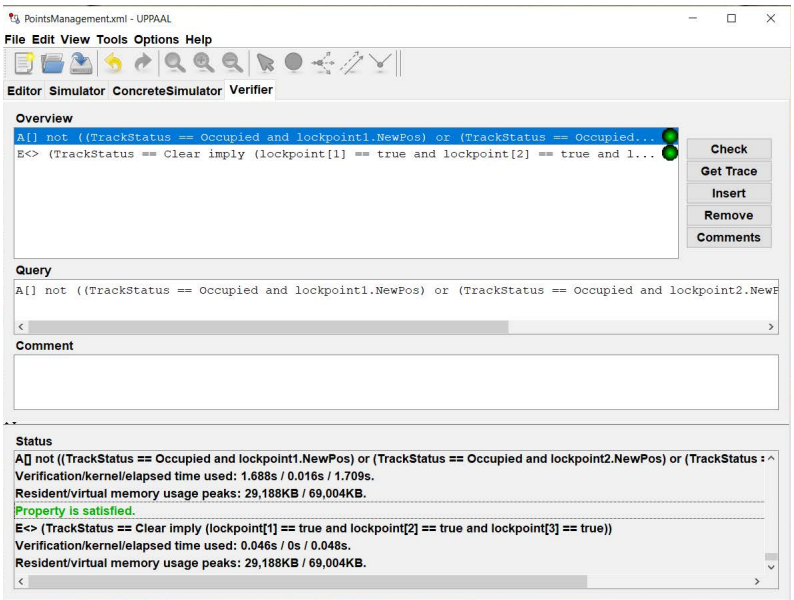


Fig. 5.9. Verification of 2 Points Control local properties using UPPAAL verifier

The second property verifies the reachability of $TrackStatus == Clear$, to avoid the property's trivial satisfaction. The properties are verified by considering the signals

$tsarelease\_ch[N\_TRAINS]$ and $tsaoccupy\_ch[N\_TRAINS]$ sent by $TrackStatusManager$ to $PointsManager$; the former uses them to set the track status to $Clear$, and $Occupied$, respectively.

Due to lack of time, a full verification of the integration scenario involving the UPPAAL formal models of the $PointsManager$, $TrackStatusManager$, $RouteManager$, and $ReservedStatusManager$ has not been carried out yet.

5.2. Loss/Restore of Communication - Scenario (b)

The SAN model introduced in [7] and further developed and discussed in this document provides a practical mean to analyse the MB system behaviour at a high level of abstraction regarding different parameters, to evaluate the performance of the line. The analysis also provides indications about some quantitative aspects that may raise issues in the system. Although SANs are a stochastic extension of Petri Nets, simulation is used as the MB model is not restricted to be of the type of stochastic processes that are analysable by analytic solvers. Some details about the simulation in the Möbius tool [17] are due, to better describe the experimental trials.

5.2.1. Simulating the MB SAN model

The model description in Section 4.6 reports how the SAN model catches the high-level behaviour of the MB systems and introduces the structural elements of the model. The dynamics of the model is defined by providing input and output gates with functions that are implemented in C++ code. Once the composed model has been defined, the following steps must be performed:

- Reward model definition: *measures* must be specified to quantify the model's behaviour through the definition of reward variables. Two types of rewards are possible: Impulse reward, associated with state changes (at activity completion), and Rate rewards assigned to markings. A difference is given between Interval-of-Time or Instant-of-Time measures. In particular, for Rate rewards, the former measures accumulate over the interval of time that the model spends in those markings, whereas the Instant-of-Time measure gives the rate reward associated with the markings at time *t*.
- Study model specification: this step creates studies and experiments. At this aim, single values or a range of values can be assigned to each global variable. Indeed, the aim is to study the behaviour of the system for several values: they represent the parameters of the model. If a range for a variable is provided, separate experiments will be created for each value. So, for example, if two variables may assume two different values, four experiments are created.
- Simulator generation: discrete event simulation is performed. Confidence intervals are generated for the performance variables defined in the Reward model using the replication method for terminating simulation. The simulator will run several batches to generate data for the confidence interval. It will continue to run more batches until all the reward variables have converged to their specified confidence interval or the maximum number of batches is reached. A minimum number of batches can be specified to cope with rare events.

5.2.2. Main objectives of the analysis

In the following, some experiments are presented aiming at providing indications about the impact of a loss of communication on different configurations of the MB system. The performed analyses aim at evaluating the impact of loss/restore of communication on the behaviour of the train with respect to brake events and the number of trains able to exit the line.

In the analysis, a twenty convoys fleet entering the line is considered. A first experiment is done, reporting results in case the communication between the trackside and the train on-board unit is never lost.

In all the subsequent experiments, just one train in the fleet losses the communication with the trackside at the specified instant of time. It is supposed to be the second train entering the line and that the communication is restored after a given delay. Therefore, the first train acts as a forerunner and the impact of the loss of communication is evaluated on the following trains.

