International Journal on Software Tools for Technology Transfer
https://doi.org/10.1007/s10009-022-00653-3

GENERAL

Special Issue: FMICS 2019/2020

Exploring the ERTMS/ETCS full moving block specification: an experience with formal methods

Davide Basile[1] · Maurice H. ter Beek[1] · Alessio Ferrari[1] · Axel Legay[2]

Accepted: 8 March 2022

Abstract
Shift2Rail is a joint undertaking funded by the EU via its Horizon 2020 program and by main railway stakeholders. Several Shift2Rail projects aim to investigate the application of formal methods to new ERTMS/ETCS railway signalling systems that promise to move European railway forward by guaranteeing high capacity, low cost and improved reliability. We explore the ERTMS/ETCS level 3 full moving block specifications stemming from different Shift2Rail projects using UPPAAL and statistical model checking. The results range from novel rigorously formalised requirements to an operational model formally verified against scenarios with multiple trains on a single railway line. From the gained experience, we have distilled future research goals to improve the formal specification and verification of real-time systems, and we discuss some barriers concerning a possible uptake of formal methods and tools in the railway industry.

Keywords Formal methods · Railways · ERTMS/ETCS · Moving block · UPPAAL · Statistical model checking

1 Introduction

The railway sector is known for its robust safety requirements and the use of formal methods is indeed highly recommended in the relevant standards for the software requirements specification and software system design in order to be certified at the highest safety integrity levels. As a result, formal methods and tools have been successfully applied to railway systems [10,18,21,36–39,47,51,77,78]. However, the railway sector is notoriously cautious concerning the adoption of technological innovations, in particular if compared with other transport sectors. Hence, while satellite-based positioning systems have been in use for quite some time in the avionics and automotive sectors, modern railway signalling systems still prevalently use traditional ground-based train detection systems with fixed block distancing. However, the faster trains are allowed to run, the longer their braking distance, which results in an increased safety distance and a decreased line capacity. Therefore, to make the railway sector more competitive, robust and attractive, a recognised challenge constitutes the development and operation of moving block signalling systems that are as effective and precise as possible [50,66]. While this includes satellite-based positioning, its performance strongly depends on the signal propagation conditions. This calls for an integrated (multi-sensor) solution to cope with signal blockages (think, e.g., of tunnels) and degraded accuracy due to so-called multipaths or non-line-of-sight receptions, all of which typically affect satellite positioning in urban environments [16,35,55,73].

The work presented in this paper builds on previous European Rail Traffic Management System (ERTMS) models [9,11] developed in the context of the H2020 EU project ASTRail [1] (SAtellite-based Signalling and Automation SysTems on Railways along with Formal Method and Moving Block Validation) funded by the European Shift2Rail initia-

✉ Davide Basile
davide.basile@isti.cnr.it

Maurice H. ter Beek
maurice.terbeek@isti.cnr.it

Alessio Ferrari
alessio.ferrari@isti.cnr.it

Axel Legay
axel.legay@uclouvain.be

[1] ISTI–CNR, Via G. Moruzzi 1, 56124 Pisa, Italy

[2] Université Catholique de Louvain, Louvain-la-Neuve, Belgium

[1] http://www.astrail.eu

Published online: 10 April 2022

tive.[2] These previous models were based on a simplification of the moving block specification that considered only one train. They were developed using Simulink for requirements elicitation and consolidation, while UPPAAL was used for formal verification and sensitivity analysis.

In this paper, we discuss an extension and refinement of the aforementioned UPPAAL model. The model has been extended to consider more than a single train which concurrently communicate with the same (trackside) radio block centre (RBC). The safety-critical logic is updated accordingly. The movement authority that was previously considered constant, is now computed dynamically according to the traffic on the railway line, and in particular the tail of the train ahead. The physical behaviour of a train has been tuned and validated according to parameters available in the literature [62] about high-speed trains. An extended formal analysis has been carried out to formally verify the correctness of the function that computes the movement authority, which has been formalised in first-order logic. The concurrent nature of the radio block centre has led to the detection and mitigation of corner cases. We have carried out experiments to validate candidate parameter setups to reduce the risk of a train exceeding its movement authority. Finally, we have drawn conclusions acquired from the modelling and analysis experience, providing indications on how to further improve the existing state-of-the-art on formal verification of real-time models and its adoption by the railway industry.

Outline Section 2 contains an overview of related work, while some background information on statistical model checking and ERTMS level 3 moving block railway signalling systems is provided in Sect. 3. The core of the paper, presented in Sects. 4 and 5, concerns the formal specification and verification of an extended and refined UPPAAL model of the satellite-based ERTMS level 3 moving block signalling system from [11]. Section 6 contains some final considerations on the possible uptake of formal methods and tools, such as UPPAAL, in the railway industry, while Sect. 7 concludes the paper.

2 Related work

We are aware of numerous attempts at modelling and analysing ERTMS level 3 signalling systems. Most notably, ERTMS hybrid level 3 systems (using virtual fixed blocks) and its radio block centre component have recently been modelled and analysed with members of the B family (formal methods B, Event-B, iUML-B, and the tools ProB and Rodin), with the model checker SPIN and its specification language Promela, with the lightweight formal specification language Electrum that extends the Alloy language and tool, with the systems modelling language SysML, and with the mCRL2 specification language and toolset [1,4,8,26,30,58,65,69,79].

However, these approaches are less suitable to perform quantitative modelling and analysis, which is fundamental to demonstrating the reliability of the operational behaviour of satellite-based ERTMS level 3 moving block railway signalling system models.

One of the earliest quantitative evaluations of moving block railway signalling systems can be found in [60], where GSM-R communication between a train and a radio block centre is modelled in StoCharts, a conservative QoS-oriented extension of UML statechart diagrams, and analysed with the stochastic model checker ProVer. Only delays and communication loss are modelled in [60], whereas physical models of trains (e.g., braking), satellite positioning, and the logic of a radio block centre computing the space each train is allowed to travel are not taken into consideration.

We are also aware of attempts at modelling hybrid or stochastic models of ERTMS level 3 (moving block and virtual coupling) scenarios with Simulink, with UML, with the bounded model checker HySAT, with the probabilistic hybrid automata verifier ProHVer, with the symbolic model checker SMV, with timed Petri nets and the timed Petri net analyser Tina, and with Stochastic Activity Networks and the Möbius simulator [48,49,52,53,59], generally applying classical (i.e., not statistical) model checking.

We believe that together with some of the above-mentioned approaches, the work reported in this paper contributes to an increased understanding of satellite-based ERTMS level 3 moving block railway signalling systems. This resembles the view of *formal methods diversity* [67], according to which the application of diverse formal methods and tools on different variants of a case study design may increase confidence in the correctness of the analysis results.

3 Background

Uppaal UPPAAL SMC [27] is a variant of UPPAAL [15], which is a well-known toolbox for the verification of real-time systems. UPPAAL models are stochastic timed automata, in which non-determinism is replaced with probabilistic choices and time delays with probability distributions (uniform for bounded time and exponential for unbounded time). These automata may communicate via broadcast channels and shared variables.

Statistical Model Checking (SMC) [2,64] involves running a sufficient number of (probabilistic) simulations of a system model to obtain statistical evidence (with a predefined level of statistical confidence) of the quantitative properties to be checked. Monte Carlo estimation with Chernoff-

[2] http://www.shift2rail.org

Hoeffding bound executes $N = \lceil (ln(2) - ln(\alpha))/(2\epsilon^2) \rceil$ simulations $\rho_i, i \in 1...N$, to provide the interval $[p' - \epsilon, p' + \epsilon]$ with confidence $1 - \alpha$, where $p' = (\#\{\rho_i \mid \rho_i \models \varphi\})/N$, i.e., $\Pr(|p' - p| \leq \epsilon) \geq 1 - \alpha$ where p is the unknown value of φ being estimated statistically and ϵ and α are the user-defined precision and confidence, respectively [64]. SMC offers advantages over exhaustive (probabilistic) model checking [7,24]. Most importantly, it scales better, since there is no need to generate and possibly explore the full state space of the model under scrutiny, thus avoiding the combinatorial state-space explosion problem typical of model checking. Indeed, the parameter N is *independent* from the size of the state-space. Moreover, the required simulations can trivially be distributed and run in parallel. This comes at a price. Contrary to (probabilistic) model checking, exact results (with 100% confidence) are impossible to obtain. A further advantage is related to its possible uptake in industry. Compared to model checking, SMC is simple to implement, understand and use, and it requires no specific modelling effort other than an operational system model that can be simulated and checked against (state-based) quantitative properties. In fact, SMC is becoming more and more widely accepted in industry [5,13,14,22,46,54,56,61,71,72,76].

ERTMS Moving Block Railway Signalling ERTMS is an international standard aiming to enhance safety and efficiency and improve cross-border interoperability of trains in Europe by the replacement of national railway signalling systems with a single European standard for train control and command systems. ERTMS relies on the European Train Control System (ETCS), an Automatic Train Protection (ATP) system continuously supervising the train to ensure that safety speed and distances are not exceeded. The ERTMS/ETCS standard distinguishes four levels of operation, from level 0 to level 3 (L3), depending on the role of trackside equipment and on how the information is transmitted to/from trains. It is currently deployed on several lines throughout Europe at most in its level 2 (L2).

ERTMS L2 uses trackside equipment (viz., track circuits) to detect the occupancy of a section of a railway track by trains, determining the location of trains with a coarse granularity. This information is sent to a trackside unit, called Radio Block Centre (RBC), which sends a Movement Authority (MA) to each train. The MA is computed by summing the free track circuits ahead, meaning L2 is based on *fixed block signalling*. A block is a section of the track between two fixed points, which start and end at signals, with their lengths designed to allow trains to operate as frequently as necessary (i.e., ranging from many kilometres for secondary tracks to a few hundred metres for busy commuter lines). The block sizes are determined based on parameters like the line's speed limit, the train's speed, the train's braking characteristics, drivers' sighting and reaction times, etc. Since the railway sector's stringent safety requirements impose the length of fixed blocks to be based on the worst-case braking distance, regardless of the actual speed of the train, the faster trains are allowed to run, the longer the braking distance and the longer the blocks need to be, thus decreasing the railway track's capacity. The MA provides a train with the maximum distance it is allowed to travel, the maximum speed (depending on the track) it is allowed to travel at, and data about the track ahead (like temporary speed restrictions and (un)conditional emergency stops). The Onboard Unit (OBU) of each train uses the MA and data stored on board (e.g., the train's braking capability) to compute the braking curve or the dynamic speed profile that determine the speed limit, triggering an emergency brake whenever this limit is exceeded. In L2, so-called Eurobalise responders on the rails of a railway are used for exact train positioning, while the required signalling information is provided to the driver's display by continuous data transmission via GSM-R with the RBC. Further trackside equipment is needed for train integrity detection.

ERTMS L3 no longer relies on trackside equipment for train position detection and train integrity supervision. Instead, the OBU is responsible for monitoring the train's position and autonomously computing its current speed through its odometry system. To this aim, the OBU periodically sends the train's position to the RBC and the RBC, in turn, sends back an MA to each train. The MA is computed by exploiting knowledge of the position of the rear end of the train ahead, thus computing a safe zone around the moving train, meaning L3 is based on *moving block signalling*. The resulting moving block signalling systems considerably reduce the headways between subsequent trains, thus allowing trains in succession to close up, in principle to the braking distance. As a result, moving block signalling allows for more trains to run on existing railway tracks, in response to the ever-increasing need to boost the volume of passenger and freight rail transport and the cost and impracticability of constructing new tracks. For this to work, the precise absolute location, speed, and direction of each train needs to be known. These can be determined by a combination of sensors: active and passive markers along the track, as well as trainborne speedometers.

This envisioned future switch to next generation signalling systems would not only optimise the exploitation of railway tracks due to the adoption of moving block signalling, but the removal of trackside equipment would result in lower capital and maintenance costs. Actually, L3 does not explicitly refer to the moving block concept, but it admits any implementation able to periodically provide the RBC with the train positions and using limited trackside equipment. A few pilot implementations referred to as Hybrid L3 (cf. Sect. 2), use virtual fixed blocks: a line is logically divided into fixed length blocks and the OBU is in charge of communicating, at specific points of the line, the train's position, computed

using its onboard odometry system. Moving block signalling based on continuous communication and MA computation is currently implemented in some automatic metros, as part of CBTC (Communication-Based Train Control) systems.

