https://doi.org/10.1007/s00165-021-00556-1
BCS © 2021
Formal Aspects of Computing

Formal Aspects of Computing

Analysing an autonomous tramway positioning system with the UPPAAL Statistical Model Checker

Davide Basile[1] and Alessandro Fantechi[2] and Luigi Rucher[3] and Gianluca Mandò[3]

[1] Institute of Information Science and Technologies "Alessandro Faedo" - ISTI, National Research Council, Pisa, Italy
[2] Department of Information Engineering, University of Florence, Florence, Italy
[3] Thales Italia S.p.A., Florence, Italy

Abstract. The substitution of traditional occupancy detecting sensors with an Autonomous Positioning System (APS) is a promising solution to contain costs and improve performance of current tramway signalling systems. APS is an onboard system using satellite positioning and other inertial platforms to autonomously estimate the position of the tram with the needed levels of uncertainty and protection. However, autonomous positioning introduces, even in absence of faults, a quantitative uncertainty with respect to traditional sensors. This paper investigates this issue in the context of an industrial project: a model of the envisaged solution is proposed, and it is analysed using UPPAAL Statistical Model Checker. A novel model-driven hazard analysis approach to the exploration of emerging hazards is proposed. The analysis emphasises how the virtualisation of legacy track circuits and on-board satellite positioning equipment may give rise to new hazards, not present in the traditional system.

Keywords: Railway, Signalling, Statistical model checking, Uppaal, Autonomous positioning system

1. Introduction

Innovations in the railway sector are currently studied for improving the capacity of the railway network whilst reducing the costs for installing and maintaining line-side equipment. Indeed, modern computer-based, safety-critical railway signalling systems are often considered expensive both for what concerns investment costs to equip railway lines, and for the costs of their periodic preventive maintenance.

One promising solution to contain such costs is the substitution of costly occupancy sensors (such as track circuits or axle counters) by on-board location units based on satellite positioning; such alternative has however to preserve, or even improve, the safety level of consolidated sensor technologies. This option is one of the themes under the Multiannual Programme of the Shift2Rail Joint Initiative [Shi15] (Innovation Programme 2, Technical demonstrator 2.4 "Fail-Safe Train Positioning, including satellite technology") and is the subject of several research projects.

Correspondence to: Davide Basile, e-mail: davide.basile@isti.cnr.it

Published online: 29 July 2021

This paper presents results of the SISTER project [CBB+19], funded by Tuscany Region, which has focused on a scaled down objective at this respect, that is, substituting track circuits installed on tramway lines with an integrated on-board solution including satellite positioning and an inertial navigation platform. In such applications, track circuits are adopted for detecting track occupancy and allowing an *Interlocking* to grant free routes to incoming trams through a junction. Note that in the case of tramways, safe distancing and compliance to signals is a responsibility left to the human driver and not to automated equipment.

The principles adopted for the substitution of track circuits are based on the on board computation of the current tram position calculated autonomously from the received satellite signals, compared against information received from other sensors. This information is then sent to the Interlocking system that need to know what track circuits are occupied. Satellite positioning comes however with uncertainty, worsened in urban environments by buildings along the tramway tracks, which create the so-called "urban canyons". This is known as the *multi-path problem* in signal reception, i.e., interference due to waves refraction and reflection [BM12, RCN+13, GJRS13, FNF18]. These problems could in principle generate false track section occupancy detection, or miss some true track section occupancy, with obvious consequences on the safety of the overall system.

For this reason, the SISTER project has developed a safety assessment procedure, particularly focused on the added hazards introduced by satellite positioning uncertainties. In this paper, we address such hazards by means of a formal model of the SISTER system, in which Stochastic Timed Automata are used to enable reasoning on the probability of events, by means of Uppaal SMC, a Statistical Model Checker. Our primary interest is to investigate whether some particular hazard may actually occur, and the relation between their probability of occurrence and their known uncertainty sources. Note that we assume the presence of a safety layer to protect against threats identified by CENELEC 50159 [Eur10] (e.g., masking, corruption, insertion) preventing, e.g., man-in-the-middle attacks. Threats due to high uncertainty in the positioning system are analysed.

This paper is an extension of the conference paper [BFRM19]. In particular, all models have been revised and a new entity originally present in the system specification, the Operational Control Center, has been modelled and analysed. Moreover, the Interlocking and On-board Computer components have been integrated with new modules and all components are described in detail. The hazard analysis in [BFRM19] is extended to model and analyse 27 new hazards, together with new scenarios and set-up of parameters. Finally, different possible mitigations are proposed to fix problems encountered in the original specification.

We first provide a short background section on statistical model checking and Uppaal (Sect. 2), after which we describe the project and the proposed methodology (Sect. 3). The system and how substitution impacts its principles of operations is discussed in Sect. 4. The formalization of Virtual Track Circuits is discussed in Sect. 5, and the formal model is presented in Sect. 6. The conducted analysis and experiments are in Sect. 7. Finally, related work is in Sect. 8 whilst conclusion and future work are in Sect. 9.

2. Background

Statistical model checking (SMC) [LLT+19, AP18] is concerned with running a sufficient number of (probabilistically distributed) simulations of a system model to obtain statistical evidence (with a predefined level of statistical confidence) of the quantitative properties to be checked. SMC offers advantages over exhaustive (probabilistic) model checking. Most importantly: SMC 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. Moreover, the required simulations can be easily distributed and run in parallel. This comes at a price: contrary to traditional and to probabilistic model checking, exact results are out of reach. Another advantage of SMC is its uptake in industry: compared to model checking, SMC is very simple to implement, understand and use, due to the widespread adoption of Monte Carlo simulation.

Uppaal SMC [DLL+15] is an extension of Uppaal [BDL+06], a well-known toolbox for the verification of real-time systems modelled by (extended) timed automata. Uppaal SMC models are Stochastic Timed Automata: Timed Automata are finite state automata enhanced with real-time modelling through *clock* variables; their stochastic extension replaces non-determinism 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. A signal sent by a broadcast channel (e.g., through a transition with synchronization `chan!`, where `chan` is the name of the broadcast channel) can be received by all entities that are listening (i.e. whose transition with synchronization `chan?` is enabled in their current state). Note that the operation of sending a signal through a broadcast channel is non-blocking: thus a signal could not be received by any entity and this would not lead to a deadlock. Uppaal SMC allows to check (quantitative) properties over

simulation runs of an UPPAAL SMC model (i.e. a network of stochastic timed automata). These properties must be expressed in a dialect of the Metric Interval Temporal Logic (MITL) [BDL+13].

When evaluating a probabilistic property φ, the tool will not return the exact value of φ, but rather an interval obtained through simulations. The number of simulations is decided based on the statistical parameters of the model checker. In Sect. 7 we will specifically refer to the probability of false negatives α and probability uncertainty ε (also referred to as u). Let p' be the probability that has been estimated through simulations and p be the unknown "real" probability whose value is approximated by p'. The tool executes $N = \lceil (ln(2) - ln(\alpha))/(2\varepsilon^2) \rceil$ simulations ρ_i, $i \in 1...N$, sufficient to guarantee that $\mathtt{Pr}(\mid p' - p \mid \leqslant \varepsilon) \geqslant 1 - \alpha$, that is the interval $[p' - \varepsilon, p' + \varepsilon]$ contains p with probability greater than or equal to the confidence interval $1 - \alpha$, where $p' = (\#\{\rho_i \mid \rho_i \models \varphi\})/N$. Basically, reducing α or ε requires a greater number of simulations to provide the above guarantees on the estimated value of φ.

A drawback of SMC is that it can have difficulties in efficiently observing rare events. Indeed, the number of simulations could not be sufficient to observe rare events that can be found in only a small fraction of runs, or in no run at all. An example of a rare event could be the estimation of the position with higher level of uncertainty, because this has a very low probability. Since we are interested in evaluating the probability of occurrence of particular hazards occurring in the presence of these rare events, we easily run in this problem. To avoid this, we artificially inflate the probability of occurrence of such rare events, to be able to observe them in some run (see Sect. 7).

We briefly summarize the three possible queries used for the description of the properties shown in Sect. 7.

- Firstly, the query `Pr(<>[t,t'] ap)` denotes the probability that a random simulation run of a model M reaches a configuration satisfying the atomic proposition `ap` in the interval of time that goes from t to t' time units.
- Similarly, the query `Pr[x<=bound](<>cond)` evaluates the probability that the condition `cond` is satisfied before the variable `x` exceeds the `bound` value, which is an integer. In case `x` is not specified, the formula must be satisfied before `bound` time units. Generally the condition specifies a certain system configuration. This property can also be expressed in the form `Pr(<>[0, bound] cond)` with the same semantics.
- Finally, queries of the form `Pr(<>[0, bound1]([][0, bound2] cond))` calculate the probability that a system configuration satisfying `cond` within `bound1` time units will be reached. This condition must continue to be satisfied for subsequent `bound2` configurations. For example, the property `Pr(<>[0,150]([][0,7]connectedCounter [0]>0&&IXLR_0.WaitingLoc))` reported in Sect. 7 measures the probability that a state is reached in which the number of connected trams is positive (`connectedCounter[0]>0`) and the IXL module is waiting to receive a new position from one of the connected trams (`IXLR_0.WaitingLoc`) for at least 7 units of time (`[][0,7]`), and this must occur within 150 time units (`<>[0,150]`). In this case the property measures the probability of verification of a hazard. In fact, we want that within a certain time threshold the IXL actually receives the position from at least one of the currently connected trams. In the formula, this threshold is set to a value lower than 7 time units.

Finally, UPPAAL SMC allows modeling the system to capture the complex interactions by modeling the behavior of the system, discover and analyze hazards described by tailored formula. Moreover, it allows to express in the model the intrinsic uncertainty of the autonomous positioning as well as real time delays of communications for a more accurate analysis of the system specification.

3. Project and methodology

One of the goals of the SISTER project [CBB+19], funded by Tuscany Region, is to study the substitution of track circuits installed on tramway lines with an integrated on-board solution including satellite positioning and an inertial navigation platform. Initially, the industrial partners provided a description of the current tramway system (i.e. without autonomous positioning and virtual devices). This description was in the form of deliverables describing the requirements of the system and the operative scenarios. It was initially supposed that substituting physical devices with virtual ones would have not produced any adverse effect. During the project, an investigation has been carried out to explore new requirements and possible issues raising from the substitution of physical devices (e.g. track circuits) with corresponding virtualised ones.

The devised methodology of the SISTER project combines aspects of qualitative and quantitative analysis exploiting the expertise of the involved partners. Starting from initial requirements and operational scenarios, these have been further refined, together with the SISTER partners, to build operational scenarios for the new tramway system with autonomous positioning and virtualised equipment. Each scenario has been broken down into small steps, and for each step potential malfunctionings and mitigations were identified by applying guidewords. This activity, carried out jointly by the consortium members, provided the basis for the various tasks of the

project, and in particular it represented the starting point for the formal modelling and analysis phase described in this paper.