According to the operational scenario, the conducted analysis evaluates the following performance indicators defined in Section 3.4 Table 3.33:

- **#Trains**: number of trains exiting the line in the time interval;
- **#Emergency**: number of times that trains exceed the speed imposed by the specific ETCS “emergency” braking curve, within the total simulation time;
- **#Warning**: number of times that trains exceed the speed imposed by the specific ETCS “warning” braking curve, within the total simulation time;
- **#Permitted**: number of times that trains exceed the speed imposed by the specific ETCS “permitted” braking curve, within the total simulation time;
- **#Indication**: number of times that trains exceed the speed imposed by the specific ETCS “indication” braking curve, within the total simulation time.

5.2.3. A baseline simulation

In this section, the railway system configuration adopted in WP4 is used to describe the different steps introduced above and provide a baseline for the subsequent simulations. In the next section, the results from the analyses performed over the MB variants specified in Section 3.4 are presented.

5.2.3.1. Baseline Reward model

The Reward model specifies the measures associated with the performance indicators. Therefore, five performance variables are defined in the Reward model, they are shown in Fig. 5.10.

The performance variables related to braking events are Rate rewards on the markings of the on-board sub-model, as specified in their reward functions. In Fig. 5.10 the reward function associated to the $emergencyBrakings$ performance variable is shown: as the rate rewards is defined to be an instant-of-time measure, it returns the marking of the place $emergencyBrakingActivated$ at time t for each instance of the on-board sub-model. The same definition is given for the three performance variables that measure the number of times the train exceeds the speed limit of the other ETCS braking curve (indication, permitted and warning).

The performance variable $trainExitingLine$, measuring the number of train exiting the line

in the time interval, is instead an Interval-of-time Rate reward, it provides the throughput of the transition $trainExit$ in the given interval of time, over all the instances of the on-board model.

The Instant-of-Time reward function is evaluated at the specified points in time. The Interval-of-Time variables returns the weighted sum of all of the values of the reward function, where each value is weighted by the amount of time the value is present between the starting and ending times of the specified interval.

Fig. 5.11 shows the time points for Instant-of-Time measures, and the interval of time selected for all the experiments described in this section: 20 time points have been considered, 150 seconds apart from each other, in the time interval $[0 : 3200sec.]$. Therefore, the braking related performance variable will be evaluated at each time point, whereas the Interval-of-Time Rate reward will be evaluated at the instant of time 3200.