Shift2Rail Shift2Rail stimulates developing safe and reliable technological advances that allow to complete the single European railway area with an ambitious aim: "double the capacity of the European rail system and increase its reliability and service quality by 50%, all while halving life-cycle costs." To this aim, it supports transitioning to next generation (L3) ERTMS/ETCS railway signalling, with a focus on satellite-based train positioning based on a Global Navigation Satellite System (GNSS), moving block distancing, and automatic driving. In [57], ASTRail partners confirmed how the use of additional sensors is recommended to improve the robustness of a GNSS-based positioning in rail scenarios. In ASTRail, satellite-based ERTMS L3 moving block railway signalling scenarios were mainly considered for two different purposes:

1. In a simplified format, for trial applications of formal modelling and analysis to assess the usability and applicability of formal methods and tools in the railway domain. This assessment is an important issue for the successful uptake of formal methods and tools in the railway industry [10]. In [9], we presented such a trial experience in modelling and (statistical) model checking a satellite-based moving block railway signalling scenario with UPPAAL. In [12], we modelled a moving block signalling system with autonomous driving and synthesised safe and optimal driving strategies by applying a combination of SMC and reinforcement learning with UPPAAL Stratego.
2. For modelling and validating more detailed models as major portions of an integrated system design of moving block signalling with automated driving technologies to provide rigorous and verified definitions of functional, interoperability, and dependability requirements. As part of the assessment, we conducted a survey with railway practitioners to identify the most mature (semi-)formal methods and tools to be used in the railway context [40]. As a result of this survey, a total of 14 tools were carefully reviewed by means of a systematic evaluation based on a set of 34 evaluation features, upon which eight tools were selected for the aforementioned trial application phase, in which we modelled principles of the moving block scenario in all eight tools. This comparison experiment is reported in [45]. Simulink and UPPAAL were among the eight selected tools. In particular, Simulink was considered appropriate for functional requirements elicitation, early prototyping and animation involving domain experts, while UPPAAL was considered the appropriate choice for formal verification of real-time systems.

In [11], we presented models of the above-mentioned detailed satellite-based ERTMS L3 moving block signalling system model in both Simulink and UPPAAL, with a specific focus on the second one. The Simulink model was obtained from a requirements elicitation and refinement activity performed with the industrial partners of ASTRail, carried out to consolidate an initial set of requirements for the moving block signalling system into an executable specification, after which we developed a corresponding UPPAAL model. Thus, these models are a formalisation of the initial set of requirements. We reported on this modelling experience and subsequent analyses with UPPAAL and drew some lessons. We chose to perform SMC with UPPAAL rather than simulation and analysis with Simulink, because we had all the monitoring infrastructure for temporal properties. Moreover, UPPAAL primitively supports real-time hybrid systems and has many graphical facilities that helped communicating with the stakeholders. In [11], we showed how UPPAAL can also assist in fine tuning communication parameters that are fundamental for the reliability of the model's operational behaviour. In particular, we validated that (i) the frequencies of the messages exchanged between the train and its trackside control system as well as (ii) the unit of distance that a train is allowed to proceed based on a MA can be set such that the probabilities of failures (like the train exceeding its MA, i.e., failing to brake in time if it lacks permission to proceed) are close to zero. While numerical constraints for (i) and (ii) were previously defined by railway experts, in ASTRail we wanted to explore to which extent UPPAAL and SMC can be exploited to validate such constraints and to support sensitivity analysis on the parameters. Moreover, the possibility of graphically depicting the state machine models and animating them during simulations has been of great help for communicating with the stakeholders concerning the modelled requirements.

Case Study The moving block railway signalling system considered in this paper uses a scenario from [75], where four different L3 moving block systems are proposed. These are combinations of moving or fixed virtual blocks and with or without trackside train detection. The specific scenario modelled in this paper is the full moving block without trackside detection, which is the base system type considered in [75], depicted in Fig. 1. In this scenario, the system can issue MAs based on the rear of the preceding train, and the so-called end of authority can therefore be an arbitrary location on the railway. Note that this is different from the L2 currently in operation where, e.g., a new end of authority occurs each 12 s for a block section of 1 km and a train running at 300 km/h [31]. The ERTMS hybrid L3 system specification mentioned in Sect. 2 is a possible implementation of the system type fixed virtual blocks with trackside train detection (cf. [75] for details).

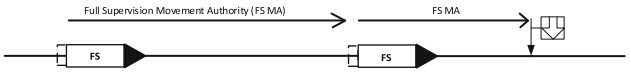

Fig. 1 Overview of full moving block system type without trackside train detection taken from [75]

The ERTMS/ETCS L3 moving block system architecture is composed of the OBU and an L3 trackside unit [75]. The OBU includes the Location Unit (LU), while the RBC is the only L3 trackside unit considered. The LU receives the train's location from GNSS satellites and other sensors' data which are aggregated. The location (and the train's integrity) are transmitted to the OBU, which, in turn, sends the location to the RBC. Upon receiving a train's location, the RBC sends an MA to the OBU (together with speed restrictions and route configurations), indicating the space the train can safely travel based on the safety distance with respect to preceding trains. The RBC computes the MA by communicating with all trains under its supervision to know their position (head and tail), neighbouring RBCs in case of handover [80], and by exploiting its knowledge of the positions of switches and routes of trains through communications with a Route Management System (RMS). In our scenario, we abstract from an RMS and from communications among neighbouring RBCs. Instead, we consider trains communicating within one RBC supervision area. The location and the MA are continuously updated and if the OBU does not continuously receive a fresh MA, the OBU is required to force the train to brake.

4 Formal model

In this section, we discuss the UPPAAL formalisation of the moving block system scenario. The various models are publicly available,[3] each one containing a different parameter setup and corresponding queries according to the experiments described in Sect. 5, to aid easier experiments reproducibility. The models and analyses have been developed using the latest UPPAAL distribution (4.1.24 academic, November 2019). Next, we outline the automata constituting the models in detail.

In [11], we described the original Simulink/Stateflow model, output of a requirement elicitation phase, and its mapping into a UPPAAL model.

In this paper, we discuss a refinement of the UPPAAL model from [11], where details of the guidelines used for translating the Simulink model in UPPAAL can be found. The models in [11] considered only one train and one RBC. The MA transmitted from the RBC to the train did not account for the possibility of having other trains on the track, thus the MA was fixed to a certain amount of meters. Here, we model the scenario with more trains on a line, and the MA of each train is based on the location of the train ahead (cf. Fig. 1). The model has also been simplified and three previous automata for generating location requests, for replying to the location requests, and for simulating the train behaviour have now been merged into a single basic model called location unit. The physical behaviour of the train has been redefined and validated based on target values retrieved from the literature.

The model consists of a number of automata composed as a synchronous product. Below, we list its constituting components, which are three main entities, namely the RBC, the LU, and the OBU, each represented by a different automaton. We do not account for artificial random failures but focus only on those failures that can be raised from the modelled logic.

All components listed next are *templates* in UPPAAL, which is a mechanism allowing to instantiate different instances of an automaton. This makes it possible to perform simulations and analyses with a certain number of RBCs, OBUs, and LUs; not fixed beforehand in the model. However, in line with the specification from our industrial partners and [75], we assume that each component communicates with other components of the same index. For instance, RBC_0 always communicates with OBC_0 and never with OBC_1, which communicates with RBC_1. As a consequence, an RBC has different threads, represented by different instantiations of the RBC template, and each RBC thread communicates with one assigned OBU. All these RBC threads interact by means of a shared memory, used for reading the locations of other trains and storing the location received by the assigned train.

Furthermore, this model is parametric and highly customisable. It is possible to analyse different operational scenarios of the moving block system by instantiating the individual parameters of the model. For instance, it is possible to customise the period of each of the various messages such as the period of requesting the location or the period of sending the MA (these values are parameters prefixed by `freq`). By changing these parameters, we can perform different evaluations of the properties of interest, as we will show in the next section, so as to fine tune the setup of these parameters.

We now describe the model's components. To avoid confusion between a location of a train and that of the automaton, we always refer to the automata locations as states. Thus, a state can be intended as either a location of the automaton or as the tuple composed of the location and values of variables and clocks, and contextual information is given to distinguish between the two.

Global Declarations We use the global declarations below to define constants of the whole system, channels and shared variables, as well as the parameter setup for the analysed scenario. In Sect. 5, we refer to the values of parameters below

[3] https://github.com/davidebasile/ASTRail/tree/master/STTT2021

as default values. The unit measures for time and space are always seconds and meters, respectively. The default values have been derived from earlier specifications. Concerning delays, we opted for a pessimistic approach to be able to observe potential requirements otherwise undetected.

```
//parameters of the model

//number of trains
const int nTrain=3;

//exponential rates of train and RBC
const double rateOBU=0.5;
const double rateRBC=0.5;

//rate by which a new location is required to LU (sec.)
const double freqLU = 0.5;

//timeout to reception of MA (sec.)
const int OBU_out_timer= 30;

//the period in which the OBU location is sent to the RBC (sec.)
const int freqOBU = 1;

//the period of sending MA before the ack is received (sec.)
const int freqRBC=1;

//attempts in sending MA before receiving ack
const int MaxAttemptsToSendMa=3;

//movement authority when no train is ahead (meters)
const int maIfNoTrainAhead = 6000;

//constants for identifying who caused the failure
const int SENDLOC=1;
const int RECEIVEMA=2;
int whofailed=0;

//channels

//LU --> OBU_SendLoc
broadcast chan LU_send_location[nTrain];

//OBU_Send_Loc --> RBC
broadcast chan OBU_send_location_to_RBC[nTrain];

//RBC --> OBU_ReceiveMA, OBU_SendLoc, LU
broadcast chan OBU_rec_MA[nTrain];

//OBU_ReceiveMA --> RBC
broadcast chan OBU_send_MA_ack[nTrain];

//OBU_SendLoc, OBU_ReceiveMA --> OBU_SendLoc, OBU_ReceiveMA, LU
broadcast chan OBU_fail[nTrain];

//buffers for value passing
double x=0.0;
int y=0;
double z=0.0;

//locations shared by RBC
double loc[nTrain];
```

We note that, to perform statistical model checking, it is necessary that all channels are broadcast, i.e., outputs are non-blocking and discarded if not received by anyone. In Sect. 5, we analyse possible message loss. We use an array of channels to have a channel for each train. The array `loc` is only used by the RBC automata, to share the locations received by trains.

Location Unit The template automaton for the LU is depicted in Fig. 2. The LU groups all aspects of the train related to the computation of the location. In this model, we do not focus on errors introduced due to satellite positioning, and the location is estimated using a simple physical model of the movement

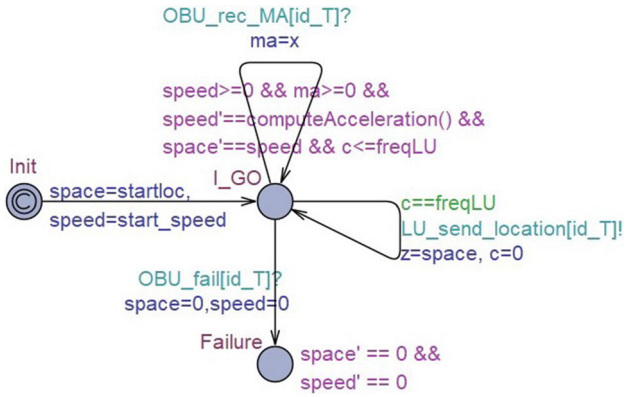

Fig. 2 Location Unit template

of the train, its speed, and the acceleration and deceleration that are triggered by approaching the limit described by the MA. In particular, the position of the train is stated in an uni-dimensional space and identified by one coordinate. This represents the route that the train is following.

This template considers five parameters of the train:

`const int id_T`, its unique identifier;
`const double startloc`, its starting location;
`const double stoploc`, its arrival location;
`const double start_speed`, its starting speed;
`const double average_speed`, its average speed.

From the initial state, depicted with two inner circles, the automaton starts by initialising the position (called space) and speed with the corresponding parameters values. The location is committed: the initialisation is the first operation performed by the simulation. Both `speed` and `space` are *hybrid* clocks, used to represent physical continuous entities. State `I_GO` represents the train travelling. The invariant requires that both speed and space are positive, and that the value of the clock `c` must be less than the parameter `freqLU`. Indeed, since this is a timed automaton, the value of the clock `c` increases uniformly during the permanence in that state. When the condition `c==freqLU` is satisfied, the train's location is transmitted to the OBU, with signal `LU_send_location[id_T]!` and buffer `z=space`; moreover, clock `c` is reset. We note that OBU and LU share the same identifier `id_T`, since they are part of the same train. From the same state, the automaton also receives messages from OBU about new MAs, via channel `OBU_rec_MA[id_T]?`. The value of the buffer is stored in the temporary variable `ma`. In case a failure is triggered by one of the OBU components, the signal `OBU_fail[id_T]?` is received, bringing the automaton to a `Failure` state where the train's location and speed are set to zero.