The CENELEC norms [Eur11, Eur17a, Eur17b, Bou15] standardising the railway development process (highly) recommend formal methods in most of the phases of the railway development process, including also the requirements phase where our work is positioned. The adoption of formal methods in the railway domain is the subject of recent studies [GtBvdP20, tBGK18]. Quoting [Bou15], "the use of a model without a textual requirement is tantamount to discovering our requirements without having expressed them". Indeed, we have used UPPAAL SMC as an aid for exploring novel requirements and hazards of the envisioned system. Statistical model checking has been used to perform quick verification 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 are fully established. Thus, the purpose of our study is exploratory, and it is not meant to detect all possible hazards nor to analyse all possible scenarios that could occur in the future realization of the new system.

The formalisation and analysis carried out in this paper has been of help in formalising crucial requirements of the Autonomous Positioning System (APS) and in identifying new hazards previously overlooked, and it has been a step forward in the future realisation of the envisioned system.

In particular, initially it was established that the position uncertainty results in overapproximating each tram with an envelope containing the tram, where the longer the envelope the higher the probability of it containing the real position of the tram (see Sect. 4). It was conjectured that this approximation would suffice for ensuring safety. The formal modelling and analysis has surprisingly discovered corner cases where safety could be at risk also when approximating the real position of each tram (see Sect. 7).

Model-driven hazard analysis A hazard is the potential for harm or an adverse effect on the system under analysis. Hazard Analysis [Eur17a, Eur17b] identifies hazardous events that may have a potentially serious impact on the system, on the environment and on people who interact with the system either directly or indirectly. For the system to be considered safe, the probability of hazards to occur (known as Tolerable Hazard Rate (THR))) must be lower than the identified Safety Integrity Level (SIL). The safest level is SIL4 $\approx 10^{-9}$, whilst, e.g., SIL3 $\approx 10^{-6}$.

When hazard occurrence probability cannot be provided by reliable estimation techniques, e.g. for hazards related to software faults, SIL is adopted to grade the robustness that a system is expected to achieve through an appropriate set of recommended development techniques (see, e.g., [Eur11]). In particular, adoption of formal methods in software development is highly recommended for SIL4 systems [Bou15], in order to reduce, and even get rid of, the residual software faults that can trigger hazards. The formal model in this paper is however not targeting software development, but the occurrence of hazards at the system level. It is aimed neither to provide a precise quantitative evaluation of the THR, but rather to verify qualitatively whether in the presence of adverse rare events (e.g. worse connection, wide positioning error) the system behaves as expected or incurs in a hazard. Hence, we do not give a quantitative guarantee of safety by means of showing that THR is lower than the admitted level. Instead, by means of inflating the probability of occurrence of adverse conditions, we quickly verify whether the hazard is likely to occur or its probability of occurrence remains low, and we leave to further work the accurate quantitative evaluation of the THR.

To the best of our knowledge there are no well-established, largely accepted tools that allow automating the application of a hazard analysis: the activity is usually performed by hand, thus becoming a time-consuming and error-prone activity, albeit some supports exist [E+15].

In this paper, we advocate a model-driven approach to the hazard analysis where each element of the formal model (e.g., states, transitions, constraints) is systematically reviewed for discovering potential malfunctioning. This approach consists in exploring new hazards by expressing, using logic formulae, the negation of the expected/modelled behaviour. Thanks to the rigorous formal framework, it is possible to formalize the discovered hazards in an unambiguous way, and formally verify them against the proposed operational model, as discussed in Sect. 7. Of course, if the specification is correctly modelled and configured, it is expected that the probability of a hazard to occur is low, although this is not always the case (see Sect. 7). However, it is important to note that the discovered hazards are independent from the behavioural model used to elicit them and the specific scenario in which they are analysed. The hazards must occur with probability close to zero in all other behavioural models designed from the same specification of the system for all safe configurations. Indeed, whilst the target of the proposed formal model is the exploration of new hazards, there could be future models used for other goals, e.g., code generation, model-based testing, or new configurations of the system to be verified.

The adoption of a formal model to drive the discovering of new hazards is part of the novel contribution of this paper, which increases the confidence on the usefulness of formal methods in the earlier phases of a system design. It must also be noted that the hazards discovered in this phase must pass through the same process used

for other hazards, e.g., to evaluate their impact and gravity, and that the proposed methodology is not supposed to substitute any phase of the railway development process, but to integrate and enhance existing practices.

4. Description of the system

The goal of the SISTER project is to provide a new localization system that replaces the old positioning devices such as track circuits and axle counters with virtual ones. A major advantage of adopting such a solution is to reduce the cost of maintenance of ground-based equipment. Moreover, urban environment does not facilitate the deployment of such equipment, due to road shared usage with other vehicles, pedestrian crossings, etc.: this disadvantage can be overcome by equipment virtualization, freely optimizing the position and number of virtual devices.

The new localization system is based on a proprietary sensor fusion algorithm developed by Thales, which we consider to be a black box. This algorithm calculates the (virtual) position of the tram based on different sensors including satellite tracking, inertial measurement unit, odometer. The position is provided together with an estimation of the uncertainty. This is of help for reducing the satellite positioning uncertainty when, for example, a tram is traversing a gallery and the satellite signal is not available.

In this paper we will focus on the satellite-based positioning feature of the system, that gives the position information in terms of the current location on a line with a related uncertainty: for the purpose of studying rare, but possible hazards, we take a pessimistic approach about uncertainty estimation. The positioning information is communicated periodically (e.g., at a frequency of 10Hz) by a tram to the relevant interlocking (IXL) and to the operations control centre (OCC).

A first proposal is aimed at leaving unchanged the configuration parameters of the existing system, i.e. the position and number of tram detectors on the ground are exactly the same as their virtual counterpart. This solution still takes into account physical installation constraints that are no longer necessary, but it is compatible with the legacy system, i.e. it does not require modification and a consequent new certification of the specific IXL application.

Figure 1 shows the principles of operation of the legacy system vs. the new solution, when a tram is approaching a junction. Notice that in tramway systems junctions are relatively simple, and traffic over the line between two junctions is not subject to signalling. A local IXL system is used instead for guaranteeing a safe transit through a junction: when approaching a junction, a tramway has to ask for a route among those available through the junction: this is automated in the legacy solution by asking for a route to the local IXL after reading a *Tag* (i.e., a physical object detected by the tram containing information such as its exact position [ERAMK18]).

The route is assigned by IXL on the basis of the occupancy of *Track Circuits (TC)* (green ones are free, the red one is occupied), points are set according to the route, and the protecting signal is set to green. Notice that it is the driver's responsibility to drive safely through the assigned route, respecting signals. Also, in tramway lines TC are not contiguous and are located in specific positions. Although very simple, the junction layout of Fig. 1 can be considered representative enough of the few different junction layouts that are typically considered in a tramway system.

In the satellite-based solution, shown in the lower part of Fig. 1 the tram knows its position, and compares it with an onboard map listing the *Virtual Tags (VT)* coordinates on the network. If the position matches a VT, a connection is established with the local IXL (that is, the IXL controlling the junction area referred by the VT), requesting a route to tram's destination. The connection establishes a communication session in which a tram periodically communicates its position to IXL, which check the received position against *Virtual Track Circuit (VTC)* coordinates, to detect their occupancy. Indeed, in order to keep the legacy IXL unchanged, connection and VTC mapping are performed by a wrapper layer that interfaces to IXL. Route, points and signals setting are the same as in the legacy solution. The connection session terminates when a tram leaves the junction area.

Notice that the correct mapping of position to VTC is a safety-critical function, rated SIL4, as are the other IXL functions. On the converse, the position-to-VT mapping is not considered safety critical. Indeed, the failure to connect to an IXL does not open any protecting signal, and there are no safety threats.

Communication with OCC by trams and IXLs (not depicted in Fig. 1) is aimed at monitoring and performing high level functions of traffic regulation. Notwithstanding its safety-critical nature, the IXL is not fully modeled, because it is considered a legacy system, already certified at SIL4, which is not changed in this solution, apart from virtualising track circuits. Only the SISTER layer will be modeled and analysed. Note that we consider virtual track circuits part of the SISTER layer. Basically, the unchanged portion of the IXL is oblivious of whether the track detection has been performed physically (through track circuits) or automatically (through virtual track circuits).

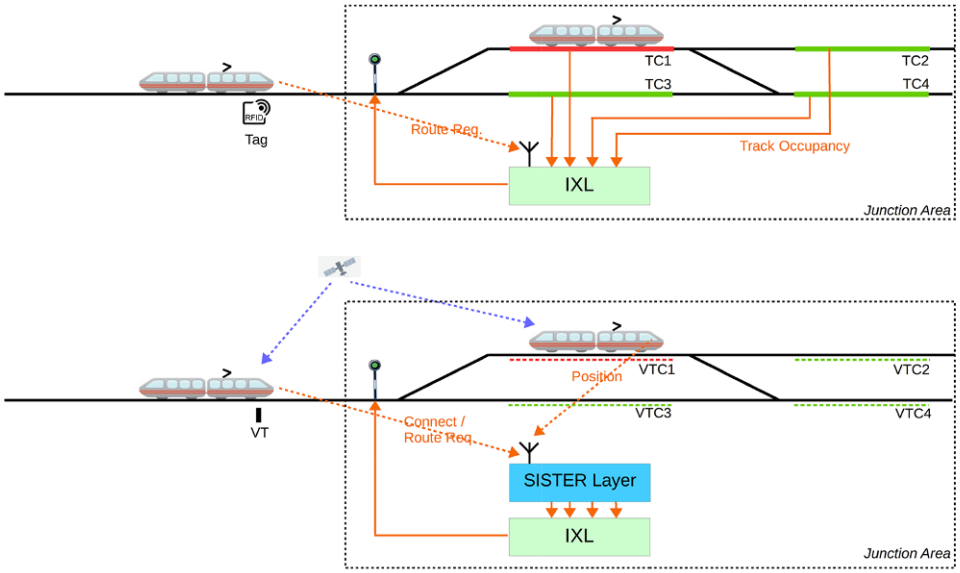

Fig. 1. Substituting track circuits with satellite positioning

In the following, we assume that position information (i.e. coordinates of tram, VT and VTC) are actually uni-dimensional, i.e., are mapped over positions on a line (representing the route a tram is following). Indeed, the current setting of points (for track branches) is assumed to be performed according to the assigned route, allowing therefore precise positioning information at this respect, ensuring that the tram is in its assigned track (even in the case of parallel tracks). Therefore, the experiments that will be discussed in Sect. 7 do not require multiple routes assigned to several trams. Hence, abstracting away from two dimensional space is of help for improving performances and complexity factors.

Virtual Tags and Virtual Track Circuits are two new components of the SISTER system:

- *Virtual Track Circuits* (VTC) correspond to location ranges (track sections) within a virtual map on the ground system (i.e., interlocking, operations centre): they are hence considered as intervals.
- *Virtual Tags* (VT) are similar but are only available on the map of each tram. Traditional tags could be seen as points on the line. However, in order to properly compare the satellite-based position with the tag position, it is safer to define a VT as a (shorter) interval as well.

In both cases, the detection (i.e. if a tram occupies a VTC or has read a VT), will be implemented through a function (called *LocationReferencing*) that will compare the coordinates of the tram with the map, where both VTC and VT are identified by intervals $[a, b]$.

5. Formalising virtual track circuits

In this section the function *LocationReferencing*, used for checking if a virtual track circuit or virtual tag is occupied, is formally defined. At a given time, let L_v be the virtual position of a tram, i.e. the positioning data calculated by the algorithm on board the tram, and let L be its actual position. As a first contribution, we argue that safety must be guaranteed assuming the presence of an error ε between L_v and L such that $L = L_v \pm \varepsilon$. If such uncertainty is ignored, potential hazards are introduced, as it will be described below.