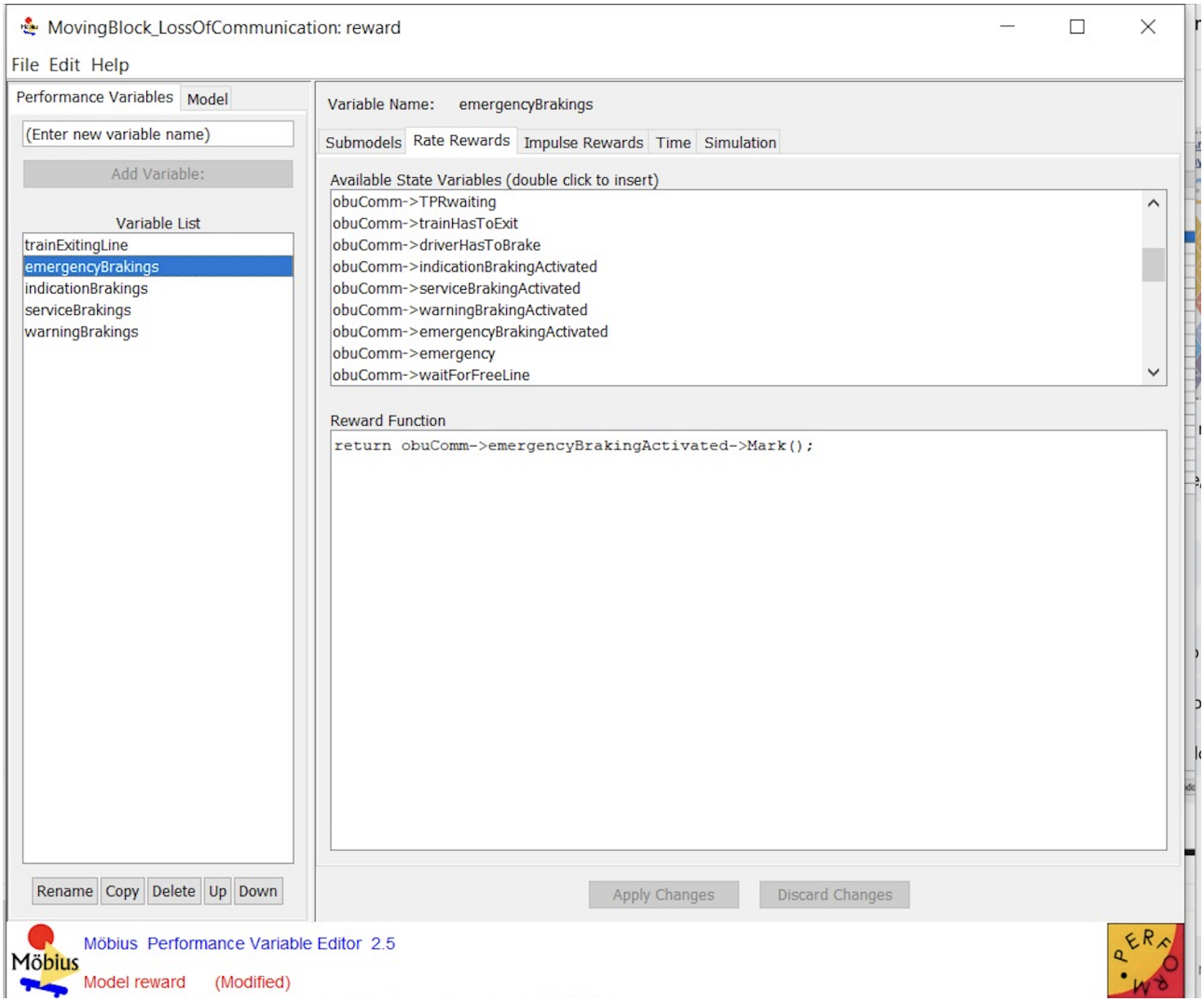


Fig. 5.10. SAN model - reward model

5.2.3.2. Sample Study model

This first analysis was performed considering the case study used in WP4, which is a double-track railway line having a total length of 24 km (24000 m.) with a maximum speed of 80