Physical Model Two ordinary differential equations are also present, used to represent the continuous evolution of physical quantities. In the `Failure` state, there is no evolution of location and speed, thus both derivatives are equal to zero. In state `I_GO`, as usual, space is the derivative of speed and speed is the derivative of acceleration. Concerning the representation of acceleration, we abstract away from a detailed physical model that transforms the actuators controlled by the computer into a specific acceleration or deceleration of the train. In [62], such a model is provided. Moreover, it is reported that it needs `100` s for a train to stop from a speed of `300` km/h, and that the braking distance is `5000` m. Figure 8 of [62] shows the curve of train braking speed and the curve of braking displacement. It takes `94.2` s from normal operation to stop, and the braking distance is `4647.16` m, which is reported to meet the national standard. The analysis in [60] confirms the headway of 1 min for trains travelling at 300 km/h with their given parameter setup, i.e., at a distance below 5000 m from the end of authority, the trains start to brake. Finally, [82] reports that a high speed train can accelerate from `0` to `300` km/h in `316` s.

Accordingly, we report the following declarations of the LU template and the function for computing the acceleration of the train, tuned with the above data (we omit declarations of temporary variables).

```
1   const double distance_to_brake = 5000.0; //m
2   clock speed= 2.0; // m/s
3   clock space=0.0;  //m
4
5   double computeAcceleration()
6   {
7       const int acceleration_constant = 95;
8       double headway = ma - space;
9       bool stop = space+0.0>=stoploc;
10
11      if ((speed+0.0)<=1.0) {
12            if ((headway<distance_to_brake)|| stop) return 0.0;
13            else  return 30;
14      }
15      else if ((headway<distance_to_brake)|| stop )
16          return -1.32/log(speed);
17      else return ((average_speed-speed)/acceleration_constant);
18  }
```

This function `computeAcceleration()` computes a train's acceleration and deceleration based on the current location, the current speed, the MA, and the arrival location of the train. The `acceleration_constant` is used to model the steepness of the speed curve: the lower the value, the more abrupt the acceleration. The if conditions in lines 11–12 are used for the case when the train is starting or arriving, based on its speed. Instead, the if condition in line 15 is used to decide whether to accelerate or to decelerate, based on the headway and the braking distance. Usually the headway is given in seconds [31] (e.g., 1 min headway at 300 km/h corresponds to roughly 5 km), whereas the variable in line 8 is the remaining space (in meters) to the end of the MA. To decelerate, the function `-1.32/log(speed)`

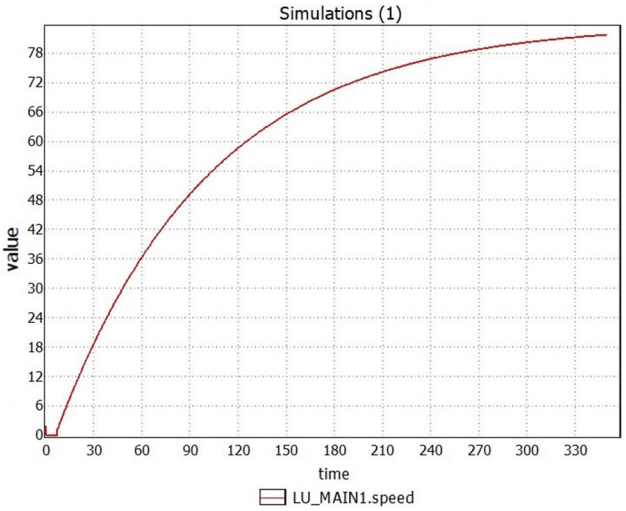

Fig. 3 Acceleration scenario: trajectory of speed

is used to model the braking curve, while the acceleration is modelled with the function:

```
((average_speed-speed)/acceleration_constant).
```

Note that while `computeAcceleration()` is deterministic, the acceleration and deceleration behaviour depends on the headway of the train. Due to the presence of stochastic delays in the communications of locations and MAs between the RBC and the trains, each train may accelerate and decelerate at different times with respect to other trains and in different simulations.

Validating the Physical Model Here, we report simulation data validating the model to respect the above data retrieved from the literature, with an average speed of `84` m per second (i.e., `302.4` km/h). These simulations validate the acceleration model to mimic complex physical models retrieved from the literature [62] for these specific target values of time, speed, and space.

The first is an acceleration scenario with a single train and the LU template instantiated as follows:

```
LU_MAIN1 = LU_MAIN_T(0,0.0,50000.0,0.0,84.0).
```

Figure 3 shows a simulation with the speed trajectory of the train. It needs around `300` s to reach a speed of `84` m per second.

Concerning braking, Figs. 4 and 5 show the space and speed for the following braking scenario:

```
LU_MAIN1 = LU_MAIN_T(0,0.0,0.0,84.0,84.0).
```

In this scenario, the train has a starting speed of `84` m per second and the starting location is equal to the arrival location, i.e., the train starts to decelerate from the first seconds

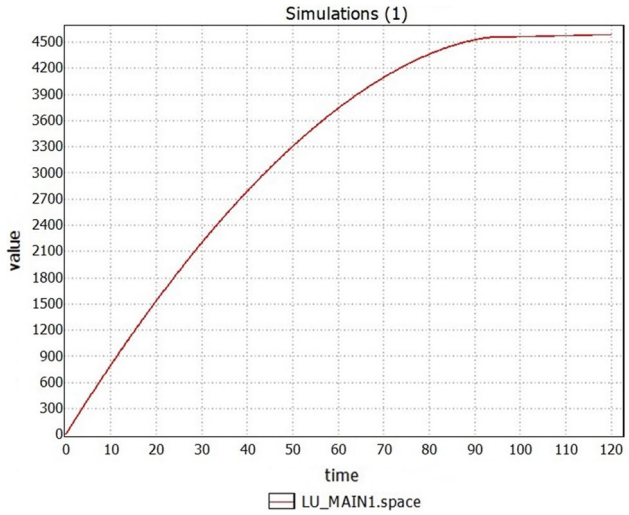

Fig. 4 Braking scenario: location trajectory (space)

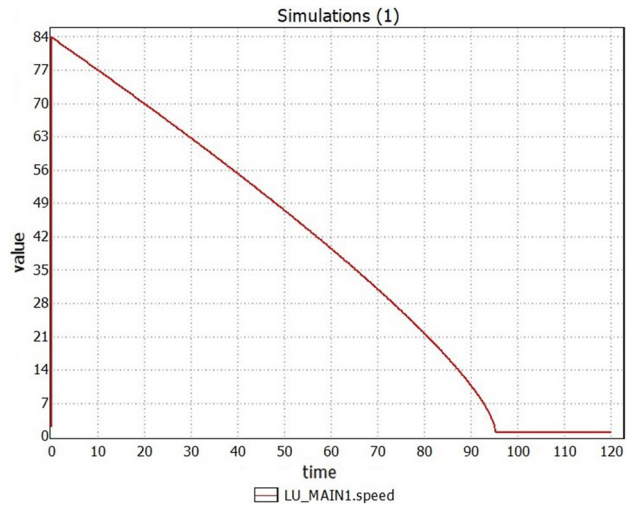

Fig. 5 Braking scenario: speed trajectory

of the simulation. Figures 4 and 5 show the space and speed trajectories: around `4600` m and `94` s are required for the train to stop. Finally, even though these figures only show a single simulation with a single speed profile, Figs. 11 and 12 display different simulations with different speed profiles (cf. Sect. 5).

We remark that the purpose of the analysis is to explore and validate early requirements for the novel ERTMS/ETCS L3 moving block system, and in particular of an RBC generating MAs according to the location of trains; hence, more concrete representations of physics are out of the scope of this paper.

OBU Send Location Depicted in Fig. 6, this template automaton is the main component of the OBU. It has only one parameter `const int id_T`, and it performs a variety of operations. The first operation is the reception of the position by the LU. Subsequently, with a certain periodicity and in the presence of a fresh location received by LU, this component sends the received position to the RBC. The same component moreover receives the MA from the RBC (after sending its position). Finally, it implements one of the safety mechanisms present in the system specification. In particular, at each instant of time, the model checks that the train's position has not exceeded the MA received from the RBC; if it has, it will enter a `Failure` state.

The automaton has four states. The nominal state `WatingLocFromLU` is the initial state. State `Failure` is entered when a failure is triggered by this automaton or by another OBU component. The remaining states are committed, and are used to link different transitions as one atomic operation with different ordered synchronisations.

State `WatingLocFromLU` has an invariant to guarantee that the MA is always positive and the received location is always within the MA. This guarantees that a failure may only be triggered when receiving either a fresh `ma` or location. The invariant condition is a disjunction contemplating the initialisation of operations, by means of a flag `boot`. The initialisation is finalised when the first MA is received.

In a potential order of execution, the first transition performed is the one with signal `LU_send_location [id_T]?`. It represents the reception of the position from the LU; `loc=z` represents the assignment of the variable `loc` that reads from the buffer variable `z` used to implement value passing. A flag `newloctosend` initialised to false is used to signal the presence of a new location to be transmitted. The buffer is immediately reset to reduce the state space and the flag is set to true. The intermediate committed location is used as a choice. If the invariant is violated by the new received value, a failure is triggered with the signal `OBU_fail[id_T]!`. At any instant a failure may be triggered by another OBU component, so the committed location has also an outgoing transition `OBU_fail[id_T]?`. In case of failure, values are reset, and a variable `whofailed` is used for logging who triggered the failure. In case the new value that is received is valid, the main state is reached again.

The transition with guard `c>=freqOBU && new loctosend=true` is activated when the guard is satisfied, i.e., when the clock reaches and exceeds the `freqOBU` parameter and a fresh location is ready to be transmitted to the RBC. Once enabled, the time to fire this transition is sampled from an exponential distribution with rate `rateOBU`, used to model potential delays in communication between trains and the RBC. This transition implements a periodic operation which is carried out with periodicity `freqOBU`. The action is that of sending the position data to the RBC. The sending operation is transmitted via the signal `OBU_send_location_to_RBC[id_T]!`, while the assignment of variables is `z = loc, c=0.0, lastlocsent=loc, newloctosend=false`, i.e., the value `loc` of the location is stored in the buffer variable,

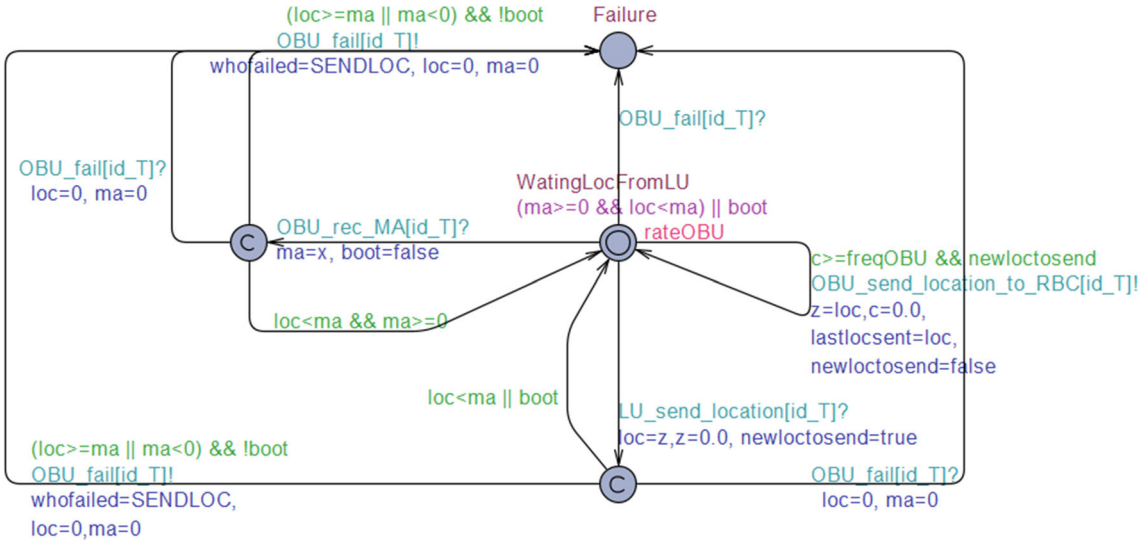

Fig. 6 OBU send location template

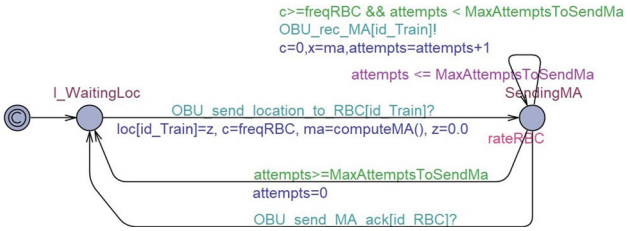

Fig. 7 RBC template (before formal analysis)

the clock `c` is reset, the variable `lastlocsent` is used to log the last known location sent to the RBC, and the flag `newloctosend` is set to false. Note that the flag not only prevents the sending of a location that is not fresh, but it also prevents the corner case in which `freqOBU>freqLU` and the first location send to the RBC is undefined.