These safety requirements are inspired by their aeronautical counterparts [ESA18, LBC+15]. In particular, the actual error is unknown, hence an uncertainty ε is introduced which is an aleatory variable that follows a Normal distribution with average zero (denoted φ_{0,σ^2}). The sensor fusion algorithm periodically provides L_v and φ_{0,σ^2}. The *protection level* (PL) is a statistical bound error computed from L_v and φ_{0,σ^2} such that it guarantees that the probability of the (unknown) real position error exceeding PL is smaller than or equal to a target value (called *integrity risk* (*IR*)). In other words, the interval $[L_v - \mathtt{PL}, L_v + \mathtt{PL}]$ contains the real position with probability greater or equal to $1 - IR$ (e.g. $1 - 10^{-9}$). The *alert limit* (*AL*) is the maximum value of PL allowed. Finally, the *time-to-alert* (TTA) is the maximum allowable time elapsed in which $PL > AL$ before an alert is issued and a degraded operation mode is entered.

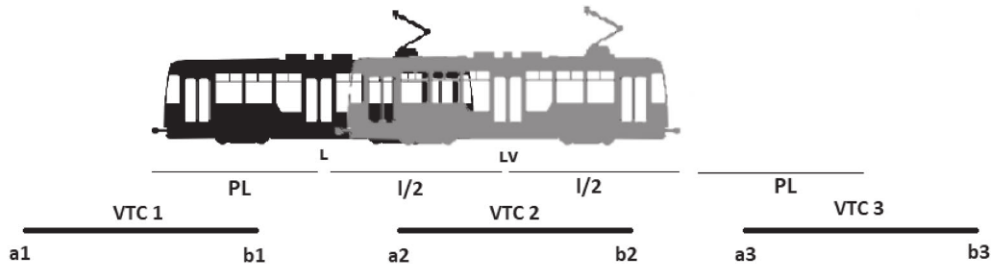

Fig. 2. Example of the protection level PL reducing the capacity of the network

In this paper, these quantities are either abstracted away or they are global constants and global variables to be set. For simplicity, it is assumed that the satellite receiver is located in the centre of the tram and hence L_v is centered in the tram. The location L_v is refreshed according to a constant travelling speed of the tram. The period in which the location is refreshed will be one of the parameters to set for the analysis performed in Sect. 7. More importantly, to simulate the behaviour of the sensor fusion algorithm, periodically a value of PL is selected non-deterministically (using the select construct of UPPAAL) from a given interval. Hence, we completely abstract away from the way in which PL is calculated and consider the sensor fusion algorithm as a black box which provides both L_v and PL.

Let l be the length of the tram, then the corresponding VTC is occupied if $LocationReferencingVTC(L_v, \mathrm{PL}, a, b) = true$ (where a and b identify the interval associated with the VTC); otherwise it is free, formally:

$$LocationReferencingVTC(L_v, \mathrm{PL}, a, b) = \left(a - \mathrm{PL} - \frac{l}{2} \leqslant L_v\right) \wedge \left(L_v \leqslant b + \mathrm{PL} + \frac{l}{2}\right) \tag{1}$$

We remark that the function $LocationReferencingVTC(L_v, \mathrm{PL}, a, b)$ is exclusively calculated by the IXL. This function is implemented in the IXL software and replaces the task performed by the train detectors and is part of the SISTER layer (see Sect. 4). Similarly, let $[a, b]$ be the position of the Virtual Tag in the map, the tram "reads" the virtual tag when the function $LocationReferencingTagV(L_v, \mathrm{PL}, a, b)$ evaluates to true, where its implementation is analogous to the function $LocationReferencingVTC$. However, this function is calculated exclusively by the on-board system.

With virtual positioning, an uncertainty about the real position of the trams is thus introduced. This implies an approximation of the number of occupied VTC.

Example 5.1 Figure 3 shows a vehicle that physically (black tram) occupies VTC 1 and VTC 2, but virtually (grey tram) also VTC 3. Indeed in this example $L_v - \mathrm{PL} < L < L_v$ and the physical position is behind the virtual one, but inside the protection level PL provided by the virtual position. Note that, in case PL would be ignored, VTC 1 would be erroneously considered free (i.e. a false negative). Through PL this false negative is removed but a false positive is introduced (i.e. VTC 3 is detected occupied whereas it is free). Indeed, $LocationReferencingVTC$ computes an *over-approximation* of the real track occupancy and in this example evaluates to true for all the three intervals. Recall that, as explained in Sect. 4, VTC are not contiguous.

6. Formal model

In this section we will provide the details of the formal model designed in UPPAAL. We modelled three main entities: the Operational Control Centre (OCC – one for the whole system), the On-Board Computer (OBC – one for each tram) and the Interlocking (IXL – one for each junction area). For readability, Fig. 3 depicts these three entities as well as their communication channels that will be described below. Quantities are, for example, the number of trams, the number of IXLs, the number of VT and VTC (of each IXL) that are instantiated as global constants. Moreover, other parameters refer to the various delay rates of messages and probability of failure in communications. In particular, all messages have a specific probability $\frac{\mathtt{Pmsg}}{1+\mathtt{Pmsg}}$ of being received (that is, the message loss probability is equal to $\frac{1}{1+\mathtt{Pmsg}}$). Each probability of a transition t to occur is given by the weight of t divided by the sum of the weights of all outgoing probabilistic transitions. The unbounded delays of the communications are exponentially distributed with a parametric rate `msgRate`. These values (probability of message loss and delay rate) are input parameters to the model, thus allowing to analyse several different scenarios, depending on, for example, the devices used.

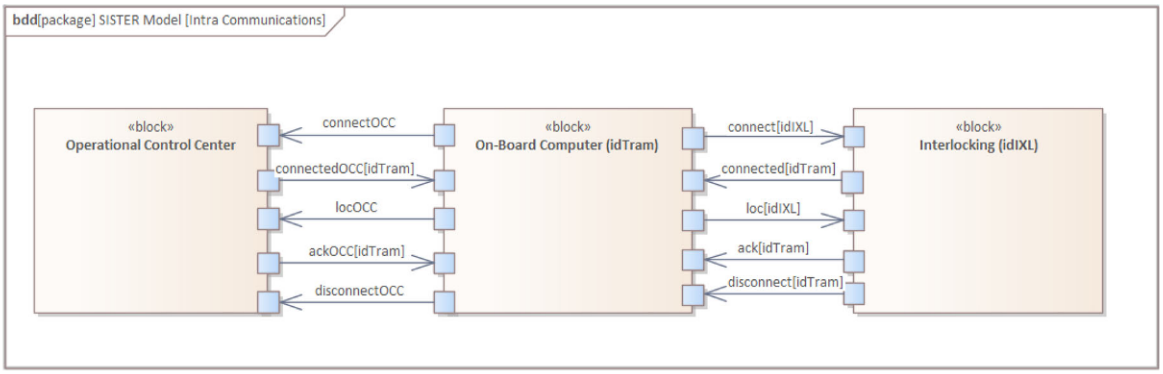

Fig. 3. Intra-Communications diagram

Some states in the model are marked as *urgent* states, i.e. instantaneous states in which the system spends zero units of time. These states are used to divide different operations considered to be atomic, i.e. they are performed in sequence in the same unit of time. Moreover, in UPPAAL each state may have a *state invariant* that must hold only in that state.[1] Indeed, if an invariant is violated then the simulation will stop returning the trace leading to the invariant violation. Invariants may be used for controlling the flow of time, e.g. by constraining a transition to be enabled and fired at a specific instant of time. If, for example, a state has an invariant `c<=k`, with clock `c` and constant `k`, and an outgoing transition guarded by `c>=k`, this will force the transition to be fired when the clock exactly reaches the value `k`, and not later (otherwise the invariant would be violated).

Each transition of an automaton in UPPAAL can contain a guard, a send (!) or receive (?) signal and an update of variables. UPPAAL does not primitively allow value-passing, hence variables (named x,y and z) are used to transmit data together with the signal, thus acting as one-position buffers. Moreover, broadcast channels are organised in arrays indexed by, for example, the identifier of the IXL or the OBC, to allow for broadcast and unicast communications.

All components listed next are *templates* in UPPAAL, which is a mechanism allowing to create different instances of an automaton. This makes it possible to perform simulations and analyses with a certain number of IXLs and OBCs, not fixed beforehand in the model. In Table 1 we report associations between names of instances of templates (used in Sect. 7) and names of corresponding templates (used in this section). Furthermore, this model is parametric and highly customisable. It is possible to analyse different operational scenarios by adjusting the individual parameters of the model. The model is publicly available [Bas21]. We now describe the different components that are combined together to form the overall system.

6.1. Operational control centre

The OCC is in charge of monitoring the whole network of trams. The requirements state that each tram will periodically connect to the OCC, send its position, and then disconnect. In particular, in case a position from one of the connected trams is not received within a certain amount of time, a degraded mode is entered. The OCC model is constituted by a set of automata that are composed together:

`OCC_Connect` this automaton is displayed in Fig. 4. It is used to handle connection requests of trams. In its initial state `Waiting` it has an invariant `OCCconnectedCounter<=nTram` to check that the number of connected trams (`OCCconnectedCounter`) does not exceed the number `nTram` of trams in the system. It receives a request of connection from a tram through the channel `connectOCC`. The identifier of the tram `tid` is transmitted through a buffer `x` (note that in `tid=x,x=0`, the use of comma denote a sequence of two instructions). Basically, the content of the buffer is stored in variable `tid`, and the buffer is reset. The probability of correct message reception is $\frac{\texttt{Pmsg}}{\texttt{Pmsg+1}}$. The probability of losing the message is $\frac{1}{\texttt{Pmsg+1}}$.

[1] To help identifying invariant in the graphical automata representation invariants are enclosed in a box and positioned closer to the (name of the) state they refer to; the box is not present in the original representation produced by UPPAAL.

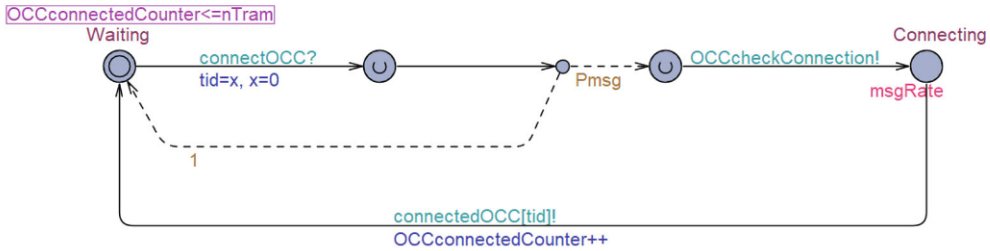

Fig. 4. The `OCC_Connect` automaton

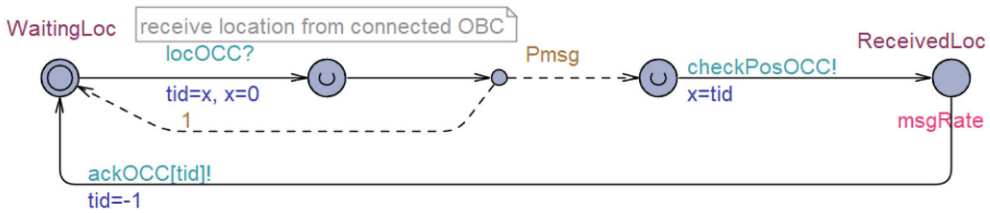

Fig. 5. The `OCC_ReceivePos` automaton

After receiving the request, it resets the module `OCC_Supervision` (signal `OCCcheckConnection`) and then acknowledges the connection to the tram (signal `connectedOCC`) and increases the number of connected trams. The delay in message communication is exponentially distributed with rate `msgRate`, an input parameter.