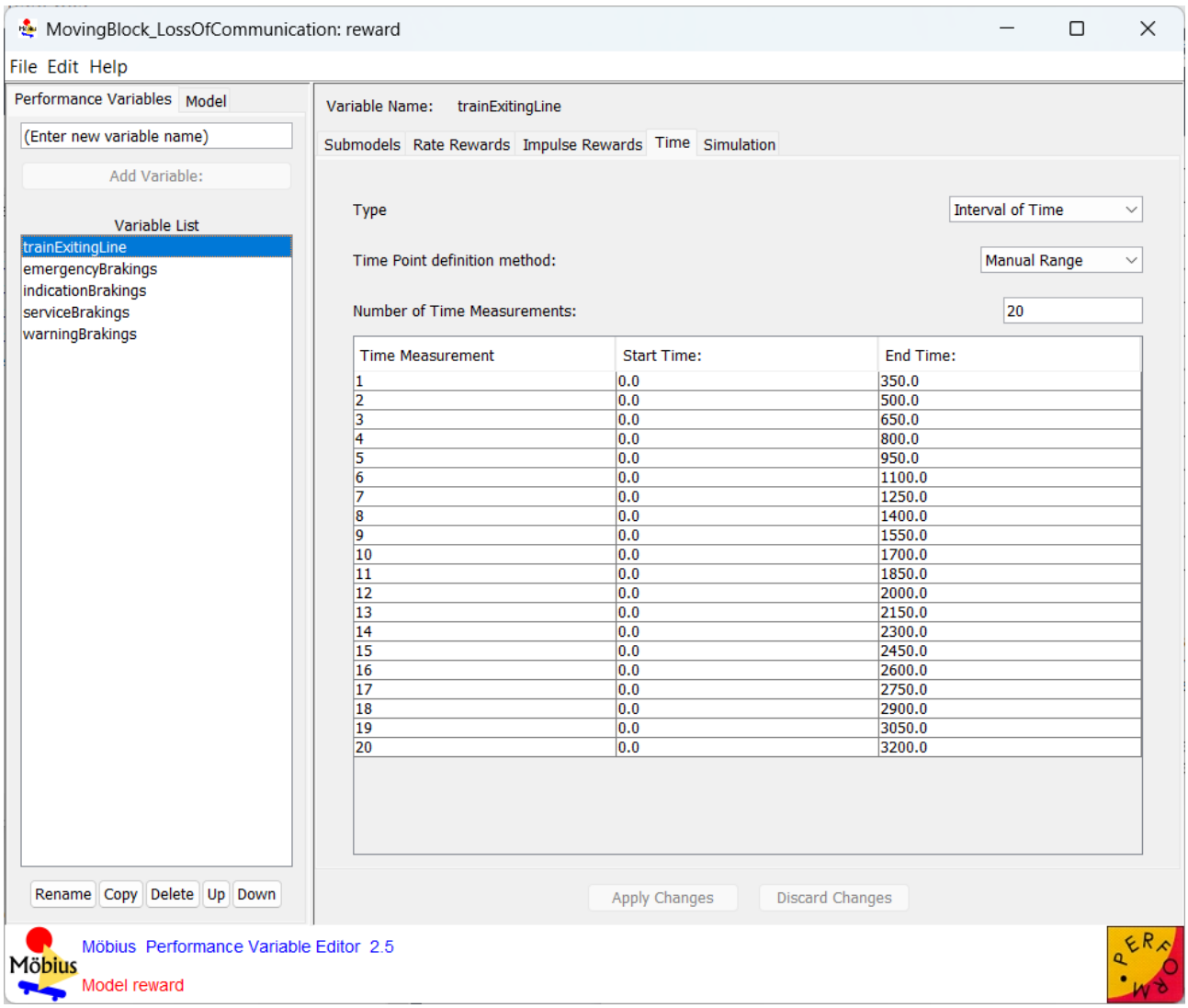
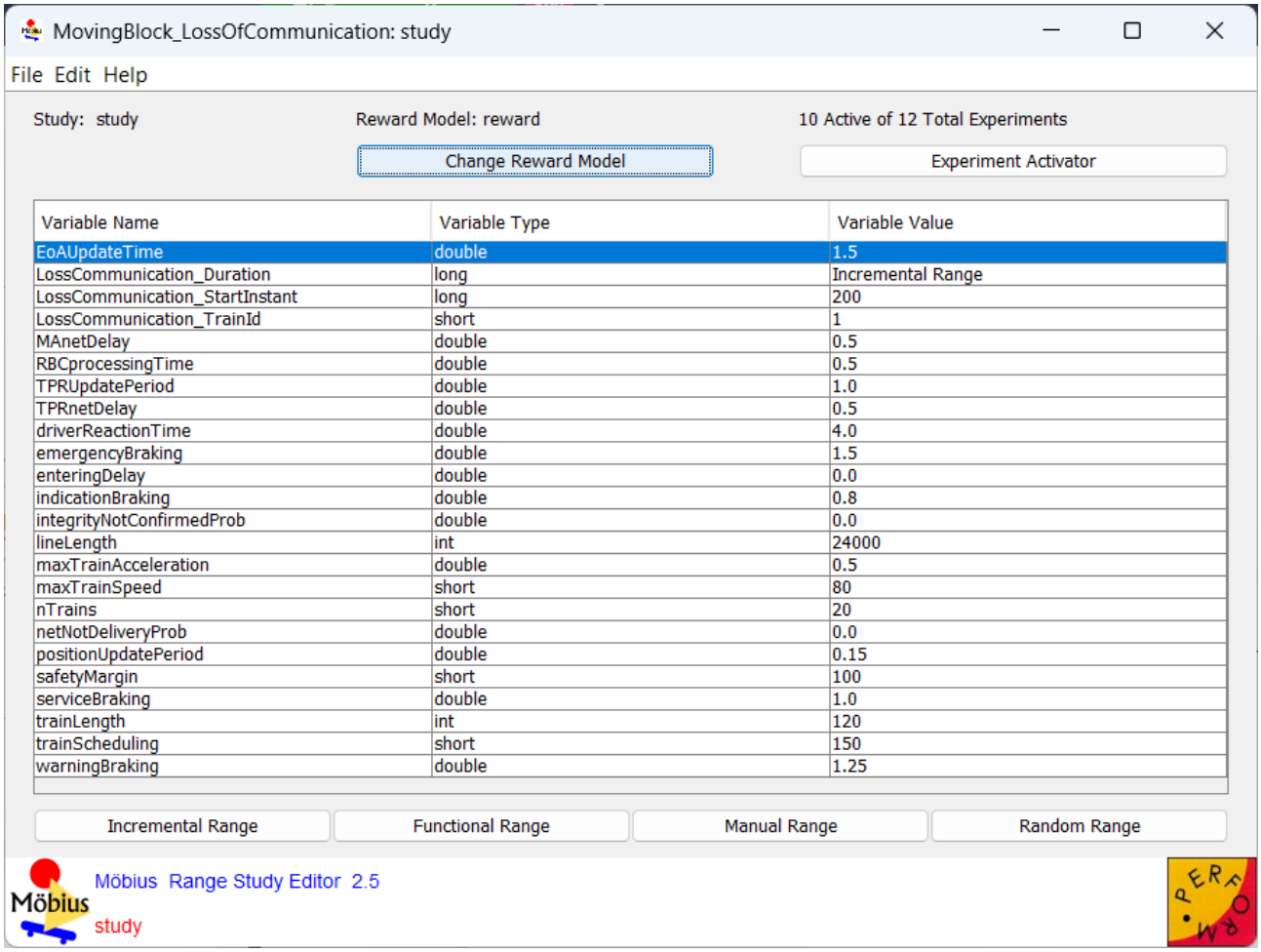


Fig. 5.11. SAN model - reward model

m/sec. (i.e. 288 km/h). A total of 20 trains are analysed which operate along one single direction of movement, having a minimum technical running time of 300 sec. (24.000 m / 80 m/sec.) from origin to destination. Trains run at a scheduled headway of 150 sec., which results in a distancing of 12 km (80 m/sec · 150 sec.). When running in undisturbed nominal conditions, all twenty trains are able to reach their destination at 3150 s (300 s + 19 · 150 s). This means that the nominal timetable cycle is set to 3150 s and the observation time interval (simulated time) of [0 : 3200], obtained by adding a spare interval of 50 sec is enough to ensure the exit of the last train.

Fig. 5.12 reports the study definition for the simulations that have been executed, where the scheduled headway is reported as $trainScheduling$ in the parameter list, and a range of values is provided for the duration of the loss of communication, ranging from 2 to 10 sec., with a step of 1 seconds. Hence, this study generates nine different experiments. In addition, the experiment in which no loss occurs has been also executed. The parameter $LossCommunication\_StartInstant$ and $LossCommunication\_TrainId$, whose values are respectively to 200 and 1, say that the communication loss starts at the time instant 200s for

the train whose ID is 1 (the trains are numbered from 0 to 19). The values of time durations in Fig. are always in seconds and the distances in meters. For example, the RBC processing time in this study is set to 0.5 sec.