The transition with the synchronisation `OBU_rec_MA[id_T]?` models the reception from the RBC of the MA, which is stored in the variable `ma`. Similar to the reception of a location, the reception of a new MA leads to a choice, where it is tested whether or not the new value is satisfying the invariant and due operations take place. As stated above, the first reception of a MA causes the assignment `boot=false` to terminate the initialisation procedure. Note that the interactions with the RBC are happening at the index of the channel `id_T`, which corresponds to the train's identifier.

RBC This template automaton, depicted in Fig. 7, is a model of one RBC thread. It has two parameters: `const int id_RBC` is the identifier (id) of the RBC and `const int id_Train` that of its associated train. It has three states: `I_WaitingLoc`, `SendingMA`, plus an initial committed state. From `I_WaitingLoc`, the RBC is ready to receive a location, with synchronisation `OBU_send_location_to_RBC[id_Train]?` and the actions `loc[id_Train]=z`, `c=freqRBC`, `ma=computeMA()`, and `z=0.0`. The location value in buffer `z` is stored in the shared array `loc[]`, at the position that is the train's id. The buffer is instantaneously reset to reduce the state-space. The clock `c` is initialised to `freqRBC`, so that the first operation in the next state is that of sending the MA. The MA, stored in variable `ma`, is computed with the function `computeMA()` described next.

```
1 double computeMA(){
2    int i; double val=0.0;
3    for(i=0;i<nTrain;i++){
4       if (i!=id_Train && loc[i]>loc[id_Train]
5           && (val==0||loc[i]<val))
6         val=loc[i];
7    }
8    if (val>0.0) return val;
9    else return loc[id_Train]+maIfNoTrainAhead;
10 }
```

This function basically checks the minimum location that is greater than the location of the associated train (lines 2–7). If such a location exists, then this becomes the MA, which is exactly the location of the train ahead of the train with id `id_Train` (line 8). Otherwise, if no train is ahead, a fixed constant value `maIfNoTrainAhead` is used to extend the MA, similar to [11]. In state `SendingMA`, the RBC attempts to send the MA to the train for a number of times equal to the constant `MaxAttemptsToSendMa`. Each attempt is performed periodically, with periodicity `freqRBC`. Since this interaction is with a remote train computer, the delay is sampled exponentially with rate `rateRBC`. The sending operation is done with synchronisation `OBU_rec_MA[id_Train]!` and the value stored in buffer `x=ma`. The clock is reset and the counter of attempts is incremented. When the number of attempts is exceeded, they are reset (`attempts=0`) and state `I_WaitingLoc`

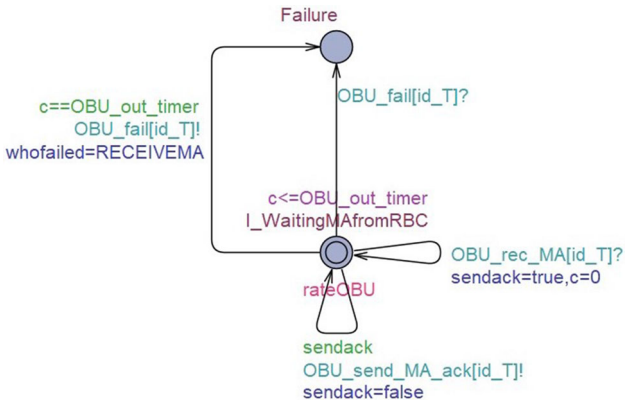

Fig. 8 OBU receive MA template

is reached. Similarly, if an ack is received from the train concerning reception of the MA, with `OBU_send_MA_ack[id_RBC]?`, state `I_WaitingLoc` is reached. The location has invariant `attempts <= MaxAttemptsToSendMA` to guarantee that no more than the declared number of attempts is performed.

OBU Receive MA The last template automaton, depicted in Fig. 8, models the logic of the OBU. It has one parameter: `const int id_T`. It consists of one state `I_WaitingMAfromRBC` and a `Failure` state. It receives an MA from the RBC and returns a corresponding acknowledgement message, via the channels `OBU_rec_MA[id_T]?` and `OBU_send_MA_ack[id_T]!`, respectively. The parameter `rateOBU` is used to model delays in communication with the RBC. A Boolean flag `sendack` is used to enable the sending of the ack once the MA is received. This flag allows to model communication delays in between receiving the MA and sending the ack. This component implements an additional safety mechanism of the system specification by means of an invarant `c<=OBU_out_timer`. The clock `c` is reset each time an MA is received. In case the clock reaches the threshold, a failure is triggered. Indeed, the MA must be received within a given amount of time.

5 Formal analysis

In this section, we discuss the analysis of the model by means of SMC. We only use probability estimation, and the parameters are set to 0.05 for both probability of false negatives (α) and probability uncertainty (ϵ or u). The probabilistic deviation is 0.01 (δ). We emphasise that SMC is particularly useful for a quick, non-costly, and preliminary model analysis. This is possible with due setup of the above statistical parameters, to avoid a huge number of simulations. A full and costly state-space exploration can be performed at a later stage. We remark that different data configurations and different instantiations of the templates can be performed. Indeed, in this section we discuss various experiments conducted with different parameter setups and different numbers of trains.

We anticipate that the model in Fig. 7 contains issues that were detected during the formal analysis phase. We opted to report the direct experience of modelling and analysis with UPPAAL rather than reporting only the mitigated model, i.e., we report how some problematic issues were detected and subsequently mitigated. For a matter of space and presentation, we do not report all issues that were found and mitigated in detail. In particular, the use of the Boolean flags `boot` and `newloctosend` was not originally planned during the requirements elicitation and modelling phase. Their use was introduced after model checking detected that failures could happen due to undefined initial values of location (before it is actually computed by LU) and MA (before it is actually computed by RBC). The mitigated model is reported at the end of this section. The default parameters of the model are set to those described in Sect. 4 under global declarations. If not stated otherwise, the average speed is 84 m/s.

Location Freshness We start with a scenario with only one train and therefore one RBC thread. We start with the following requirement:

```
Pr[<=1000]
    (<> OBU_MAIN_SendLocationToRBC1.lastlocsent!=loc[0]
     && OBU_MAIN_SendLocationToRBC1.lastlocsent!=zero) ≈ 0
```

where `zero` is a constant with value zero and 1000 is the time bound. The right part of the conjunction is used to avoid the dummy initial state in which no location has been sent yet. The left part of the conjunction checks whether the last location sent by the OBU is different from the last one stored by the RBC. Basically, the probability that the location of the RBC is not fresh is measured. The requirement asks whether this measure is close to zero. In only 29 runs, the formula is evaluated to lie in the interval [0.901855, 1] with confidence 0.95.

Indeed, the RBC model in Fig. 7 violates such a requirement. This is because in state `SendingMA` the RBC is ignoring new locations that could arrive from the train in case of delays in sending the MA. To meet the requirement, the RBC model is updated. The transition for receiving the location from the train is duplicated in a new transition whose source and target are both state `SendingMA`. Since there are only two states, no location message from OBU will be lost.

Message Loss The previous analysis has showed that, due to the usage of broadcast channels, potential loss of important messages may be undetected. We use the so-called *demonic*

completion for input-enabledness to overcome this issue. This consists in adding a transition that leads to a sink state for each non-expected input in each state of each automaton, as opposed to *angelic completion*, according to which unspecified inputs are discarded (modelled by adding self-loops) [28]. We do not perform any completion of failure states, as the analysis always terminates if a failure is triggered. What happens after a failure is triggered is not significant in this model. Similarly, we do not complete any initial committed state: all these states will be exited before any interaction occurs. By statically analysing the model, we note that both LU and OBU Receive MA have only two inputs, namely `OBU_rec_MA[id_T]?` and `OBU_fail[id_T]?`, which are already enabled in their only significant state. Hence no completion is necessary for them. On the converse, RBC has an input `OBU_send_MA_ack[id_RBC]?` not enabled in state `I_WaitingLoc`. This message could potentially be lost. Similarly, OBU Send Location has two committed states where two inputs are not enabled, namely `LU_send_location[id_T]?` and `OBU_rec_MA[id_T]?`. Thus, a total of four transitions are added to OBU Send Location.

The next step is to check if there exists a path where one of the sink states is reached. We actually measure the probability that this happens. We perform the analysis on the scenario with three trains, assuming to find values close to zero. These trains have their starting positions duly spaced, but the same terminating position and speed. We start by analysing the sink states of the OBU, with the following formula:

```
Pr[<=1000] (<> OBU_MAIN_SendLocationToRBC1.sink
            || OBU_MAIN_SendLocationToRBC2.sink
            || OBU_MAIN_SendLocationToRBC3.sink)
```

In 29 runs, as expected, the formula is evaluated to be in the interval [0, 0.0981446] with confidence 0.95. Thus, the probability of losing those messages is indeed close to zero. Next, we analyse sink states of the RBC:

```
Pr[<=1000]
  (<> RBC_MAIN1.sink || RBC_MAIN2.sink || RBC_MAIN3.sink)
```

Surprisingly, using 145 simulations, the formula is evaluated to be in the interval [0.851567, 0.951396] with confidence 0.95. The reason this was not expected is that `MaxAttemptsToSendMA=3` and the rates of message delays are set to `0.5`. We thus analysed the trace provided by UPPAAL, using the message sequence chart automatically generated by the tool. We noted that the trace is a sequence of ≈ 800 transitions, not easily reproducible using manual simulation driven by a human user. A relevant portion of this chart is displayed in Fig. 9.

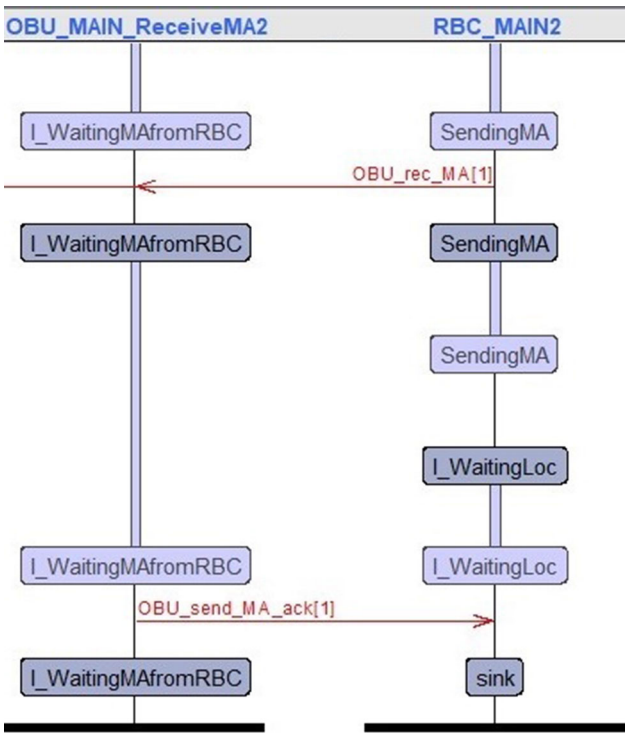

Fig. 9 The message sequence chart generated by UPPAAL revealing the issue in the RBC

By analysing the terminal part of the trace, an incorrect interaction of the RBC is detected: after only one attempt of sending an MA in state `SendingMA`, the state `WaitingMA` is reached before receiving any ack from the OBU. This is only possible if the number of attempts has reached the threshold `MaxAttemptsToSendMA`, which is equal to 3. However, only one attempt was performed in the analysed trace, and this should not be possible. This trace has detected a bug that was present in the RBC: the counter of attempts is only reset (i.e., `attempts=0`) when such threshold is reached, and never when an ack is received from the OBU. Basically, after a series of correct interactions between the OBU and the RBC, the threshold is reached and all further ack messages are ignored by the RBC.

We fixed this issue by adding the effect `attempts=0` in the transition with synchronisation `OBU_send_MA_ack[id_RBC]?`. The bug was introduced by the modeller in an attempt to reduce the state space. Indeed, in the model in [11] this counter was reset when entering state `SendingMA`, rather than when leaving it. Resetting the counter `attempts` when leaving state `SendingMA`reduces the state space. Indeed, with this solution in state `I_WaitingLoc`, the value of `attempts` will always be zero. However, only one transition is entering state `SendingMA`, whereas two are leaving it. Thus, in [11] the counter was reset in only one transition. With the new fix,

the counter reset should occur in two transitions rather than one.