`OCC_ReceivePos` this automaton is used to receive the position of each tram, and is depicted in Fig. 5. The position is received through the channel `locOCC`. Before acknowledging the reception of position through channel `ackOCC`, this automaton communicates with `OCC_Supervision` to supervise the positions of the connected trams (more details below).

`OCC_Disconnect` this automaton is used to handle disconnections of trams, and is depicted in Fig. 6. It is always ready to receive a disconnection request through the channel `disconnectOCC`. After reception, the number of connected trams is decreased. For sanity check, it is the only state that has an invariant ensuring that there will always be a non-negative number of connected trams.

`OCC_Supervision` this automaton is used to supervise the connected trams, and is depicted in Fig. 7. An array of booleans `check` of length equals to the number of trams is used to flag if a position has been received by a tram (i.e. `check[tid]==true` where `tid` is the identifier of the tram). Two functions `resetCheck()` and `sumCheck()` are used to, respectively, reset the counter and count the number of connected trams that have sent their position within the time limit. Basically, `resetCheck()` reset to false all positions of the array `check`, whilst `sumCheck()` counts the number of elements in `check` that are set to true (i.e. the number of trams that have sent their position). Initially such counter is reset together with a clock `c1`, and the state `NormalMode` is entered. Two invariants are enforced to hold in this state: the time limit `TmaxSafeOCC` is not exceeded or there are no connected trams. Each time a new tram connects, by reception of signal `OCCcheckConnection`, the clock and the counter of received positions are reset (`c1=0, resetCheck()`). Each time a new position is received (`checkPosOCC`), either the counters are reset (if all positions of all connected trams have been received) or the counter of received positions is increased (that is, `check[tid]` is set to true).

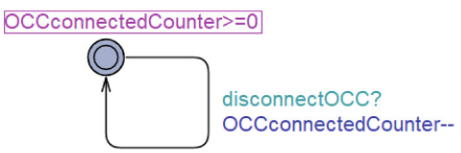

Fig. 6. The `OCC_Disconnect` automaton

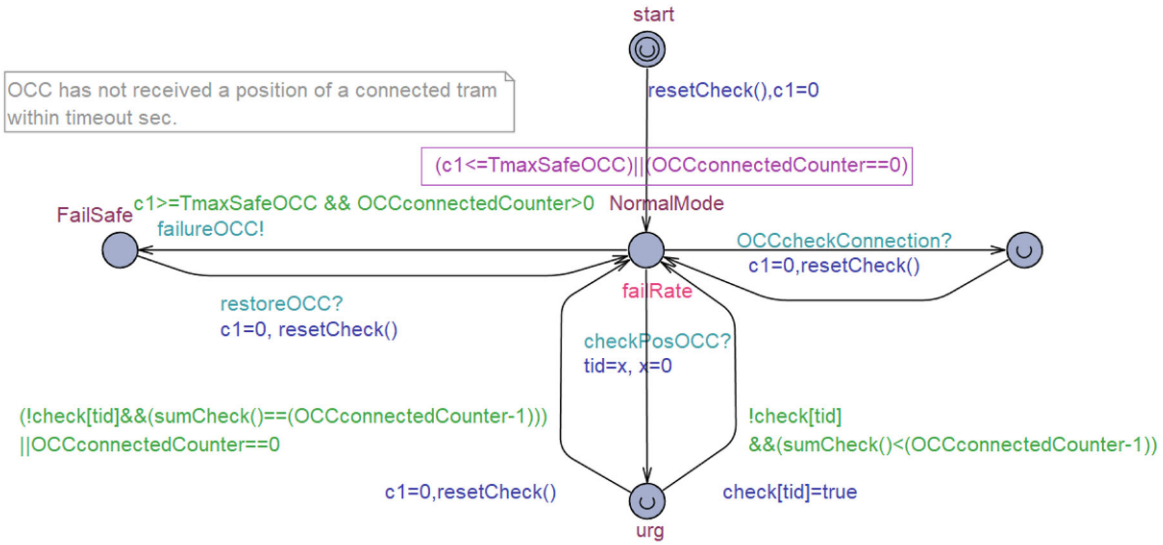

Fig. 7. The `OCC_Supervision` automaton

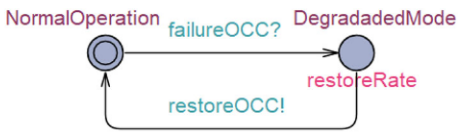

Fig. 8. The `OCC_Mitigation` automaton

If the time limit is exceeded and there are connected trams the `FailSafe` state is reached. The delay causing the entrance in the failsafe mode is exponentially distributed with rate `failRate`. The system then enters again in `NormalMode` state upon occurrence of mitigation (signal `restoreOCC`).

`OCC_Mitigation` this automaton is used to collect the various alerts for entering a degraded mode and actuating the mitigation, and is depicted in Fig. 8. This model abstracts away from physical intervention to restore the system to its normal state. Here the mitigation is modelled as a (broadcast) signal `restoreOCC` and the *mean-time-to-recovery* is modelled as a delay with exponential distribution whose rate is an input to the model (`restoreRate`).

6.2. On-Board Computer

The On-Board Computer (OBC) component provides the functions that are necessarily performed on board of a tram. It interacts with the IXL of each encountered junction area and with the OCC. The autonomous positioning functionalities are implemented inside the OBC. Several hazards may arise and are handled in this unit as described below, e.g. failure in establishing and maintaining a connection with both the OCC and an IXL. Each OBC, uniquely identified by its identifier, is modelled as a set of automata that are composed together:

`OBC_CommOCC` this automaton is handling the OBC interactions with the OCC, and is depicted in Fig. 9. In the initial state the tram is `Disconnected`. A parameter `freqOCCconnection` models the period in which the tram connects to the OCC to send its position (i.e. one time each `freqOCCconnection` seconds). Clock `c1` is used to check that this period is respected. When ready, the tram sends a connection request `connectOCC` to the OCC, and moves to state `Connecting`. In this state, the tram is waiting a connection acknowledgement from the OCC. The tram will continue to send requests of connection with that period until an acknowledgement is received. However, in case no acknowledgement is received within `NmaxOCC` time units, a failure is issued. This check is performed by both a state invariant (`c2<=NmaxOCC`) and a transition guard (`c>=NmaxOCC`).

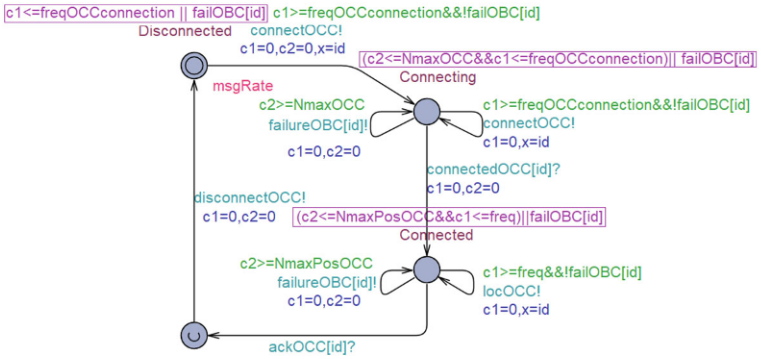

Fig. 9. The `OBC_CommOCC` automaton

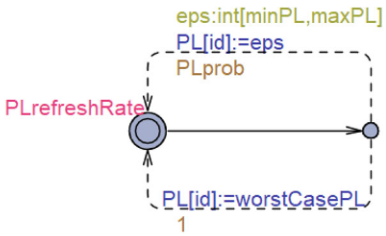

Fig. 10. The `OBC_PL` automaton

When the acknowledgement is received (`connectedOCC`), the tram moves to state `Connected`. In such state, the tram will proceed to send its position (`locOCC`) to the OCC, until an acknowledgement is received (`ackOCC`). The position is sent each `freq` time unit, another parameter in input to the model. A time threshold `NmaxPosOCC` is set such that if the acknowledgement is not received within the limit, a failure is issued (`failureOBC`), to enter a degraded mode. Finally, after the acknowledgement the train disconnects with the signal `disconnectOCC` and returns to its initial state. Note that this automaton (as well as the other OBC ones) operates when there is no failure. This ensured with the condition `!failOBC[id]` guarding transitions, which requires that the OBC has not failed (the prefix `!` is the notation for negating a Boolean variable, not to be confused with the postfix `!` used to represent an output operation on a channel). In case of failure, operations can be restored by reception of the signal `restoreOCC`.

`OBC_PL` this automaton is used to simulate the protection level, and is depicted in Fig. 10. With a probability (`1-PLprob`) the protection level is set to a worst value `worstCasePL`. Otherwise, with probability `PLprob` the protection level is updated with a stochastic delay modeled with an exponential distribution with rate `PLrefreshRate`. The protection level is selected non-deterministically in an interval `[minPL,maxPL]`. We remark that `minPL, maxPL`, `worstCasePL`, `PLprob` and `PLrefreshRate` are input parameters and can be tuned to account for the different conditions to which the real system is exposed (e.g. crowded urban area).

`OBC_SendPosIXL` this automaton is used to periodically send the position to the IXL when connected, and is depicted in Fig. 11. Initially, the automaton is in `NotConnected` state. Connection to the IXL (more below) occurs when the signal `connected` is received. In state `Connected` the location and identifier of the train are sent periodically to the IXL (according to `freq` parameter) with signal `loc`. When the signal `disconnected` is received the automaton goes back to its initial state.

`OBC_Drive` this automaton is used to simulate the tram movement, and is depicted in Fig. 12. The initial state is used to initialise the variables with the function `initialise()` and the next state is `Moving`. In this state, the location L_v of the tram is refreshed according to a constant travelling speed, modelled by the hybrid clock state invariant `Lv[id]'==speed` (the derivative of the location is the speed), where `speed` is an input parameter. Indeed, by only using standard clocks, the speed would be constrained to be equal to 1. Upon reception of signal `stop` (due to a failure) the state `Stop` is reached where the location is no longer updated.

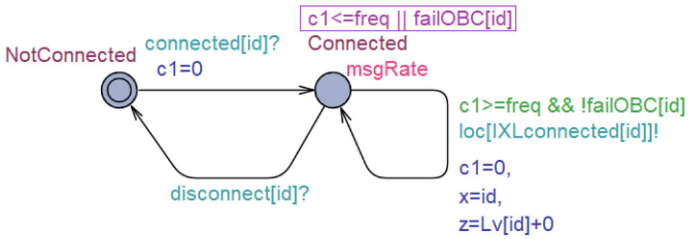

Fig. 11. The `OBC_SendPosIXL` automaton

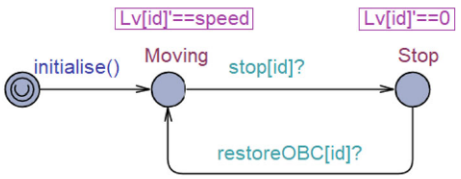

Fig. 12. The `OBC_Drive` automaton

After receiving the signal `restoreOBC` of mitigation of the failure, the tram continues its travel in state `Moving`. The physical behaviour of the tram acceleration and deceleration has been abstracted away mainly because there were no requirements on this aspect. Recall that the driving responsibility is left to the (human) driver and it is not automatic, although some work on modelling train driver behaviour exists [HEPGP21].