Variable Name	Variable Type	Variable Value
EoAUpdateTime	double	1.5
LossCommunication_Duration	long	Incremental Range
LossCommunication_StartInstant	long	200
LossCommunication_TrainId	short	1
MAnetDelay	double	0.5
RBCprocessingTime	double	0.5
TPRUpdatePeriod	double	1.0
TPRnetDelay	double	0.5
driverReactionTime	double	4.0
emergencyBraking	double	1.5
enteringDelay	double	0.0
indicationBraking	double	0.8
integrityNotConfirmedProb	double	0.0
lineLength	int	24000
maxTrainAcceleration	double	0.5
maxTrainSpeed	short	80
nTrains	short	20
netNotDeliveryProb	double	0.0
positionUpdatePeriod	double	0.15
safetyMargin	short	100
serviceBraking	double	1.0
trainLength	int	120
trainScheduling	short	150
warningBraking	double	1.25

Fig. 5.12. SAN model - baseline study

5.2.3.3. Baseline Simulator

A simulator is automatically generated by the Möbius tool according to the provided simulation execution parameters. Among them, the maximum and minimum number of batches. In this regard, it must be considered that when the model is solved by simulation, the system executes multiple times (batches) using different randomly generated event streams and each execution generates a different trajectory in the possible event space of the system. The reward variables are evaluated for each trajectory to create an observation. Statistical estimates of the reward variable are computed from the observations.

For each analysis (i.e., for each experiment defined by the study), the number of batches spans from 1.000 to 10.000, with a relative error lower than 1%.

5.2.3.4. V&V results for the baseline study

For each experiment, first, all the 20 trains can exit the line at the nominal time interval, without any communication interruption. Results are reported in the graph in Fig. 5.13, which plots the number of circulating trains versus the simulated time. This means that all

trains can exit the line when the system works correctly, with the considered delays and processing times.

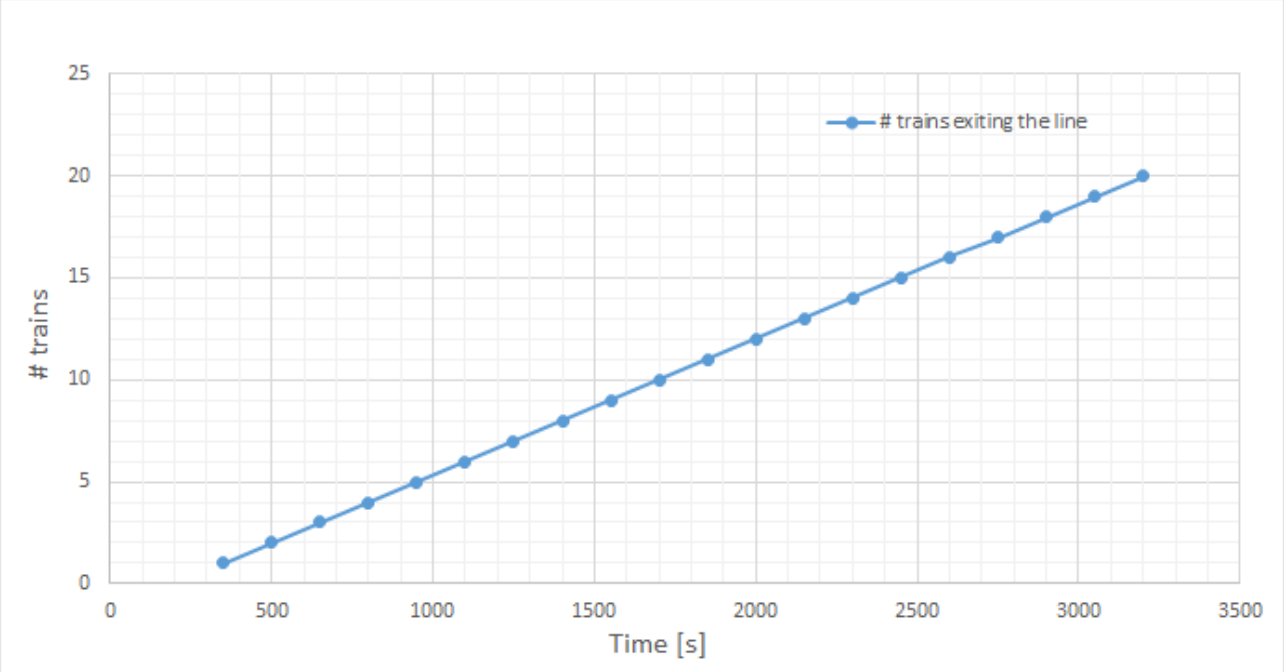


Fig. 5.13. SAN model - #Trains vs simulated time without loss of communication

The simulations also report that a loss of the communication from 2 to 10 seconds has no impact on the number of trains able to cross the line in the considered time interval. In fact, in all the simulations, the 20 trains can exit the line in 3200sec. Given the analysed time measurements in the reward variable, all trains can exit the line with a delay lower than 50sec. The number of measured braking are reported in Table 5.1. Results say that a loss of communication up to 4 sec has no impact on the safe running of trains in the considered conditions since trains are sufficiently separated, while some braking occurs when the loss of communication duration is higher than 4 sec. Moreover, also in the worst case, only one service braking has been obtained.