After amending the model, the probability of the RBC entering a sink state is close to zero. Note that with a different parameter setup, i.e., by reducing `MaxAttemptsToSendMA` and incrementing the message delays, it is possible that the RBC receives the OBU ack in state `I_WaitingLoc`. This is however not harmful and compliant with the RBC specification. Indeed, the fact that the RBC may ignore missing acks and change state is a requirement of the original semi-formal model in [11]. Finally, we remark that the demonic completion has only been used for this specific analysis and it is not present in the model.

Verifying Movement Authorities One of the novel contributions of our model is the RBC logic, whose computation of MAs is based on the locations of other trains. Although the computation with `computeMA()` may look simple, manually reviewing the code of it is not enough to ascertain correctness of this functionality. We will show that subtle issues may arise only upon analysing the system as a whole, i.e., not evident in case of an isolated MA computation. The requirement that the RBC must satisfy is expressed using first-order logic:

```
loc[id_Train]<=ma && forall(id:int[0,nTrain-1])
    ( loc[id]<=loc[id_Train] || (loc[id]>=ma
        && exists(id1:int[0,nTrain-1]) ma==loc[id1]) )
```

The formula is rendered as a state invariant of the RBC model. This invariant is assigned to state `SendingMA` of the RBC template. Indeed, this is the only state where the MA is computed, while in state `WaitingLoc` the RBC is waiting for a new location.

Recall that `id_Train` is the identifier of the train associated to the RBC. The above formula states that whenever an RBC is in state `SendingMA`, trivially, the MA must be ahead of the train, and for all trains, either their location is smaller than that of `id_Train`, i.e., they are behind, or one such a train has a location greater than the computed MA. In this last case, where at least one train is ahead of `id_Train`, there must exist one train whose location is equal to the `ma`. In other words, the formula states that either the assigned train is ahead of all other trains, or if this is not the case, then there is no other train whose location is between `id_Train` and its MA, and the MA is equal to one such a train ahead.

After testing the model against the invariant using the scenario with three trains, UPPAAL finds that the invariant is violated in some trace. However, after carefully reviewing the function `computeMA()`, whose returned values are stored in variable `ma`, no issues were found. Indeed, as previously anticipated, we found out that the violation was not due to an error in computing the MA. Rather it is due to the concurrent nature of the RBC, where different threads share the same memory. The invariant is satisfied only when state `SendingMA` is entered, but after some execution steps, the other RBC threads receive new locations and modify the shared array of locations `loc`. As a result, the MA computed and stored in variable `ma` becomes outdated and the invariant is violated. This is possible in case of delays in communicating the MA to the OBC. Note that an outdated MA may lead to issues such as emergency braking, which can be avoided if the MA is guaranteed to be freshly computed before sending it to the OBU.

To mitigate this issue, the RBC automaton is further modified such that no temporary variable `ma` is used, and each occurrence of `ma` (including also occurrences in the invariant) is substituted with a call to the function `computeMA()`. The mitigated model is reported to never violate the state invariant in all subsequent analyses that were performed. The stable version of the RBC model that was obtained is reported in Fig. 10.

Probability of Exceeding Movement Authority We conclude the formal analysis of the model by analysing the probability of failure due to a train exceeding its MA. Note that in [11], only one train was present on a railway line. Here, we analyse several trains, and since the MA can be computed as the tail of the train ahead, this failure may result in a collision between

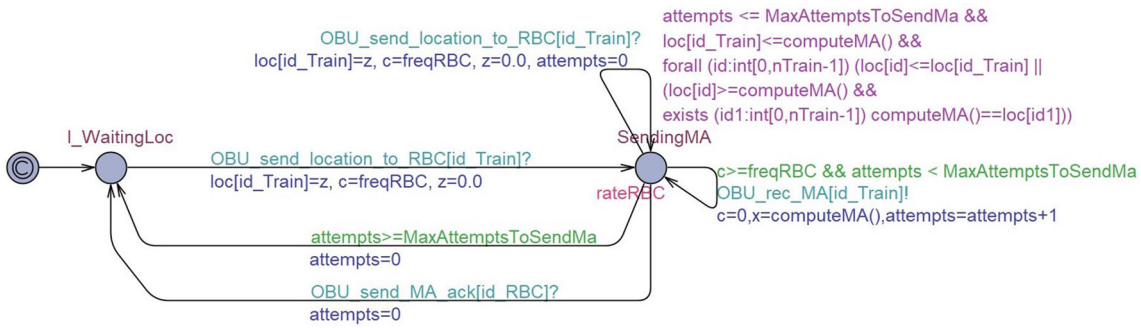

Fig. 10 The amended RBC template automaton

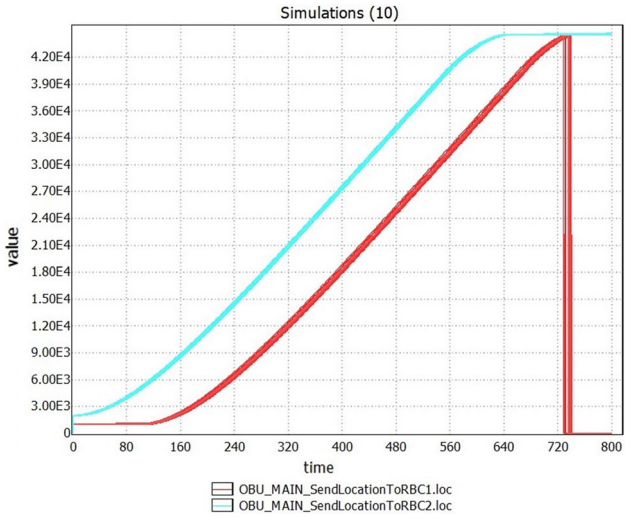

Fig. 11 A scenario in which two trains crash

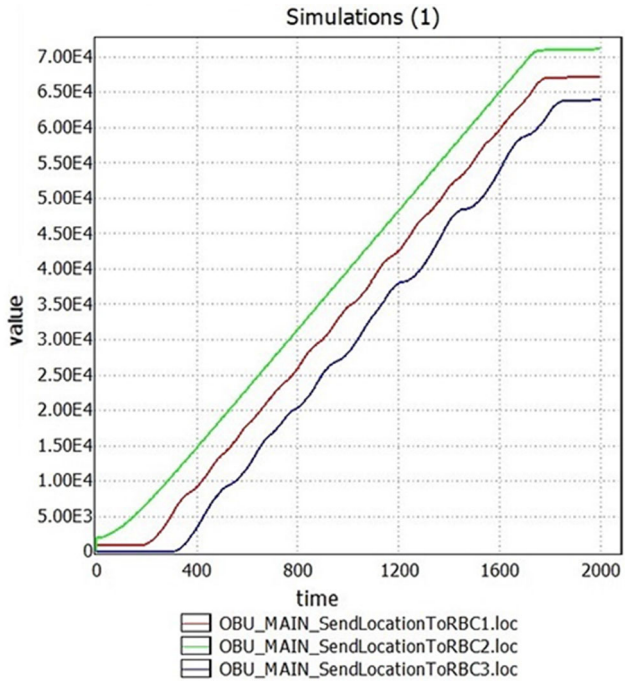

Fig. 12 A scenario with a slower leading train

two trains. The formula to be analysed is as follows:

```
Pr[<=1000] (<> whofailed==SENDLOC)
```

The requirement is as usual to evaluate the query with a value close to zero. As before, we first try a scenario with degraded values and two trains. To do so, the two LUs are set as follows:

```
LU_MAIN1 = LU_MAIN_T(0,1000.0,80000.0,0.0,84.0);
LU_MAIN2 = LU_MAIN_T(1,2000.0,40000.0,0.0,84.0).
```

Basically, the first train starts at position 1000, followed by a second train that starts at position 2000. However, the second train stops at position 40000, whereas the first train should proceed its trip to its stop position at 80,000 m along its route. To observe the crash, we change the braking distance parameter of the LU from `5000` to `4000`. All other parameters have their default values. As expected, after 29 runs, the query is evaluated to be in the interval [0.901855, 1] with confidence 0.95.

Figure 11 reports a plot with 10 simulations running for 800 s, plotting the location of the two trains as seen by the OBU. In all simulations, we observe that the train behind has not enough space to brake and crashes against the train ahead. This is visible because the train detecting the failure resets its location. Of course, right after the crash, the data of the simulations are no longer significant.

In the next scenario, we simulate three trains with the default values for all parameters, including braking distance set to `5000`. The three LUs are set as follows:

```
LU_MAIN1 = LU_MAIN_T(0,1000.0,80000.0,0.0,84.0);
LU_MAIN2 = LU_MAIN_T(1,2000.0,70000.0,0.0,42.0);
LU_MAIN3 = LU_MAIN_T(2,0.0,80000.0,0.0,84.0).
```

In this particular scenario, the leading train is the one with index 2, because its starting position is at 2000 m. The other two trains are behind, with the train with index 1 in the middle. Moreover, the train with index 2 ahead has half the average speed of the trains behind and its arrival position is at an earlier position than the other trains. The requirement is satisfied, and the query is evaluated, in 29 runs, to be in the interval [0, 0.0981446] with confidence 0.95. One simulation is depicted in Fig. 12 to show the locations of the three trains. The analysis confirms that the default parameters are safe: the trains behind both succeed in lowering their speed when approaching the braking distance and to stop in time to avoid crashing against the train ahead, even if their mission is not yet completed (i.e., reaching the arrival position).

We conclude this section with a summary of the analyses that have been assessed and the takeaway messages of the results that have been identified for each experiment. It is reported in Table 1. We refer to the respective paragraphs earlier in this section for the formulae associated with the properties verified and their intuitive meaning.

6 Considerations

In this section, we draw some considerations based on our experience in modelling and analysing the specification of the next generation ERTMS L3 moving block system reported in this paper, as well as our experience gained in the Shift2Rail

Table 1 Summary of the analyses carried out, the results obtained, and the lessons learned

Analysis	Results	Lessons learned
Location freshness	The RBC shall always be ready to receive new locations from the OBU	A communication mismatch due to modelling errors between the OBU and the RBC has been identified in an earlier model, by comparing the last location received by the RBC with the last location sent by the OBU
Message loss	In the model, all unexpected messages received from the RBC, LU, and OBU can safely be discarded	An error introduced by refactoring the model has been identified, by analysing all possible message exchanges
Verifying MA	At each attempt, the RBC shall compute the MA the moment it is sent to the OBU	Formalising the requirements of multiple trains has led to detect flaws in communication of the MA, due to the interplay between delays in communications and concurrent RBC threads
Exceeding MA	Under the given assumptions, the model confirms that trains can travel with a headway of 1 min	A model has been identified with a parameter setup that has confirmed the values in the literature about headway in high-speed trains

projects ASTRail and 4SECURail.[4] We shed some light on the interplay of semi-formal and formal notation, and provide some feedback on future improvements of real-time formal modelling tools such as UPPAAL.

It is important to note that the aforementioned specification is currently under investigation and official requirements do not yet exist. The model presented in this paper is based on two earlier UPPAAL models, presented in [9,11]. Both models were in turn based on two semi-formal specifications. The first was a Simulink/ StateFlow model, used for requirements elicitation with railway stakeholders, discussed in [11]. The second was a more simple Real-Time UML model directly provided by railway stakeholders, discussed in [9].

The mapping from these two semi-formal models to UPPAAL models was done manually by the first author. These semi-formal models were used as informal specifications, with no assumptions on their precise semantics. Indeed, for an automatic translation, a standard and official formal semantics should be available for both the source and the target models. This is required to formally prove their behavioural equivalence and thus the correctness of the translation. However, semi-formal modelling languages are not equipped with formal semantics. Any attempt in filling this gap suffers either from semantic aspects intentionally left open by the standards or from unintentional ambiguities [20,25,29]. For instance, the literature contains several attempts to provide a formal semantics for a fragment of Simulink/Stateflow [3,17,23,68,74], together with mappings to timed automata [46], with discording results.

In this paper, SMC was used as a means to analyse and discover potential corner cases, which has led to an accurate, more refined specification with respect to the earlier models. Of course, the usage of a rigorous notation equipped with well-defined formal semantics comes with further benefits. Indeed, requirements expressed using either natural language descriptions or semi-formal modelling languages may suffer from ambiguities in the description of the system and the semantics of the used semi-formal language. Our formal specification is free of such ambiguities. The functional requirements could be unambiguously described in a state machine formalism, thus overcoming the ambiguities of natural language requirements. Of course, the converse is also possible, and from the state machine it is possible to derive structured natural language requirements. Moreover, since it is an executable formal model, it allowed us to automatically detect and mitigate corner cases that were overlooked in earlier specifications. Thus, the bugs in the model are related and traceable to missing or ambiguous requirements. If, by contradiction, the interface between the train (i.e., the OBU) and the RBC for exchanging locations and MA messages were not part of the requirements (but instead referred to as implementation details), then different implementations from different vendors of the OBU and RBC sub-systems could not be interoperable with each other, due to these aspects left unspecified in the requirements.