`OBC_Mitigation` this automaton is depicted in Fig. 13 and is similar to `OCC_Mitigation`. Its initial state is `NormalOperation`. Upon reception of a signal for entering the degraded mode (`failureOBC`) the tram will be stopped by emitting the signal `stop` and the `DegradedMode` will be entered. The tram restarts with a delay that is exponentially distributed with rate `restoreRate`, an input parameter. The mitigation is notified with signal `restoreOBC`, and the initial state is reached.

`OBC_IXLConnectionSupervision` this automaton is used for handling the connection with an IXL when approaching a junction area, and is depicted in Fig. 14.[2] The initial state is `Disconnect`; when a request for reading a Virtual Tag is received (signal `read`), the state `CheckingTV` is reached. It is required that `IXL!=-1`, that is, the automaton has not been connected so far. The identifier of the VT is stored in the variable `bTVid`, whilst the identifier of the `IXL` to which the VT belongs is stored in the `IXL` variable. The *LocationReferencingTagV* function (see Sect. 5) is guarding the outgoing transitions from state `CheckingTV`, to check whether or not the position of the tram is within the VT.

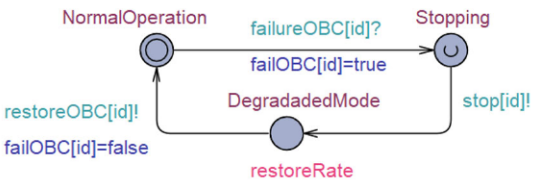

Fig. 13. The `OBC_Mitigation` automaton

[2] In the UPPAAL models extracted from the Italian SISTER project's deliverables, the Italian acronyms TCV and TV are used instead of, respectively, VTC and VT.

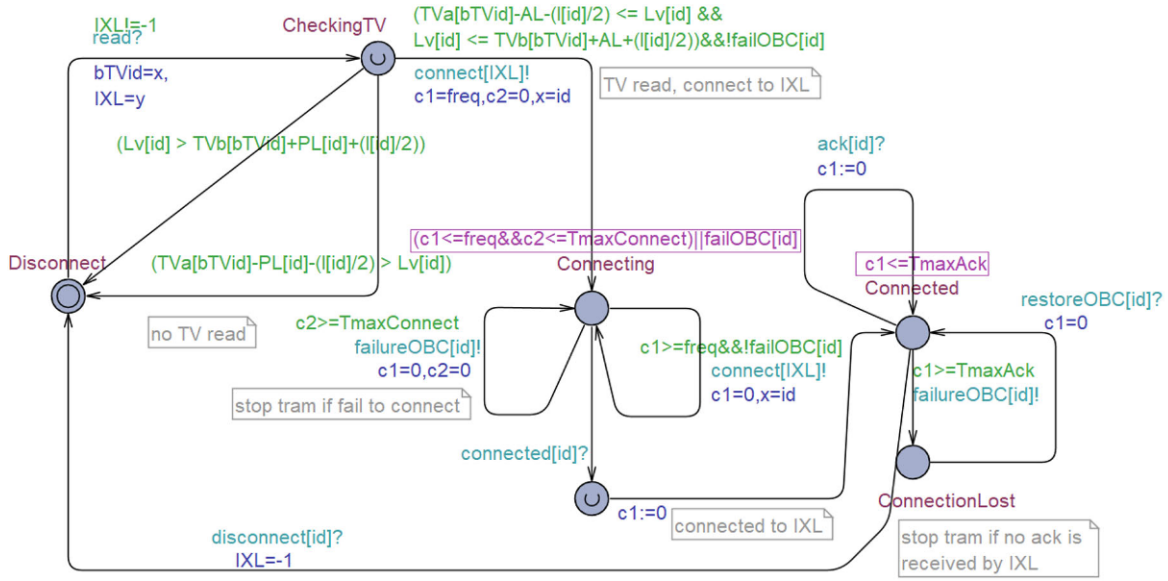

Fig. 14. The `OBC_IXLConnectionSupervision` automaton

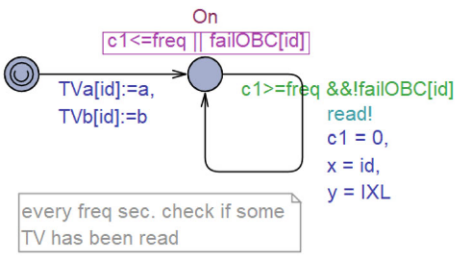

Fig. 15. The `OBC_VT` automaton

If it is not the case, then the automaton returns to its initial state. Otherwise, the VT has been read and the tram proceeds to connect to the IXL in state `Connecting`. Each `freq` time units a signal `connect` is sent to the IXL. A clock `c2` is used to monitor whether the IXL has not replied to the connection request within `TmaxConnect` time units. If this is the case, a failure is issued by the signal `failureOBC`, and the tram will continue to try to connect to the IXL. When the IXL acknowledges the successful connection (signal `connected`), the automaton moves to state `Connected`. A signal `restoreOBC` is issued in case the connection has been restored after a failure. If there was no failure this signal will be ignored (i.e. not received by any other automaton). In state `Connected`, the tram is connected to the IXL and will receive acknowledgement of position reception by the IXL. In case an acknowledgement from the IXL is not received within `TmaxAck` time units, a failure is issued (`failureOBC`) and the automaton moves to state `ConnectionLost`. When the mitigation has been performed the signal `restoreOBC` will be received and the automaton will return to state `Connected`. Finally, when a signal of disconnection is received (`disconnect`), from state `Connected` the automaton will return to state `Disconnect`, meaning that the tram has been successfully disconnected from the IXL junction area.

`OBC_VT` this automaton is used to check periodically if a Virtual Tag has been read and is depicted in Fig. 15. The period is the parameter `freq`. It takes as parameters the interval $[a, b]$ on the map and the identifier of the IXL related to the read VT (`IXL`), whilst `id` is its identifier. The invariant and the guard on the loop transitions ensure that the checking is performed exactly each `freq` time units. The checking is performed by sending a signal `read` to `OBC_IXLConnectionSupervision` (Fig. 22).

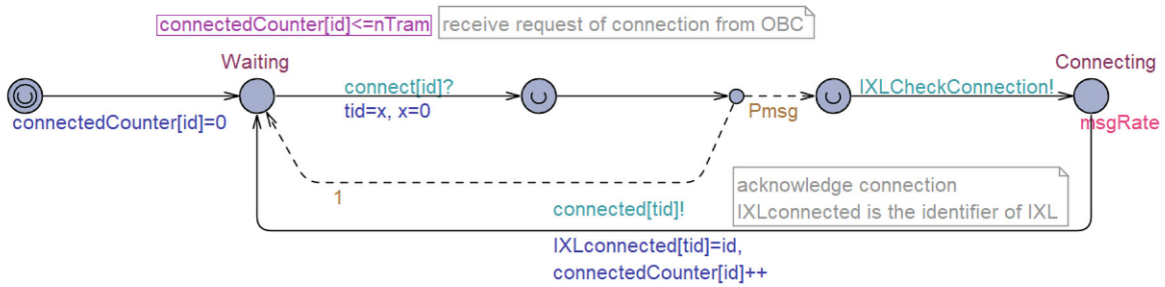

Fig. 16. The `IXL_Connect` automaton

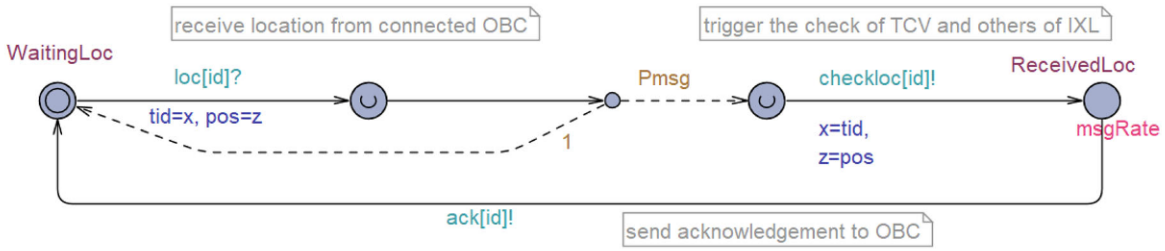

Fig. 17. The `IXL_ReceivePos` automaton

6.3. Interlocking

The IXL automata models the SISTER layer implemented on top of the legacy Interlocking. Its functionalities are related to receive connection from each tram willing to enter the junction area controlled by the IXL, and to disconnect each tram that has traversed it. Moreover, the IXL receives periodically the position of each connected tram and checks the occupancy of the virtual track circuits. Different safety mechanisms have been identified that are related to functionalities of the IXL. Among others, in case a position from one of the connected trams is not received within a certain amount of time, a degraded mode is entered; or in case the uncertainty in a received position is higher than a given threshold, a failure is triggered. Each IXL has a unique identifier, that is a parameter of the automata whose composition builds the IXL module. The automata are:

`IXL_Connect` this automaton is used for serving the connection request from a tram, and is depicted in Fig. 16. Initially, the counter of connected trams is initialised to zero, and the IXL is in state `Waiting`. Upon reception of a connection request (in case the communication succeeded), this automaton sends a signal `IXLCheckConnection` to `IXL_Tmaxsafe_Supervision` for notifying the connection of a new tram, and switches to state `Connecting`. Finally, with a certain delay the successful connection is notified to the tram through signal `IXLconnected`, and the counter of tram (`connectedCounter`) is updated.

`IXL_ReceivePos` this automaton is used to collect the positions of trams and update the IXL status, and is depicted in Fig. 17. The automaton is initially waiting for a location in state `WaitingLoc`. A successful reception of a location by means of the `loc` signal is notified to the automaton `IXL_Tmaxsafe_Supervision`, `IXL_Lsafemargin_Supervision` and the virtual track circuits of competence through the (broadcast) channel `checkloc`, and the state `ReceivedLoc` is reached. After a stochastic delay the acknowledgement is sent to the tram by means of the signal `ack`, and the automaton returns back to state `WaitingLoc`.

`IXL_Tmaxsafe_Supervision` this automaton is used to supervise the trams connected to this IXL, and is depicted in Fig. 18. It is similar to the supervision performed by `OCC_Supervision`. In particular, each time a new tram is connected (`IXLCheckConnection`) the stopwatch is reset. If all positions of connected trams are not received within `TmaxSafe` time units a failure is issued and state `FailSafe` is reached. Normal conditions are restored upon reception of the signal `restoreIXL`.

`IXL_Lsafemargin_Supervision` this automaton is used to check if `PL` exceeds the maximum limit allowed (parameter `Lsafemargin`), and is depicted in Fig. 19.