5.2.4. V&V results for the market variants

Many experiments have been executed to evaluate the impact of the loss of communication scenario on different market segments. The baseline choices have been maintained for all the experiments, where $maxTrainAcceleration$, $trainLength$, $trainScheduling$ and $maxTrainSpeed$ change according to the values reported in Table 4.5. The length of the track is always 24000 m.

Further, the values $LossCommunication\_StartInstant$ are varied to guarantee that this variable is higher than the $trainScheduling$, so ensuring that the second train of the fleet (whose ID is 1) loses its communication after entering the line.

The simulations showed the differences between the market segments.

Starting from High-Speed lines case, it was immediately clear that the number of generated messages and the processing time of the trackside play a crucial role, and the balance needle result to be the EoA extent. Thus, the final aim became to obtain the minimum EoA extent that should be granted by the trackside to avoid the impact of a communication loss

Table 5.1: Loss and Restore of Communication - impact on braking

LossComm_duration	#Emergency	#Warning	#Permitted	#Indication
2	0.0	0.0	0.0	0.0
3	0.0	0.0	0.0	0.0
4	0.0	0.0	0.0	0.0
5	0.0	0.0	0.26	0.22
6	0.0	0.0	0.84	0.15
7	0.0	0.0	1.0	0.0
8	0.0	0.0	1.0	0.0
9	0.0	0.0	1.0	0.0
10	0.0	0.0	1.0	0.0

on the service. Hence, different runs consider different values of the EoA extent, according to increment of 100 m, and the Loss of Communication Duration, in the interval from 2 to 10 sec, according to the scenario specification.

Intuitively, the higher the speed of the train, the longer should be its agreed EoA before it loses communication with the trackside, so avoiding the activation of the braking. Since the maximum EoA that the trackside can grant to a train should be lower than the distance from the preceding train, which is proportional to its speed, it is intuitive to expect that in high-speed lines the trackside could grant longer EoAs.

These intuitions have been confirmed by simulations. As reported in the graph in Fig. 5.14:

1. the correlation between the minimum required MA extent and the duration of the Loss of Communication has been measured;
2. this relationship strictly depends on the market segment (namely, on the values used for the model data).

In fact, in high-speed lines the minimum extension of the EoA is 5 km, which increases to 5.7 km to tolerate a Loss of Communication of 10 sec. Instead, urban lines, where speeds are significantly reduced, require a minimum EoA extent of 0.6 km to tolerate a Loss of Communication up to 5 sec., increased to 800 m. to tolerate 10 sec of a loss of communication. The other market segments are positioned in the middle of these two extremities, as reported in the graph. Of course, the reported results are specifically tailored to the entire set of considered values, also of the additional model data. However, the SAN model represents a valid mean to evaluate the correlation among the entire set of considered variables on the service.

The study that has been performed on the Loss of Communication scenario demonstrates that the SAN model may help to conduct a in-depth analysis of the impact of the loss of communication on the service of a train fleet, regarding signalling system features (i.e., trackside processing time, delays in the communication network, period between subsequent TPRs, driver reaction time) and the set of system configuration parameters (e.g., train headway,