Level of Abstraction It is important to underline that the results presented in this paper are strongly based on the assumptions present in the model and on the adopted abstraction level. A matter of future research is the investigation of guidelines to distinguish correct from too coarse abstractions for a given formal analysis to be carried out.

For example, while concrete parameters (e.g., speed and braking distance) cannot be abstracted away when evaluating certain (quantitative) properties (e.g., probability of exceeding the MA), these can be abstracted away for other (qualitative) properties. As another example, the violation of the state invariant of the RBC concerning the MA as described in Sect. 5 is observed if delays in communica-

[4] https://www.4securail.eu/

tions between the RBC and the OBU have been modelled, but it is independent from the specific values of `rateRBC`, `freqRBC`, and `maxAttemptsToSendMA`.

Validating the level of abstraction of a formal specification to be the appropriate one for the analysis to be carried out would provide further confidence in the results. The degree of such abstracted away aspects could be considered a measure of the level of abstraction of the specification. However, such measures can only be estimated by an a posteriori analysis, i.e., when the implementation has been produced and it becomes possible to measure how much details were abstracted away.

Ideally, the target aspects of the analysis (i.e., the requirements) should not be abstracted away, whereas, to be as general as possible, all other unnecessary details should be abstracted from. Attempts to provide guidelines for designing formal specifications are present in the literature [81]. They are inspired by design patterns as used in software engineering, but the correct level of abstraction is not addressed. Agile development of formal specifications with tight interactions with software developers by means of user interaction, as proposed in [70], could also be a solution to address the right level of abstraction.

Note that we only target abstractions used in formal specification models (i.e., abstracting away details not automatically realisable). Formal specification models are not to be confused with other Model-Based Systems Engineering (MBSE) models, where the abstraction is on the underlying execution support (e.g., models that are automatically compiled into implementations and which are used to graphically depict the source code).

Since the ERTMS/ETCS L3 moving block requirements are currently under investigation, validating whether the level of abstraction is adequate is difficult. Our analysis has however been useful in identifying corner cases not previously evident, formalising the identified requirements, and verifying them in the model.

We provide a final consideration on a possible future improvement of formal tools, in the direction of MBSE. Even though formal tools account for state-of-the-art formal verification not currently available in mainstream MBSE tools (e.g., temporal logic properties evaluated on simulation statistics), we believe that the relation between the formal specification and its implementation should be improved. This would help in identifying what would be the aforementioned more appropriate level of abstraction.

Separation of Concerns Hybrid stochastic formalisms (as used in UPPAAL) allow to model both physics (e.g., using ordinary differential equations) and discrete logic in one single automaton. While this can be of help to the modeller, there is no neat separation of concerns between the model of the external physical environment and the model of the software. If the modelled algorithm is using, e.g., predictive physical models to perform choices (e.g., the braking force to apply), distinguishing them from external physical models of the environment becomes even more cumbersome. Ideally, the software implementation should only be derived from the software models.

We argue that this separation could be improved. For example, the LU model in this paper describes a software component that is reading location values from sensors, sending them to the OBU with a certain periodicity, and receiving the MA. The LU also models the physics of the train, i.e., how actuators affect position, speed, acceleration and deceleration. Accurate modelling of physics and of software are two very distinct problems. Ideally, software modellers should not focus on accurate physical modelling. External support should be made available to formal tools for such aspects.

Constraining a modeller to clearly separate the software specification from the physics would generally result in a clearer design and a clearer relation with the implementation. For example, MBSE tools using, e.g., SysML, allow to express hierarchical blocks to divide the physical model from the software model, and indicate their interfaces.

Deployment Another important and somewhat related aspect in the design and verification of real-time distributed systems is the deployment of logical units to physical units. While it is reasonable to assume that communications occurring among units executing on the same computer can be modelled without stochastic delays (e.g., with a certain fixed periodicity), wireless communications among remote computers should account for possible stochastic delays in communication. A desirable feature in formal tools for real-time systems, such as UPPAAL, would be the declaration of deployment of such units. This would enforce a correct design, where entities deployed in remote units can only communicate by means of stochastic delayed channels. Moreover, this would help the modeller to avoid potential errors in modelling real-time systems due to, e.g., instantaneous communications among remote objects. In MBSE, deployment models are used to map logical units to physical units.

For instance, in our model we assume that the LU and the OBU are two logical units residing in the same physical board, whereas the RBC is a separate, remote computer. Accordingly, there are no stochastic delays between the OBU and the LU, whereas all communications between a train and the RBC are affected by stochastic delays. Since a new location is read every 0.5 s, the requirement that the location must be fresh and not older than one second is satisfied by construction. In earlier models, delays were not modelled also in communications among remote units. We argue that not modelling delays led to a model that was too coarse for the analysis to be performed.

Finally, our model assumes that no messages are lost; they can only be delayed. As such, the model is relying on an underlying transportation layer ensuring that all messages are correctly delivered. Such a transportation layer is abstracted away in the model. Another useful feature to have would be primitive support to enforce stochastic delayed lossy channels.

We conclude this section by framing the modelling and analysis carried out in this paper inside the Railway Development Process, specifically considering the tool UPPAAL.

Uppaal in the Railway Development Process One of the main challenges of introducing formal methods in railways is understanding how to integrate tools and practices in the existing process [42,43], and how to adapt the overall workflow to accommodate the innovation. The railway process is standardised according to the CENELEC norms [19,32–34], and when introducing a new technique, one should consider at least the following main factors:

1. In which *phase* of the process is the technique applied?
2. What is the level of *qualification* that is expected from the tools that support the specific technique?
3. How are the produced *artefacts* affected by the introduction of the technique?
4. *Who* is expected to apply the technique, who is expected to use its results, and how?

In the case of our work, we did not perform a structured case study within a company, but rather performed a set of experiments to identify the potential benefits of using SMC and UPPAAL. In this setting, the above-mentioned issues cannot be addressed in full, but we can at least provide some reflections based on our experience.

1. **Phase** The CENELEC norms indicate different development phases, structured according to the well-known V-process, namely Concept, Requirements, Architecture and Design, Detailed Design, Coding, Testing, Integration, Validation, and Maintenance. Formal methods are generically recommended, or highly recommended, in most of the phases, without providing any specific guidance on their usage.
   Quoting [19], "the use of a model without a textual requirement is tantamount to discovering our requirements without having expressed them".
   Considering our experience, we argue that UPPAAL can be exploited in the Requirements phase, for the identification and consolidation of both qualitative and quantitative requirements to be used. Section 5 reports candidate functional requirements identified through our model. UPPAAL has been proved useful in formalising such requirements in rigorous mathematical notation and in constructing a formal model, and has been an aid for exploring novel requirements of the envisioned system. SMC has been useful to perform quick verifications to provide early feedback to engineers for requirements discovering, leaving a full state-space generation to a later phase when the requirements and the models have been fully established. Moreover, starting from input parameters (e.g., communication delays), which are assumptions to be validated in a separate process, the values resulting from the simulations and the verification could indeed be included in the non-functional system requirements. This would lead to a radical decrease in terms of costs for a company, since these parameters, although strongly dependent on physical aspects, can be fine tuned on a common laptop, thus reducing the need for time-consuming field experiments. Following the analysis with UPPAAL, experiments with real trains or hardware-based simulators can be dedicated to confirmation of the results produced by the tool.
2. **Tool Qualification** Tool qualification, as common also in the avionic sector [63], consists of producing tests and documents that ensure that a certain tool was developed following the high-quality standards required by the railway process for safety-critical systems [6,41]. The highest standard is of course required for those tools that manipulate code, such as, e.g., compilers (normally referred as T3 tools) and that can therefore affect the safety of the final product by introducing errors, while the T2 category is for tools where a fault could lead to an error in the results of the verification or validation, and the T1 category is reserved for tools which have no impact on the verification nor on the final executable file. Intuitively, UPPAAL can be regarded as a T1 or T2 tool for the Requirements phase. In this paper, we addressed UPPAAL as an aid for exploring novel requirements. In this case, requirements must pass through the same verification and validation process used also without the aid of UPPAAL. As such, no specific certification is required for the tool. However, if UPPAAL is meant to be introduced in the current industrial process as T2 tool, evidence shall be provided by the vendors that the results produced by the tool are actually reliable, and that the tool has followed a documented process of development and maintenance. To our knowledge, this is currently lacking for UPPAAL, and this could seriously hamper its adoption as T2 tool. As reported in [19], the question of qualification of formal methods tools is "absolutely crucial". This is in contrast with a recent survey on formal methods among high-profile academic experts [51], reporting that academia "should not spend time on polishing the things that matter for acceptance in industry, such as user interfaces, have round-the-clock available help desks, liability, etc.", thus "leaving the development of professional tools to industry" because "there is no business case for long

term support of (academic) tools; industry needs stability and performance, academics need to innovate". Hence, it seems more likely that this technology transfer will eventually be completed by industry rather than by academia, of course exploiting decades of public research and collaborations with academia.

3. **Artefacts Implications** When using UPPAAL in the Requirements phase, specific artefacts are created that need to be reported and documented as part of the process. In particular, models and properties, together with an accurate explanation of the results, should be provided in a standardised format that can be reviewed by an assessor and submitted in the form of a paper-like document, to validate the process and certify the product. Producing and maintaining these documents, whenever changes are applied to the model, can be extremely time consuming. Therefore, means are required to support the automatic generation of documentation that can be easily inspected and understood also by subjects who are not necessarily experts in formal modelling and verification, like railway system assessors. This is however a limited process accommodation required if compared to the ones that have been documented in previous work [41,42], where the introduction of modelling and code generation led to a complete refactoring of the development process. Therefore, we argue that the process impact for UPPAAL, when used in the Requirements phase, can be considered limited.
4. **Involved Roles** The models presented in this paper were developed by a single subject, viz., the first author, who had previous expertise with UPPAAL. The use of graphical diagrams makes the language of the tool rather understandable by a subject with a background in computer science or engineering [44,45]. On the other hand, the creation of such models, the definition of properties, and the use of UPPAAL in general may require more advanced formal methods and logic skills. We recommend, also considering the limited uptake of formal languages by railway practitioners [10,44,45], to have a formal methods expert, or an external consultant, develop the models, while railway system experts interact with them to provide the required background, e.g., in terms of standard parameter values, or existing physical models. Of course, this has process implications in itself—roles need to be specified, and input-output artefacts shall be defined—that further increase the level of complexity of introducing the tool in an established workflow.

7 Conclusion

We have presented an UPPAAL model of the ERTMS L3 moving block specification currently under investigation in several European projects within the H2020 Shift2Rail initiative. The model is a refined version of earlier models. The formal analysis has helped to shed light on novel requirements, necessary to deal with corner cases such as start-up operations, message loss, and concurrency issues. In fact, we have formalised new requirements that a movement authority computed by a radio block centre should satisfy. These requirements have been expressed using rigorous, mathematical notations. The formal model has moreover been formally verified to satisfy several requirements against various scenarios with multiple trains on a single railway line, with a given confidence level computed by means of statistical model checking. From the gained experience, we have distilled future research lines to improve the formal specification and verification of real-time systems and discussed some barriers concerning a possible uptake in the railway industry.

The moving block specification considered in this paper could be extended in various directions: modelling the explicit handover between radio block centres, considering positioning errors, differentiating between freight trains and passenger trains, to mention a few. It would also be of interest to provide a prototypical implementation of the current specification, and formally establish their relation.

CRediT author statement D. Basile (first author): conceptualisation,methodology, validation, formal analysis, investigation, writing - original draft, visualisation. M.H. ter Beek: investigation, writing - original draft, visualisation, supervision, project administration. A. Ferrari: writing - review & editing, visualisation. A. Legay: project administration.

Acknowledgements Research partially funded by ASTRail and 4SECURail. The ASTRail and 4SECURail projects received funding from the Shift2Rail Joint Undertaking under the European Union's Horizon 2020 research and innovation programme under grant agreements 777561 (ASTRail) and 881775 (4SECURail) in the context of the open call S2R-OC-IP2-01-2019, part of the "Annual Work Plan and Budget 2019", of the programme H2020-S2RJU-2019. The content of this paper reflects only the authors' view and the Shift2Rail Joint Undertaking is not responsible for any use that may be made of the included information. We also acknowledge funding from the Italian MIUR PRIN 2017FTXR7S project IT MaTTerS (Methods and Tools for Trustworthy Smart Systems).