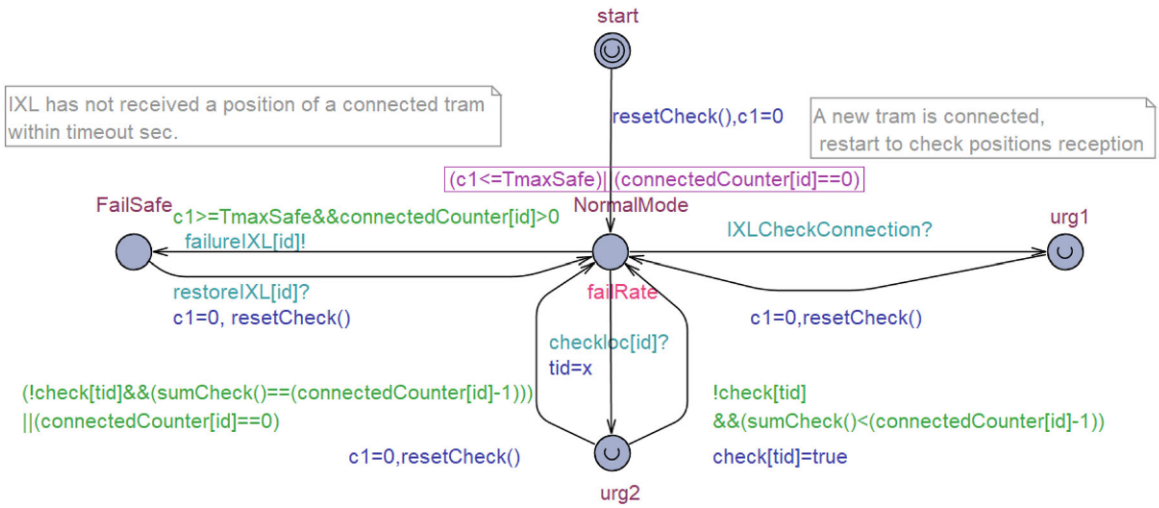

Fig. 18. The `IXL_Tmaxsafe_Supervision` automaton

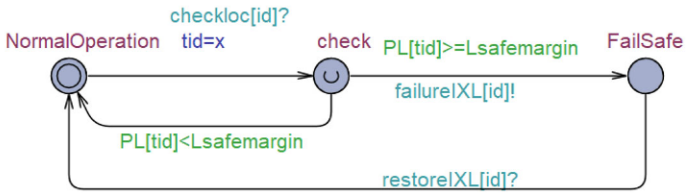

Fig. 19. The `IXL_Lsafemargin_Supervision` automaton

In particular, the automaton starts in state `NormalOperation` and after a reception of a location of a tram, it is checked whether its protection level `PL` is lower than `Lsafemargin`. If this is not the case a failure is issued (`failureIXL`) and state `FailSafe` is reached. Normal operations are restored after the message `restoreIXL` is received.

`IXL_Mitigation` this automaton is used to trigger mitigations in case of failures, and is depicted in Fig. 20. Basically, it has two states `NormalOperation` and `DegradedMode`. Initially, the system is in normal operation conditions. Upon reception of a failure `failureIXL` a degraded mode is entered. The mitigation takes place within a certain stochastic time, modelled as an exponential distributed delay with rate `restoreRate` (an input parameter). The successful mitigation is notified to the other components through the signal `restoreIXL`.

`IXL_Disconnect` this automaton is used for disconnecting a tram once it has traversed the junction area controlled by the IXL and is depicted in Fig. 21. It takes the identifier of the last VTC to be traversed as input parameter for disconnecting the tram (`idtcv`), and a boolean flag (`releaseOccupied`) stating whether the disconnection occurs when the corresponding VTC is occupied or when it is freed (in the legacy system these are called *release conditions*). Initially, the system is in state `Waiting`. In case of release condition set to occupied (i.e. disconnection happens when the last VTC involved is occupied, `releaseOccupied==true`), upon reception of a signal of occupation (`occupied`) from one of the virtual track circuits the check is performed. Otherwise (`releaseOccupied==false`) the check is performed upon reception of signal `freed` from one of the VTC.

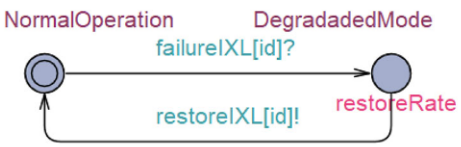

Fig. 20. The `IXL_Mitigation` automaton

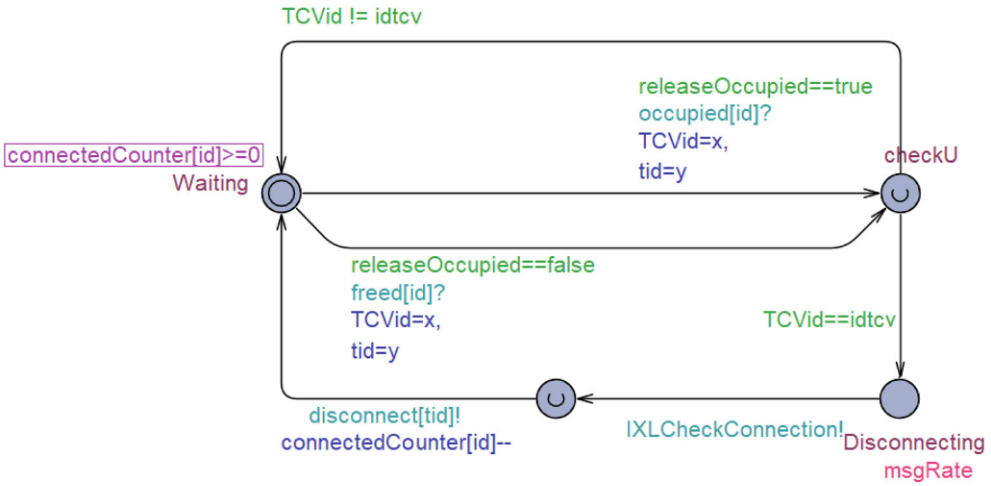

Fig. 21. The `IXL_Disconnect` automaton

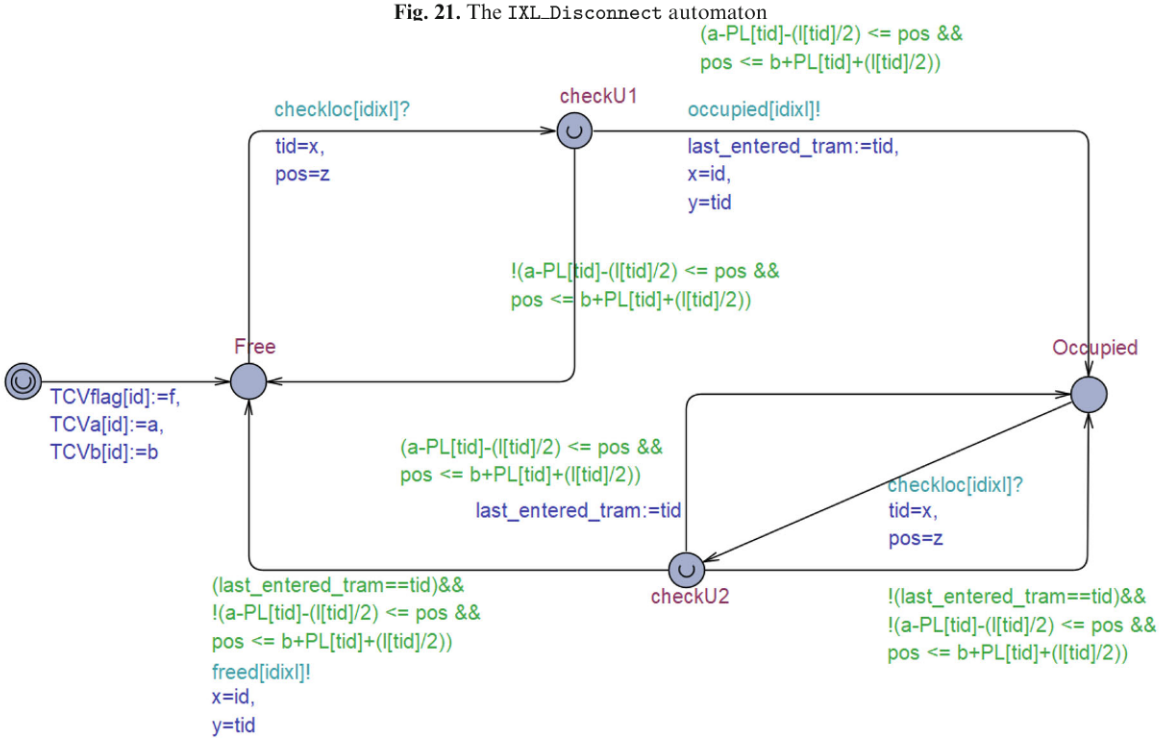

Fig. 22. The `IXL_VTC` automaton

After that, in state `checkU`, if the received signal does not concern the last VTC controlled by the IXL (i.e. `TVCid != idtcv`) the automaton returns to state `Waiting`. Otherwise in case the VTC is indeed used to disconnect the tram (i.e. `TVCid == idtcv`), state `Disconnecting` is reached. From this state, a signal `IXLCheckConnection` is sent to `IXL_Tmaxsafe_Supervision` and a signal of successful disconnection is sent to the tram `tid` which has traversed the last VTC involved (`disconnect`). Finally, the number of connected trams is decreased (i.e. `connectedCounter[id]--`), and the automaton returns to state `Waiting`.

`IXL_VTC` this automaton implements the logic described in Sect. 5 and is depicted in Fig. 22.

This automaton takes in input a unique identifier `id`, a unique `idixl` IXL identifier, two integers `a, b` which identify the track section where the VTC is positioned. The VTC has two main states `Free` and `Occupied` that intuitively identify the occupation and release status of the VTC. The initial transition assigns the respective parameters to globally shared variables. Note that initially the VTC is in state `Free`. Indeed, no tram is assumed to be inside a junction area at start-up and in normal operation conditions. We did not modelled degraded operation conditions and recall that the responsibility is left to the driver. From initial state `Free`, a transition is triggered by the reception of a signal on the channel `checkloc`, a signal coming from the relevant IXL to check the position received by a tram against the VTC.
The `tid` variable stores the unique tram identifier for which the occupation is to be controlled. Likewise, the variable `pos` will contain the position of the tram `tid`. This transition is divided into two possible behaviours through the intermediate urgent state. In particular, it goes back to the `Free` state in the event that *LocationReferencingVTC* evaluates to false (conditions expressed in the transition guards). Otherwise, two operations are performed: first a signal is sent to the interlocking to acknowledge the occupation of the VTC via `occupied`, then the tram identifier is stored in a temporary variable and both VTC and tram identifiers are written to the output buffer, to be used for further operations, and the status reached is `Occupied`. Recall that in the urgent state no time passes: these operations are considered as one atomic operation.
From state `Occupied` a transition is triggered similarly to the transition that exits the `Free` state. However, there are three different conditions allowing to leave the urgent state. In the case where *LocationReferencingVTC* evaluates to true, the target state remains `Occupied`: the tram has not yet freed the VTC. In this case the temporary identifier of the last tram entered is updated. In the event that `LocationReferencing` evaluates to false, there are two possibilities. If the tram that has freed the VTC is the same stored in the temporary variable, this means that it was the last tram to have occupied the VTC. The VTC goes to the `Free` state. Otherwise, some other trams are still potentially occupying the `VTC` (this is possible due to uncertainty), so the VTC remains in the `Occupied` state.