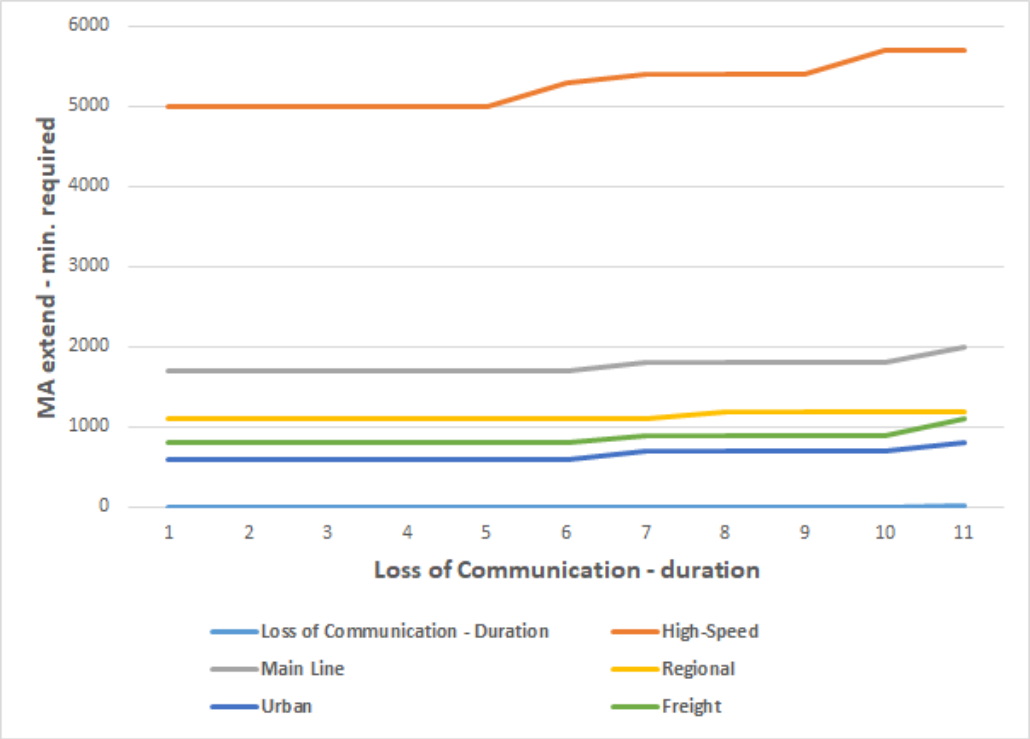


Fig. 5.14. SAN model - #Trains vs simulated time without loss of communication

safety margin, train mechanical features). The analysis described in this document provides an example of the potentiality of the SAN-based simulation approach. Of course, a more detailed model of the system's behaviour that would derive from additional high-level requirements could better support system designers. The SAN model, in fact, represents a good mean to evaluate the trade-off between performance and optimal resiliency regarding the considered scenario.

6. Conclusions

This chapter ends the deliverable, both highlighting the results and discussing their applicability in future contexts.

6.1. Summary of the results

This deliverable presented an integrated approach for the specification, the modelling, and the analysis of various OPSs relevant for operation under MB. All the reported artefacts described in this deliverable are reported in the GitHub repository https://github.com/stefanomarrone/performingrail.

The main obtained contributions can be synthesized as follows:

- the specification of four operational scenarios, that serve as relevant and challenging ”test cases” for the introduced formal modelling approach;
- the development of a formal model libraries constituted by:
  - the refinement and extension of the preliminary formal models already developed in [7];
  - the development of a number of stub models used to emulate the interaction of the main involved actors in the OPSs with extra modules and with the environment;
  - the usage of two different formalisms for the analysis of the ETCS-L3 operational scenarios;
- the conduction of sensitivity analysis based on the SAN model to determine the impact various factors on some performance criteria and investigate performance bottlenecks in ETCS-L3 systems;
- the integration of UPPAAL formal models through a compositional approach that can integrate the models on shared signals and variables.

It is important to highlight that, even though the scope of the analysis was limited to a number of functions and OPSs, the modelling activities showed to be very complex, due to the numerous involved actors and the highly interactive nature of the investigated dynamics. Moreover, as detailed earlier in the relevant sections, the development of various extra models was necessary to emulate the actual investigated behaviours. In addition, due to the involvement of various partners in the modelling activities, an important amount of work was required to align the different models and homogenize them for integration purposes.

Besides, the analysis of the UPPAAL models presented several issues. Due to the compositional nature of this approach, the resulting models may not be optimized for the analysis, requiring much time for debugging. As a consequence, formally analysing the integrated model is a hard task also from a computational point of view.

It is worth underlining the effort spent in the construction and in the integration of the models. Generally speaking, the effort needed to conduct the modelling and V&V activities was much higher compared to the previsions at the beginning of the project.

6.2. Final remarks

The set of the lessons learnt during the development of WP2 can be grouped into three different categories.

Methodological: the main result of the WP2 is the definition of a **concrete methodology** able to guide the system engineers in both modelling and analysis of ETCS-L3 systems. The **agile-like process** can cope with uncertainty in both specification and in technology, resulting able to adapt to changes. The methodology relies on solid languages supported by assessed (freely available) tools (i.e., SysML/Papyrus, TA/UPPAAL, SANs/Möbius); hence, the approach can be instantiated "as-is" without entry barriers.

The methodology is furthermore enriched by the presence of a **compositional approach between formal models**. The compositional approach allows a model developer to merge formal models to support the construction of large models from smaller components.

The approach starts from the current status of the MBS requirements as they are specified by other Shift2Rail (S2R) projects.

The methodology is replicable and adaptable to other railway systems (e.g., interlocking, metro lines, subject to the proper changes related to the specificity of the new settings).

Technical: there are two remarkable points in this category.

An **extensible SysML model**, considering a ETCS-L3 system from different views: requirements, functional, behavioural, structural and data. The model clearly states a specification for a reference ETCS-L3 functional architecture that could be a **starting point for future development and standardisation activities**. The model is based on the Papyrus toolchain, that is free available on the Internet. The model itself is released with an **open-source** licence and can be accessed at https://github.com/stefanomarrone/performingrail.

The second interesting point is constituted by a library of formal models, expressed with a modular modelling approach. The models are reusable and customizable according to different market segments and parameter values of the signalling (e.g., timeouts), physical features of the trains (e.g., train length) and communication architecture (e.g., probability of loss of communication, communication delay).

System-related: the specification, modelling and analysis phases contribute to understanding some aspects of the MBSs.