References

1. Abrial, J.: The ABZ-2018 case study with Event-B. Int. J. Softw. Tools Technol. Transf. **22**(3), 257–264 (2020). https://doi.org/10.1007/s10009-019-00525-3
2. Agha, G., Palmskog, K.: A survey of statistical model checking. ACM Trans. Model. Comput. Simul. **28**(1), 6:1-6:39 (2018). https://doi.org/10.1145/3158668
3. Agrawal, A., Simon, G., Karsai, G.: Semantic Translation of Simulink/Stateflow Models to Hybrid Automata Using Graph Transformations. In: Proceedings of the 4th International Workshop on Graph Transformation and Visual Modeling Techniques (GT-VMT 2004), *ENTCS*, vol. 109, pp. 43–56 (2004). https://doi.org/10.1016/j.entcs.2004.02.055
4. Arcaini, P., Kofroň, J., Ježek, P.: Validation of the hybrid ERTMS/ETCS level 3 using SPIN. Int. J. Softw. Tools Technol. Transf. **22**(3), 265–279 (2020). https://doi.org/10.1007/s10009-019-00539-x
5. Arnold, A., Baleani, M., Ferrari, A., Marazza, M., Senni, V., Legay, A., Quilbeuf, J., Etzien, C.: An Application of SMC to continuous validation of heterogeneous systems. EAI Endorsed Trans. Ind. Netw. Intell. Syst. (2017). https://doi.org/10.4108/eai.1-2-2017.152154
6. Ayoub, A., Kim, B., Lee, I., Sokolsky, O.: A systematic approach to justifying sufficient confidence in software safety arguments. In: Tonetta, S., Schoitsch, E., Bitsch, F. (eds) Proceedings of the 31st International Conference on Computer Safety, Reliability, and Security (SAFECOMP 2012), LNCS, vol. 7612, pp. 305–316. Springer. https://doi.org/10.1007/978-3-642-33678-2_26 (2012)
7. Baier, C., Katoen, J.P.: Principles of Model Checking. The MIT Press (2008). http://mitpress.mit.edu/books/principles-model-checking
8. Bartholomeus, M., Luttik, B., Willemse, T.: Modeling and analysing ERTMS hybrid level 3 with the mCRL2 toolset. In: Howar, F., Barnat, J. (eds) Proceedings of the 23rd International Conference on Formal Methods for Industrial Critical Systems (FMICS 2018), LNCS, vol. 11119. Springer. https://doi.org/10.1007/978-3-030-00244-2_7 (2018)
9. Basile, D., ter Beek, M.H., Ciancia, V.: Statistical model checking of a moving block railway signalling scenario with Uppaal SMC. In: Margaria, T., Steffen, B. (eds) Proceedings of the 8th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation: Verification (ISoLA 2018), LNCS, vol. 11245, pp. 372–391. Springer. https://doi.org/10.1007/978-3-030-03421-4_24 (2018)
10. Basile, D., ter Beek, M.H., Fantechi, A., Gnesi, S., Mazzanti, F., Piattino, A., Trentini, D., Ferrari, A.: On the industrial uptake of formal methods in the railway domain. In: Furia, C.A., Winter, K. (eds) Proceedings of the 14th International Conference on Integrated Formal Methods (iFM 2018), LNCS, vol. 11023, pp. 20–29. Springer. https://doi.org/10.1007/978-3-319-98938-9_2 (2018)
11. Basile, D., ter Beek, M.H., Ferrari, A., Legay, A.: Modelling and analysing ERTMS L3 moving block railway signalling with Simulink and Uppaal SMC. In: Larsen, K.G., Willemse, T. (eds) Proceedings of the 24th International Conference on Formal Methods for Industrial Critical Systems (FMICS 2019), LNCS, vol. 11687. Springer. https://doi.org/10.1007/978-3-030-27008-7_1 (2019)
12. Basile, D., ter Beek, M.H., Legay, A.: Strategy synthesis for autonomous driving in a moving block railway system with Uppaal Stratego. In: Gotsman, A., Sokolova, A. (eds) Proceedings of the 40th IFIP WG 6.1 International Conference on Formal Techniques for Distributed Objects, Components, and Systems (FORTE 2020), Held as Part of the 15th International Federated Conference on Distributed Computing Techniques (DisCoTec 2020), LNCS, vol. 12136, pp. 3–21. Springer. https://doi.org/10.1007/978-3-030-50086-3_1 (2020)
13. Basile, D., Fantechi, A., Rosadi, I.: Formal analysis of the UNISIG safety application intermediate sub-layer: applying formal methods to railway standard interfaces. In: Lluch-Lafuente, A., Mavridou, A. (eds.) Proceedings of the 26th International Conference on Formal Methods for Industrial Critical Systems (FMICS 2021), LNCS, vol. 12863, pp. 174–190. Springer. https://doi.org/10.1007/978-3-030-85248-1_11 (2021)
14. Basile, D., Fantechi, A., Rucher, L., Mandò, G.: Analysing an autonomous tramway positioning system with the UPPAAL statistical model checker. Form. Asp. Comp. (2021). https://doi.org/10.1007/s00165-021-00556-1
15. Behrmann, G., David, A., Larsen, K.G., Håkansson, J., Pettersson, P., Yi, W., Hendriks, M.: UPPAAL 4.0. In: Proceedings of the 3rd International Conference on the Quantitative Evaluation of SysTems (QEST 2006), pp. 125–126. IEEE. https://doi.org/10.1109/QEST.2006.59 (2006)
16. Beugin, J., Marais, J.: Simulation-based evaluation of dependability and safety properties of satellite technologies for railway localization. Transp. Res. C-Emerg. **22**, 42–57 (2012). https://doi.org/10.1016/j.trc.2011.12.002
17. Bouissou, O., Chapoutot, A.: An operational semantics for Simulink's simulation engine. ACM SIGPLAN Not. **47**(5), 129–138 (2012). https://doi.org/10.1145/2345141.2248437
18. Boulanger, J.L. (ed.): Formal Methods Applied to Industrial Complex Systems: Implementation of the B Method. Wiley, New York (2014). https://doi.org/10.1002/9781119002727
19. Boulanger, J.L.: Tool Qualification. In: CENELEC 50128 and IEC 62279 Standards, chap. 9, pp. 287–308. Wiley, New York. https://doi.org/10.1002/9781119005056.ch9 (2015)
20. Broy, M., Cengarle, M.V.: UML formal semantics: lessons learned. Softw. Syst. Model. **10**(4), 441–446 (2011). https://doi.org/10.1007/s10270-011-0207-y
21. Butler, M., Hoang, T.S., Raschke, A., Reichl, K.: Introduction to the special section on the ABZ 2018 case study: hybrid ERTMS/ETCS level 3. Int. J. Softw. Tools Technol. Transf. **22**(3), 249–255 (2020). https://doi.org/10.1007/s10009-020-00562-3
22. Cappart, Q., Limbrée, C., Schaus, P., Quilbeuf, J., Traonouez, L., Legay, A.: Verification of interlocking systems using statistical model checking. In: Proceedings of the 18th International Symposium on High Assurance Systems Engineering (HASE 2017), pp. 61–68. IEEE. https://doi.org/10.1109/HASE.2017.10 (2017)
23. Caspi, P., Curic, A., Maignan, A., Sofronis, C., Tripakis, S.: Translating discrete-time Simulink to Lustre. In: Alur, R., Lee, I. (eds.) Proceedings of the 3rd International Conference on Embedded Software (EMSOFT 2003), LNCS, vol. 2855, pp. 84–99. Springer. https://doi.org/10.1007/978-3-540-45212-6_7 (2003)
24. Clarke, E.M., Henzinger, T.A., Veith, H., Bloem, R.: Handbook of Model Checking. Springer, Berlin (2018). https://doi.org/10.1007/978-3-319-10575-8
25. Cook, S.: Looking back at UML. Softw. Syst. Model. **11**(4), 471–480 (2012). https://doi.org/10.1007/s10270-012-0256-x
26. Cunha, A., Macedo, N.: Validating the hybrid ERTMS/ETCS level 3 concept with Electrum. Int. J. Softw. Tools Technol. Transf. **22**(3), 281–296 (2020). https://doi.org/10.1007/s10009-019-00540-4
27. David, A., Larsen, K.G., Legay, A., Mikučionis, M., Poulsen, D.B.: Uppaal SMC tutorial. Int. J. Softw. Tools Technol. Transf. **17**(4), 397–415 (2015). https://doi.org/10.1007/s10009-014-0361-y
28. De Nicola, R., Segala, R.: A process algebraic view of input/output automata. Theor. Comput. Sci. **138**(2), 391–423 (1995). https://doi.org/10.1016/0304-3975(95)92307-J
29. Derezińska, A., Szczykulski, M.: Interpretation problems in code generation from UML state machines: a comparative study. In: Kwater, T., Zuberek, W.M., Ciarkowski, A., Kruk, M., Pękala, R., Twaróg, B. (eds) Proceedings of the 2nd Scientific Conference on