Note that all the IXL components we modelled are part of the SISTER layer shown in Fig. 1. Indeed, we do not model nor analyse the legacy system, but only the SISTER layer. Thus we abstract from, e.g., how route requests are managed by the IXL.

7. Formal analysis and experiments

In this section, the formal analysis of the model presented in the previous section will be addressed. We will analyse series of properties of interest aimed at verifying the absence of errors in the various models composing the system. The properties are expressed in the UPPAAL syntax (see Sect. 2), and discussed. Almost all formulae represent the probability of occurrence of a specific hazard, with only one exception. The properties are expressed on specific instances of templates, and as such must be replicated for all possible OBCs and IXLs present in the specific scenario under analysis. To improve readability, Table 1 contains for each instance the corresponding template automaton.

The verification results obtained with UPPAAL SMC report probabilities close to zero for reaching a failure state (occurrence of a hazard), and probabilities close to one for the properties that the system must satisfy. The results therefore confirm that the model satisfies the analysed properties. We also recall that, in order to efficiently verify these properties, both the probability of communications failure and the protection level will be artificially increased in the experiments. Basically, worse conditions are injected, to measure the effectiveness of the mitigation under analysis. Realistically, the probability of communication failure would be lower and the protection level would be tighter and a large number of simulations would be needed to be able to encounter one of these rare events.

7.1. Analysing the OCC

We start by discussing the analysis of the Operational Control Centre, where each property targets a specific component of the OCC. We list the identified hazards and the formulation of the corresponding properties for each component.

OCC CONNECT We identified two hazards related to the functionality of connection of the OCC:

*HAZ*1: OCC has received a connection request and has not replied in less then t time units.
*HAZ*2: OCC has connected to a tram that was already connected.

Table 1. Names of instances of templates (indexed by i, on the left) and names of corresponding templates (on the right).

Instance	Template
OCCC	OCC_Connect
OCCR	OCC_ReceivePos
OCCD	OCC_Disconnect
OCCS	OCC_Supervision
OCCM	OCC_Mitigation
IXLC_i	IXL_Connect
IXLR_i	IXL_ReceivePos
IXLS_i	IXL_Tmaxsafe_Supervision
IXLS2_i	IXL_Lsafemargin_Supervision
IXLD_i	IXL_Disconnect
IXLM_i	IXL_Mitigation
TCV_i	IXL_TCV
OBCC_i	OBC_CommOCC
OBCE_i	OBC_PL
OBCR_i	OBC_SendPosIXL
OBCD_i	OBC_Drive
OBCS_i	OBC_IXLConnectionSupervision
OBCM_i	OBC_Mitigation

To measure the probability of occurrence of, respectively, *HAZ*1 and *HAZ*2, the following properties are analysed (from now on, to avoid repetitions, properties are related to the corresponding hazards by their indexes):

$\phi_1 =$ `Pr(<>[0,bound]([][0,t]OCCC.Connecting && !failOBC[0]))`

$\phi_2 =$ `Pr[<= bound](OCCC.Connecting && ((OCCC.tid==0 && !OBCC_0.Connecting)))`

Basically, ϕ_1 measures the probability that (within `bound` time units) a configuration is reached where the module `OCC_Connecting` (whose instantiation is called `OCCC`) in state `Connecting` for exactly t time units. This will be the minimum amount of time needed for such hazard to occur. Note that (see Fig. 4) the automaton has reached such state only if it has received a connection request. Moreover, it is required that the system has not failed. Property ϕ_2 measures the probability to reach a configuration where the OCC is connecting a train `OCCC.Connecting` (with identifier 0 in this case `OCCC.tid==0`) but the corresponding tram is not connecting (`!OBCC_0.Connecting`). Note that this formula is instantiated to a specific tram (here `OBCC_0` is an instantiation of module `OBC_CommOCC`), and must be replicated for all trams considered in the experiments. To avoid repetitions, all properties verified in this model are related to measuring the probability of reaching a specific configuration within a given bound.

OCC RECEIVE POS Concerning the component for receiving the location of a train, we identified one hazard:

*HAZ*3: OCC has not received a position of one tram within `TmaxSafeOCC` time units.

$\phi_3 =$ `Pr(<> [0,bound] ([][0,TmaxSafeOCC] OCCconnectedCounter>0 && OCCR.WaitingLoc && (forall (tid:int[0,nTram-1]) !failOBC[tid])))`

Property ϕ_3 measures the probability that the OCC module for receiving the position (instantiated as `OCCR`) is in state `WaitingLoc` (see Fig. 5) and there is some connected tram (i.e. `OCCConnectedCounter>0`) for an amount of time `TmaxSafeOCC+1`. Moreover, no tram is in degraded mode (i.e. `(forall (tid:int[0,nTram-1]) !failOBC[tid])`).

OCC SUPERVISION One hazard has been identified for the supervision component of the OCC

*HAZ*4: all positions of connected trams are not received within `TmaxSafeOCC` but the OCC is not in failsafe mode.

ϕ_4=`Pr[<= bound](<> (OCCS.c1>OCCS.TmaxSafeOCC && OCCconnectedCounter!=0 && !OCCS.FailSafe))`

This property states that the maximum time in which a position must be received (i.e. `TmaxSafeOCC`) has been exceeded and there are connected trams (i.e. `OCCconnectedCounter!=0`) but the module has not entered a degraded mode (i.e. `!OCCS.FailSafe`).

OCC MITIGATION For the mitigation module, we have identified one hazard:

*HAZ*5: The OCC enters a degraded mode.

ϕ_5 = `Pr[<= bound](<> OCCM.DegradadedMode)`

The property measures the probability that the instantiation of OCC Mitigation module (`OCCM`) has reached a degraded mode.

7.2. Analysing the OBC

After having discussed the hazards related to the OCC, we move to the On-Board Computer hazards. Note that there could be different instantiations of the OBC. The property listed below are instantiated to a specific OBC and must be replicated (or quantified) in case of several instances of OBC.

OBC_VT Concerning the Virtual Tag reader module, one hazard has been identified:

*HAZ*6: the OBC has not checked if a virtual tag has been read within the given time interval.

ϕ_6 = `Pr(<> [0,bound]([][0,freq+1]OBCS_0.Disconnect&&OBCS_0.IXL!=-1&&!failOBC[0]))`

This property checks if the OBC Supervision module (instantiated as `OBCS_0`) remains in state `Disconnect` for at least `freq+1` consecutive time units. Indeed, each `freq` time units the virtual tag module shall send a read request to the supervision module (signal `read`) that shall react, meaning that the virtual tag has been correctly read (see Figs. 14, 15). Moreover, it is required that the system has not been disconnected already, i.e. `OBCS_0.IXL!=-1`, and it is not in degraded mode, i.e. `!failOBC[0]`.

OBC_IXL_ConnectionSupervision The OBC IXL Connection Supervision module, being one of the main modules of the OBC, has 5 hazards that are described below:

*HAZ*7: OBC connects to the corresponding IXL and the corresponding TV has not been read yet.
*HAZ*8: OBC is trying to connect for more than `TmaxConnect` time but the degraded mode has not been entered.
*HAZ*9: A TV has been read but the OBC is not connecting to the corresponding IXL.
*HAZ*10: The connection between IXL and OBC has been lost, but the tram has not entered a degraded mode.
*HAZ*11: The tram has entered a degraded mode but it has not stopped.

ϕ_7 = `Pr(<> [0,bound] ((Lv[0]+PL[0]+(l[0]/2) < TVa[0]) && OBCS_0.Connecting && !failOBC[0]))`
ϕ_8 = `Pr(<> [0,bound]([][0,TmaxConnect+1] OBCS_0.Connecting && OBCM_0.NormalOperation))`
ϕ_9 = `Pr(<> [0,bound] ([][0,t] (TVa[0] < Lv[0]-PL[0]-(l[0]/2) )&&`
`(OBCS_0.Disconnect || OBCS_0.CheckingTV)&& OBCS_0.IXL!=-1 ))`
ϕ_{10} = `Pr[<= bound](<> OBCS_0.ConnectionLost && OBCM_0.NormalOperation )`
ϕ_{11} = `Pr[<= bound] (<> OBCM_0.DegradadedMode && OBCD_0.Moving )`

Property ϕ_7 checks if a bad configuration is reached where the tag has not been read yet but the connecting state is reached in `OBCS_0` (instantiation of the supervision module). The event "the tag has not been read" is rendered as `(Lv[0]+PL[0]+(l[0]/2) < TVa[0])`, that correspond to stating that $LocationReferencingTagV$ evaluates to false. The system is required to be in normal operation. Property ϕ_8 checks whether OBC is trying to connect (is in state `OBCS_0.Connecting`) for `TmaxConnect+1` time units, but OBC mitigation module is still in normal mode (i.e. `OBCM_0.NormalOperation`).

Property ϕ_9 models the following scenario: the virtual tag has been read (the train has passed over it), but the train does not try to connect within a certain amount of time t (because connection does not happen instantaneously). This property is modeled as: for t consecutive time units (`[][0,t]`), `(TVa[0] < Lv[0]-PL[0]-(l[0]/2))`, i.e., the tag has been read. and the supervision module is still waiting to start a connection `(OBCS_0.Disconnect || OBCS_0.CheckingTV)`. The last condition `OBCS_0.IXL!=-1` is used to ensure that train has not already connected and disconnected from that IXL. Property ϕ_{10} checks whether `OBCS_0.ConnectionLost && OBCM_0.NormalOperation`, i.e. the connection has been lost by the supervision module but the mitigation module has not entered a degraded mode. Property ϕ_{11} checks if a configuration is reached where `OBCM_0.DegradadedMode && OBCD_0.Moving`, that is the mitigation module is in degraded mode, but the driving module (instantiated as `OBCD_0`) is not in a stopped state.

OBC_Mitigation The hazard for the mitigation module is similar to the one for the OCC.

*HAZ*12: the OBC has entered a degraded mode.

ϕ_{12}= `Pr[<= bound](<> (!OBCM_0.NormalOperation))`

This property measures the probability to reach a state in which normal operation conditions do not hold anymore (i.e. `!OBCM_0.NormalOperation`)

OBC_COMM_OCC The module of the OBC responsible of communicating with the OCC has two hazards:

*HAZ*13: the OBC has not connected to the OCC within `NmaxOCC` time units, but the OBC is not in degraded mode.

*HAZ*14: the OBC is connected to the OCC but has not sent its position within `NmaxPosOCC` time units and the OBC has not entered in a degraded mode.

ϕ_{13} = `Pr(<> [0,bound]([][0,NmaxOCC+1] OBCC_0.Connecting && OBCM_0.NormalOperation))`

ϕ_{14} = `Pr(<> [0,bound]([][0,NmaxPosOCC+1] OBCC_0.Connected && OBCM_0.NormalOperation))`

Property ϕ_{13} checks whether the model stays in state `OBCC_0.Connecting` for `NmaxOCC+1` consecutive time units and OBC is in normal operation condition mode (i.e. `OBCM_0.NormalOperation`). Here `OBCC_0` is the instantiation of the communication with OCC module. Property ϕ_{14} is similar to the previous one and it checks if after a given amount of failed send position attempts `NmaxPosOCC` (note that here it is in state `Connected` rather than `Connecting`), OBC is in normal operation condition mode.