The **ten OPSs** defined in the WP2 depict ten different ways to elicit knowledge on the ETCS-L3 systems using system testing techniques, as well as a guide to composing formal models. The OPSs have been assessed using an industrial survey.

Another kind of remarks is constituted by the analysis of the requirements and in findings that **some guidances still misses**. As an example, in [7] Subsection 6.3.4, a new requirement regarding the loss of the integrity of the train is proposed and the consistency with the existing requirements is analysed.

The analysis run on both TA and the SAN model highlights the **formal verification of the properties** chosen on three OPSs (see Subsection 5.1.2 and Subsection 5.1.3)) as well as

providing quantitative information on how much parameter variations affect normal train movement with undesired brakes (see Subsection 5.2).

In the end, the most valuable contribution is a general **improvement of the knowledge** on ETCS-L3 and a capability of all the specification, modelling and analysis activities to **reduce the level of the uncertainty** owned by a system whose standardization has not started — yet.

Future research efforts will be spent in extending and fine-tuning the developed models, optimizing the V&V process, and in bringing into play more powerful servers for the analysis.

Finally, it is worth mentioning that the results of this deliverable, the experience gained by formal modelling the ETCS-L3 functionalities and components will serve as valuable entries for task Task 1.3 – Recommendations and standardisation to definition of the Railway Minimum Operational Performance Standards for moving block systems (T1.3), where the lessons learnt in WP2 are summarized and reported in D1.2.

Bibliography

[1] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J.-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. Henzinger, and K. Larsen, "Contracts for systems design," INRIA, 2012.

[2] A. Cimatti and S. Tonetta, "Contracts-refinement proof system for component-based embedded systems," *Science of Computer Programming*, vol. 97, pp. 333–348, 2015.

[3] B. Meyer, "Design by contract," *IEEE Computer*, vol. 25(10), p. 40–51, 1992.

[4] N. A. Lynch, "Input/output automata: Basic, timed, hybrid, probabilistic, dynamic," in *In Roberto M. Amadio and Denis Lugiez, editors, CONCUR 2003*, vol. Lecture Notes in Computer Science, vol. 2761. Springer, 2003, p. 187–188.

[5] L. de Alfaro, T. A. Henzinger, and M. Stoelinga, "Timed interfaces," in *Proc. of the 2nd International Workshop on Embedded Software (EMSOFT'02)*, vol. Lecture Notes in Computer Science, vol. 2491. Springer, 2002, p. 108–122.

[6] C. Seceleanu, et al., "PERFORMINGRAIL D2.1 - Modelling guidelines and Moving Block Use Cases characterization," Tech. Rep., 2021. [Online]. Available: https://www.performingrail.com/

[7] S. Marrone, et al., "PERFORMINGRAIL D2.2 - Moving Block Specification Development," Tech. Rep., 2022. [Online]. Available: https://www.performingrail.com/

[8] M. Samra, et al., "PERFORMINGRAIL D1.1 - Baseline system specification and definition for Moving Block Systems," Tech. Rep., 2021. [Online]. Available: https://www.performingrail.com/

[9] UNISIG, "ERTMS/ETCS: Safety Requirements for the Technical Interoperability of ETCS in Levels 1 & 2 - SUBSET-091, issue 3.4.0," 2015.

[10] —, "X2Rail-3 Deliverable D4.2 - Moving Block Specifications — Part 2 – System Definition," Tech. Rep., 2018. [Online]. Available: https://projects.shift2rail.org/s2r_ip2_n.aspx?p=X2RAIL-3

[11] R. Milner, "Communication and concurrency," *PHI Series in Computer Science*, 1989.

[12] UNISIG, "ERTMS/ETCS: System Requirements Specification - SUBSET-026, issue 3.6.0," 2016.

[13] O. Himrane, J. Beugin, and M. Ghazel, "Toward formal safety and performance evaluation of gnss-based railway localisation function," *IFAC-PapersOnLine*, vol. 54, no. 2, pp. 159–166, 2021.

[14] D. Basile, M. ter Beek, A. Ferrari, and A. Legay, "Exploring the ERTMS/ETCS full moving block specification: an experience with formal methods," *International Journal on Software Tools for Technology Transfer*, vol. 24, pp. 351–370, 2022.

[15] J. Aoun, R. M. P. Goverde, R. Nardone, Q. Egidio, and V. Vittorini, "Towards a fault tree analysis of moving block and virtual coupling railway signalling systems," in *The 6th International Conference on System Reliability and Safety Venice, Italy, November 23-25, 2022*. IEEE, 2022.

[16] E. Quaglietta, et al., "MOVINGRAIL D4.2 - Cost-Effectiveness Analysis for Virtual Coupling," Tech. Rep., 2020. [Online]. Available: https://movingrail.eu/

[17] T. Courtney, S. Gaonkar, K. Keefe, E. W. Rozier, and W. H. Sanders, “Möbius 2.3: An extensible tool for dependability, security, and performance evaluation of large and complex system models,” in *2009 IEEE/IFIP International Conference on Dependable Systems & Networks*. IEEE, 2009, pp. 353–358.