Computing in Science and Technology (STI 2011), Monographs in Applied Informatics, pp. 36–50. Warsaw University of Life Sciences (2012)
30. Dghaym, D., Dalvandi, M., Poppleton, M., Snook, C.: Formalising the hybrid ERTMS level 3 specification in iUML-B and Event-B. Int. J. Softw. Tools Technol. Transf. **22**(3), 297–313 (2020). https://doi.org/10.1007/s10009-019-00548-w
31. Émery, D.: Headways on high speed lines. In: Proceedings of the 9th World Congress on Railway Research (WCRR 2011), pp. 1–9. http://www.railway-research.org/IMG/pdf/f2_emery_daniel.pdf (2011)
32. European Committee for Electrotechnical Standardization: CENELEC EN 50128—Railway applications—Communication, signalling and processing systems—Software for railway control and protection systems (2011)
33. European Committee for Electrotechnical Standardization: CENELEC EN 50126-1—Railway applications–The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS)—Part 1: Generic RAMS process (2017)
34. European Committee for Electrotechnical Standardization: CENELEC EN 50126-2—Railway applications—The specification and demonstration of Reliability, Availability, Maintainability and Safety (RAMS)—Part 2: Systems approach to safety (2017)
35. Falco, G., Nicola, M., Falletti, E.: An HW-in-the-loop approach for the assessment of GNSS local channel effects in the railway environment. In: Proceedings of the 31st International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2018), pp. 3463–3477. Institute of Navigation. https://doi.org/10.33012/2018.15866 (2018)
36. Fantechi, A.: Twenty-five years of formal methods and railways: what next? In: Counsell, S., Núñez, M. (eds) Software Engineering and Formal Methods—Revised Selected Papers of the SEFM 2013 Collocated Workshops: BEAT2, WS-FMDS, FM-RAIL-Bok, MoKMaSD, and OpenCert, LNCS, vol. 8368, pp. 167–183. Springer. https://doi.org/10.1007/978-3-319-05032-4_13 (2013)
37. Fantechi, A., Ferrari, A., Gnesi, S.: Formal methods and safety certification: challenges in the railways domain. In: Margaria, T., Steffen, B. (eds) Proceedings of the 7th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications (ISoLA 2016), LNCS, vol. 9953, pp. 261–265. Springer. https://doi.org/10.1007/978-3-319-47169-3_18 (2016)
38. Fantechi, A., Fokkink, W., Morzenti, A.: Some trends in formal methods applications to railway signaling. In: Gnesi, S., Margaria, T. (eds) Formal Methods for Industrial Critical Systems: A Survey of Applications, chap. 4, pp. 61–84. Wiley. https://doi.org/10.1002/9781118459898.ch4 (2013)
39. Ferrari, A., ter Beek, M.H.: Formal methods in railways: a systematic mapping study. ACM Comput. Surv. (2022). https://doi.org/10.1145/3520480
40. Ferrari, A., ter Beek, M.H., Mazzanti, F., Basile, D., Fantechi, A., Gnesi, S., Piattino, A., Trentini, D.: Survey on formal methods and tools in railways: the ASTRail approach. In: Collart-Dutilleul, S., Lecomte, T., Romanovsky A. (eds) Proceedings of the 3rd International Conference on Reliability, Safety, and Security of Railway Systems: Modelling, Analysis, Verification, and Certification (RSSRail 2019), LNCS, vol. 11495, pp. 226–241. Springer. https://doi.org/10.1007/978-3-030-18744-6_15 (2019)
41. Ferrari, A., Fantechi, A., Gnesi, S.: Lessons learnt from the adoption of formal model-based development. In: Goodloe, A.E., Person, S. (eds) Proceedings of the 4th International NASA Formal Methods Symposium (NFM 2012), LNCS, vol. 7226, pp. 24–38. Springer. https://doi.org/10.1007/978-3-642-28891-3_5 (2012)
42. Ferrari, A., Fantechi, A., Gnesi, S., Magnani, G.: Model-based development and formal methods in the railway industry. IEEE Softw. **30**(3), 28–34 (2013). https://doi.org/10.1109/MS.2013.44
43. Ferrari, A., Fantechi, A., Magnani, G., Grasso, D., Tempestini, M.: The Metrô Rio case study. Sci. Comput. Program. **78**(7), 828–842 (2013). https://doi.org/10.1016/j.scico.2012.04.003
44. Ferrari, A., Mazzanti, F., Basile, D., ter Beek, M.H.: Systematic evaluation and usability analysis of formal tools for railway system design. IEEE Trans. Softw. Eng. (2021). https://doi.org/10.1109/TSE.2021.3124677
45. Ferrari, A., Mazzanti, F., Basile, D., ter Beek, M.H., Fantechi, A.: Comparing formal tools for system design: a judgment study. In: Proceedings of the 42nd International Conference on Software Engineering (ICSE 2020), pp. 62–74. ACM. https://doi.org/10.1145/3377811.3380373 (2020)
46. Filipovikj, P., Mahmud, N., Marinescu, R., Seceleanu, C., Ljungkrantz, O., Lönn, H.: Simulink to UPPAAL statistical model checker: analyzing automotive industrial systems. In: Fitzgerald, J., Heitmeyer, C., Gnesi, S., Philippou, A. (eds) Proceedings of the 21st International Symposium on Formal Methods (FM 2016), LNCS, vol. 9995, pp. 748–756. Springer. https://doi.org/10.1007/978-3-319-48989-6_46 (2016)
47. Flammini, F. (ed): Railway Safety, Reliability, and Security: Technologies and Systems Engineering. IGI Global. https://doi.org/10.4018/978-1-4666-1643-1 (2012)
48. Flammini, F., Marrone, S., Nardone, R., Vittorini, V.: Compositional modeling of railway virtual coupling with stochastic activity networks. Form. Asp. Comp. (2021). https://doi.org/10.1007/s00165-021-00560-5
49. Fränzle, M., Hahn, E.M., Hermanns, H., Wolovick, N., Zhang, L.: Measurability and safety verification for stochastic hybrid systems. In: Proceedings of the 14th International Conference on Hybrid Systems: Computation and Control (HSCC 2011), pp. 43–52. ACM. https://doi.org/10.1145/1967701.1967710 (2011)
50. Furness, N., van Houten, H., Arenas, L., Bartholomeus, M.: ERTMS level 3: the game-changer. IRSE News **232**, 2–9 (2017)
51. Garavel, H., ter Beek, M.H., van de Pol, J.: The 2020 expert survey on formal methods. In: ter Beek, M.H., Ničković, D. (eds) Proceedings of the 25th International Conference on Formal Methods for Industrial Critical Systems (FMICS 2020), LNCS, vol. 12327, pp. 3–69. Springer. https://doi.org/10.1007/978-3-030-58298-2_1 (2020)
52. Ghazel, M.: Formalizing a subset of ERTMS/ETCS specifications for verification purposes. Transp. Res. C-Emerg. **42**, 60–75 (2014). https://doi.org/10.1016/j.trc.2014.02.002
53. Ghazel, M.: A control scheme for automatic level crossings under the ERTMS/ ETCS level 2/3 operation. IEEE Trans. Intell. Transp. Syst. **18**, 2667–2680 (2017). https://doi.org/10.1109/TITS.2017.2657695
54. Gilmore, S., Tribastone, M., Vandin, A.: An analysis pathway for the quantitative evaluation of public transport systems. In: Albert, E., Sekerinski, E. (eds) Proceedings of the 11th International Conference on Integrated Formal Methods (iFM 2014), LNCS, vol. 8739, pp. 71–86. Springer. https://doi.org/10.1007/978-3-319-10181-1_5 (2014)
55. Groves, P., Jiang, Z., Rudi, M., Strode, P.: A portfolio approach to NLOS and multipath mitigation in dense urban areas. In: Proceedings of the 26th International Technical Meeting of the Satellite Division of The Institute of Navigation (ION GNSS+ 2013), pp. 3231–3247. Institute of Navigation (2013)
56. Gu, R., Enoiu, E., Seceleanu, C., Lundqvist, K.: Probabilistic mission planning and analysis for multi-agent systems. In: Margaria, T., Steffen, B. (eds) Proceedings of the 9th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles (ISoLA 2020), LNCS, vol. 12476, pp. 350–367. Springer. https://doi.org/10.1007/978-3-030-61362-4_20 (2020)
57. Han, X., Kazim, S.A., Tmazirte, N.A., Marais, J., Lu, D.: GNSS/IMU tightly coupled scheme with weighting and FDE for

rail applications. In: Proceedings of the 2020 International Technical Meeting of The Institute of Navigation (ION ITM 2020), pp. 570–583. Institute of Navigation. https://doi.org/10.33012/2020.17162 (2020)
58. Hansen, D., Leuschel, M., Körner, P., Krings, S., Naulin, T., Nayeri, N., Schneider, D., Skowron, F.: Validation and real-life demonstration of ETCS hybrid level 3 principles using a formal B model. Int. J. Softw. Tools Technol. Transf. **22**(3), 315–332 (2020). https://doi.org/10.1007/s10009-020-00551-6
59. Herde, C., Eggers, A., Fränzle, M., Teige, T.: Analysis of hybrid systems using HySAT. In: Proceedings of the 3rd International Conference on Systems (ICONS 2008), pp. 196–201. IEEE. https://doi.org/10.1109/ICONS.2008.17 (2008)
60. Jansen, D.N., Hermanns, H.: Dependability checking with StoCharts: is train radio reliable enough for trains? In: Proceedings of the 1st International Conference on Quantitative Evaluation of Systems (QEST 2004), pp. 250–259. IEEE. https://doi.org/10.1109/QEST.2004.1348039 (2004)
61. Jensen, P.G., Jørgensen, K.Y., Larsen, K.G., Mikucionis, M., Muñiz, M., Poulsen, D.B.: Fluid model-checking in UPPAAL for Covid-19. In: Margaria, T., Steffen, B. (eds) Proceedings of the 9th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation: Verification Principles (ISoLA 2020), LNCS, vol. 12476, pp. 385–403. Springer. https://doi.org/10.1007/978-3-030-61362-4_22 (2020)
62. Jin, Y., Xie, G., Chen, P., Hei, X., Ji, W., Zhao, J.: High-speed train emergency brake modeling and online identification of time-varying parameters. Math. Probl. Eng. **2020** (2020). https://doi.org/10.1155/2020/3872852
63. Krauss, S.S., Rejzek, M., Hilbes, C.: Tool qualification considerations for tools supporting STPA. In: Proceedings of the 3rd European STAMP Workshop (ESW 2015), Procedia Engineering, vol. 128, pp. 15–24. https://doi.org/10.1016/j.proeng.2015.11.500 (2015)
64. Legay, A., Lukina, A., Traonouez, L., Yang, J., Smolka, S.A., Grosu, R.: Statistical model checking. In: Steffen, B., Woeginger, G.J. (eds) Computing and Software Science: State of the Art and Perspectives, LNCS, vol. 10000, pp. 478–504. Springer. https://doi.org/10.1007/978-3-319-91908-9_23 (2019)
65. Mammar, A., Frappier, M., Tueno Fotso, S.J., Laleau, R.: A formal refinement-based analysis of the hybrid ERTMS/ETCS level 3 standard. Int. J. Softw. Tools Technol. Transf. **22**(3), 333–347 (2020). https://doi.org/10.1007/s10009-019-00543-1
66. Marais, J., Beugin, J., Berbineau, M.: A survey of GNSS-based research and developments for the European railway signaling. IEEE Trans. Intell. Transp. Syst. **18**(10), 2602–2618 (2017). https://doi.org/10.1109/TITS.2017.2658179
67. Mazzanti, F., Ferrari, A., Spagnolo, G.O.: Towards formal methods diversity in railways: an experience report with seven frameworks. Int. J. Softw. Tools Technol. Transf. (2018). https://doi.org/10.1007/s10009-018-0488-3
68. Minopoli, S., Frehse, G.: SL2SX Translator: from Simulink to SpaceEx models. In: Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control (HSCC 2016), pp. 93–98. ACM. https://doi.org/10.1145/2883817.2883826 (2016)
69. Nardone, R., Gentile, U., Benerecetti, M., Peron, A., Vittorini, V., Marrone, S., Mazzocca, N.: Modeling railway control systems in Promela. In: Artho, C., Ölveczky, P.C. (eds) Formal Techniques for Safety-Critical Systems—Revised Selected Papers of the 4th International Workshop on Formal Techniques for Safety-Critical Systems (FTSCS 2015), CCIS, vol. 596, pp. 121–136. Springer. https://doi.org/10.1007/978-3-319-29510-7_7 (2016)
70. Nummenmaa, T., Tiensuu, A., Berki, E., Mikkonen, T., Kuittinen, J., Kultima, A.: Supporting agile development by facilitating natural user interaction with executable formal specifications. ACM SIGSOFT Softw. Eng. Notes **36**(4), 1–10 (2011). https://doi.org/10.1145/1988997.2003643
71. Paigwar, A., Baranov, E., Renzaglia, A., Laugier, C., Legay, A.: Probabilistic collision risk estimation for autonomous driving: validation via statistical model checking. In: Proceedings of the IEEE Intelligent Vehicles Symposium (IV 2020), pp. 737–743. IEEE. https://doi.org/10.1109/IV47402.2020.9304821 (2020)
72. Puch, S., Fränzle, M., Gerwinn, S.: Quantitative risk assessment of safety-critical systems via guided simulation for rare events. In: Margaria, T., Steffen, B. (eds) Proceedings of the 8th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation: Verification (ISoLA 2018), LNCS, vol. 11245, pp. 305–321. Springer. https://doi.org/10.1007/978-3-030-03421-4_20 (2018)
73. Rispoli, F., Castorina, M., Neri, A., Filip, A., Di Mambro, G., Senesi, F.: Recent progress in application of GNSS and advanced communications for railway signaling. In: Proceedings of the 23rd International Conference Radioelektronika (RADIOELEKTRONIKA 2013), pp. 13–22. IEEE. https://doi.org/10.1109/RadioElek.2013.6530882 (2013)
74. Ryabtsev, M., Strichman, O.: Translation Validation: From Simulink to C. In: A. Bouajjani, O. Maler (eds.) Proceedings of the 21st International Conference on Computer Aided Verification (CAV 2009), LNCS, vol. 5643, pp. 696–701. Springer (2009). https://doi.org/10.1007/978-3-642-02658-4_57
75. Siemens: X2Rail-1 Deliverable D5.1: Moving Block System Specification (2019). https://projects.shift2rail.org/download.aspx?id=a81c93c2-36a5-46cf-8bd8-4924ae612dd7
76. ter Beek, M.H., Legay, A., Lluch Lafuente, A., Vandin, A.: Statistical model checking for product lines. In: Margaria, T., Steffen, B. (eds) Proceedings of the 7th International Symposium on Leveraging Applications of Formal Methods, Verification and Validation: Foundational Techniques (ISoLA 2016), LNCS, vol. 9952, pp. 114–133. Springer. https://doi.org/10.1007/978-3-319-47166-2_8 (2016)
77. ter Beek, M.H., Borälv, A., Fantechi, A., Ferrari, A., Gnesi, S., Löfving, C., Mazzanti, F.: Adopting formal methods in an industrial setting: the railways case. In: ter Beek, M.H., McIver, A., Oliveira, J.N. (eds) Formal Methods–The Next 30 Years—Proceedings of the 3rd World Congress on Formal Methods (FM 2019), LNCS, vol. 11800, pp. 762–772. Springer. https://doi.org/10.1007/978-3-030-30942-8_46 (2019)
78. ter Beek, M.H., Gnesi, S., Knapp, A.: Formal methods for transport systems. Int. J. Softw. Tools Technol. Transf. **20**(3), 355–358 (2018). https://doi.org/10.1007/s10009-018-0487-4
79. Tueno Fotso, S.J., Frappier, M., Laleau, R., Mammar, A.: Modeling the hybrid ERTMS/ETCS level 3 standard using a formal requirements engineering approach. Int. J. Softw. Tools Technol. Transf. **22**(3), 349–363 (2020). https://doi.org/10.1007/s10009-019-00542-2
80. UNISIG: FIS for the RBC/RBC handover, version 3.1.0 (2016)
81. van der Poll, J.A., Kotzé, P.: What design heuristics may enhance the utility of a formal specification? In: Proceedings of the 2002 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement through Technology (SAICSIT 2002), pp. 179—194. South African Institute for Computer Scientists and Information Technologists (2002)
82. World's Fastests High-speed Trains in Commercial Operation in 2020. https://www.maglev.net/worlds-fastest-high-speed-trains-in-commercial-operation