7.3. Analysing the IXL module

We finally report the properties aimed at verifying the IXL module. Below we list for each automata composing the IXL the relevant properties. Similar to the OBC, there could be several instances of the IXL, one for each separate junction. Here we report the properties to verify a specific instance of the IXL.

IXL Connect There are two hazards related to the module responsible for connecting to trams.

*HAZ*15: IXL is not ready to serve a connection request within t time units.

*HAZ*16: IXL is connecting a tram that has not required to be connected.

ϕ_{15} = `Pr(<> [0,bound]([][0,t]!IXLC_0.Waiting))`

ϕ_{16} = `Pr[<=bound](<> (IXLC_0.Connecting && ((IXLC_0.tid==0 && !OBCS_0.Connecting))))`

The property ϕ_{15} expresses that the instantiation `IXLC` of the connection module is not in state `Waiting` for more than `t` time units. The property ϕ_{16} tests whether `IXLC` is in state `Connecting` but the corresponding tram (whose id is identified by `IXLC_0.tid==0`, tram 0 in this case) is not connecting (i.e. `!OBCS_0.Connecting`).

IXL Disconnect The module responsible of disconnecting a tram has three hazards:

*HAZ*17: IXL is disconnecting a tram that is not connected.

*HAZ*18: IXL is disconnecting a connected tram before it has traversed the last TCV involved in its route.

*HAZ*19: IXL is disconnecting a tram that has release (resp.occupied) the last VTC in a route whereas the release condition is occupied (resp.free).

ϕ_{17} = `Pr[Lv[0]<= spacebound](<> (IXLD_0.Disconnecting && ((IXLC_0.tid==0 && !OBCS_0.Connected))))`

ϕ_{18} = `Pr(<> [0,bound](IXLD_0.Disconnecting && (Lv[0]+PL[0]+l[0]/2<TCVa[1])))`

ϕ_{19} = `Pr[<= bound](<> ( ((IXLD_0.releaseOccupied==true)&&(IXLD_0.Disconnecting && TCV_1.Free)) || ((IXLD_0.releaseOccupied==false)&&(IXLD_0.Disconnecting && TCV_1.Occupied))))`

Similar to the previous module, these properties are instantiated on a specific scenario of one IXL and one tram. In case of different scenarios, to cover the hazards different instantiations of the properties are needed. The property ϕ_{17} is verified on simulations bounded by variable `Lv[0]` whose value must be less or equal to `spacebound`. It states that the instantiation `IXLD_0` of the disconnect module is in state `Disconnecting` but the tram (i.e. tram with id 0 in this scenario, i.e. `IXLC_0.tid==0`) is not connected (i.e. `!OBCS_0.Connected`). The

property ϕ_{18} states that the instantiation `IXLD_0` of the disconnect module is in state `Disconnecting` but the tram (i.e. tram with id 0 in this scenario) has not yet reached the last virtual track circuit involved in its route (VTC with identifier 1 in this scenario). This is rendered as `Lv[0]+PL[0]+l[0]/2<TCVa[1]`. The property ϕ_{19} follows the scenario of the previous properties. It is a disjunction of two sub-formulae differentiating whether `IXLD_0.releaseOccupied==true` or `IXLD_0.releaseOccupied==false` (i.e. the release condition is either true or false). The first case checks whether the corresponding virtual track circuit is free but the IXL is disconnecting the tram (it should be occupied). The second case is when the virtual track circuit is occupied but the IXL is disconnecting the tram (it should be free).

IXL Receive Pos This is the module of IXL in charge of receiving the positions sent by the trams. It has three hazards:

*HAZ*20: IXL has not received a position of a connected tram within the maximum allowed time (`maxfreq`).

*HAZ*21: IXL has checked a virtual track circuit not belonging to the junction of its responsibility.

*HAZ*22: IXL has not checked a virtual track circuit belonging to the junction of its responsibility.

ϕ_{20} = `Pr(<> [0,bound] ([][0,maxfreq+1] connectedCounter[0]>0 && IXLR_0.WaitingLoc && (forall (tid:int[0,nTram-1]) !failOBC[tid]))`

ϕ_{21} = `Pr[<= bound](<> IXLR_0.ReceivedLoc && (TCV_2.checkU1 || TCV_2.checkU2))`

ϕ_{22} = `Pr(<> [0,bound] (IXLR_0.ReceivedLoc && (TCV_0.checkU1 || TCV_0.checkU2) && (TCV_1.checkU1 || TCV_1.checkU2)))`

The property ϕ_{20} checks whether for `maxfreq` consecutive time units there are connected trams (i.e. the condition that is verified is `connectedCounter[0]>0`) but none of them has sent a position (i.e. `IXLR_0.WaitingLoc`) and all of them are not in degraded mode. Here `IXLR_0` is the instance of the receive position module. The property ϕ_{21} refers to a scenario where the only virtual track circuits of its responsibility are `TCV_0` and `TCV_1`, whilst `TCV_2` is not. Indeed, in this property the IXL has received a position (i.e. `IXLR_0.ReceivedLoc`), but `TCV_2` is being checked (see the module VTC below). The property ϕ_{22} is verifying the negated version of the corresponding hazards, that is, that all virtual track circuits under responsibility (i.e. `TCV_1` and `TCV_2`) are being verified (are either in state `checkU1` or in state `checkU2`). Since this is the negation of the hazard, this property must hold with probability closer to 1 (it is the only one in our analysis).

IXL TmaxSafe Supervision This is the module of the IXL responsible of monitoring whether all positions of connected trams are received within `TmaxSafe` time units. One hazard has been identified:

*HAZ*23: all connected tram positions are not received within a given amount of time but IXL has not entered a degraded mode.

ϕ_{23} = `Pr[<= bound](<> (IXLS_0.c1>IXLS_0.TmaxSafe && connectedCounter[0]!=0&&!IXLS_0.FailSafe))`

This property checks whether the maximum time limit has been exceeded (i.e. `IXLS_0.c1>IXLS_0.TmaxSafe`) and there are connected trams (i.e. `connectedCounter[0]!=0`) and the system is not in a degraded mode (i.e. `!IXLS_0.FailSafe`). Here, `IXLS_0` is the instantiation of this supervision module.

IXL Lsafemargin Supervision This is the dual IXL module of the position reception module. It is used for monitoring whether the protection level (`PL`) of the received position has exceeded its maximum allowed value (`Lsafemargin`).

*HAZ*24: the protection level maximum value has been exceeded but the system has not entered a degraded mode.

ϕ_{24}=`Pr(<>[0,bound]([][0,t]!IXLM_0.DegradadedMode&&connectedCounter[0]>0&&PL[0]>=Lsafemargin))`

The property ϕ_{24} checks whether within a sufficient amount of time (t time units) the IXL is not in degraded mode (i.e. `!IXLM_0.DegradadedMode`) and there are connected trams (i.e. `connectedCounter[0]>0`) and the maximum allowed value for the protection level has been exceeded (i.e. `PL[0]>=Lsafemargin`). Here, `IXLM_0` is the instantiation of this mitigation module and the property is instantiated for a scenario with one train.

IXL TCV This is the module related to the functionalities of the virtual track circuits monitored by an IXL.

*HAZ*25: a virtual track circuit is set to occupied but no tram is occupying it.

*HAZ*26: a virtual track circuit is set to free but at least a tram is occupying it.

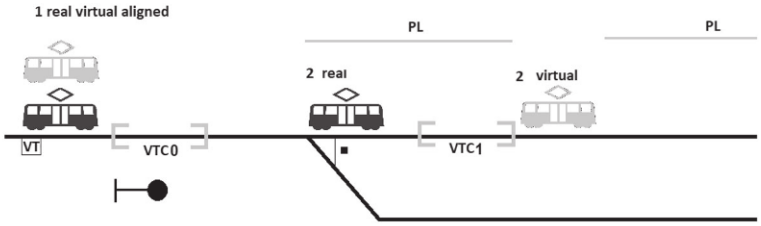

Fig. 23. A critical scenario where a tram is derailed if the protection level PL is ignored

ϕ_{25} = `Pr(<> [0,bound] ([][0,t] forall (tid:int[0,nTram-1]) !(TCV_0.a-PL[tid]-(l[tid]/2) <= TCV_0.pos && TCV_0.pos <= TCV_0.b+PL[tid]+(l[tid]/2) ) && TCV_0.Occupied))`

ϕ_{26} = `Pr(<> [0,bound] ([][0,t] exists (tid:int[0,nTram-1]) (TCV_0.a-PL[tid]-(l[tid]/2) <= TCV_0.pos && TCV_0.pos <= TCV_0.b+PL[tid]+(l[tid]/2)) && TCV_0.Free))`

The property ϕ_{25} checks whether the function $LocationReferencingVTC$ is false for all trams but the corresponding virtual track circuit is in state occupied. This property is expressed for `TCV_0` and is replicated for all virtual track circuits in the experiments. The property ϕ_{26} checks the converse situation: $LocationReferencingVTC$ is true for some tram but the virtual track circuit is free. Note that for both formulae it is required that the module remains in this configuration for t consecutive time units (i.e. to be sure that this is not due to short delays in communications). Of course, t can be fine-tuned to the specific configuration of the system.

IXL Mitigation The mitigation module is similar to those of the other entities in the system and it has one hazard:

$HAZ27$: the IXL has entered a degraded mode.

ϕ_{27} = `Pr[<= bound](<> IXLM_0.DegradadedMode)`

The property ϕ_{27} checks the probability for the IXL to enter a degraded mode.

7.4. Experiments

In this section we perform an evaluation of the properties described in the previous section, using different set-ups of parameters. The experiments were performed on a machine with a processor Intel(R) Core(TM) i7-8500Y CPU at 1.50GHz, 1601Mhz, 2 cores, and 4 logical processors with 16GB of RAM, running 64bit Windows 10. We used academic version of Uppaal 4.1.19 (rev. 5649), from September 2014. In all experiments, the probability of false negatives is $\alpha = 0.005$ and probability uncertainty is $\varepsilon = 0.05$. Moreover, when explicitly stated in Table 3 the probability uncertainty is set to $\varepsilon = 0.005$. For reproducibility of the experiments, all models with the different set-up of parameters, as well as logs of the outcomes of each experiment are available at [Bas21].

Scenario and set-up of experiments These experiments are performed on a specific scenario, depicted in Fig. 23. As stated in Sect. 3, this scenario is stemming from activities carried out jointly by the consortium members of the SISTER project. For a comprehensive verification, a full catalogue of possible scenarios complementing the one shown in Fig. 23 would need to be considered, and its completeness verified, we leave this as future work. In this scenario one junction area (i.e. one IXL) is composed of two VTCs, and there is one tram outside the junction area and one tram inside the junction area. Tram 2 is traversing its assigned route whilst Tram 1 is waiting at a red signal for its route to be assigned after reading the virtual tag VT. VTC 0 is used to detect the occupation of a route, whilst VTC 1 is used to detect the release of a route. VTC 2 belongs to another junction area, controlled by another IXL (not modelled in this scenario). It is assumed that a tram is disconnected instantaneously from the IXL when it has released its assigned route, hence the events route release and disconnection are coupled. Of course, there is only one OCC in the model. Moreover, we only instantiate IXL 0 (responsible of the considered junction area), and we only instantiate one OBC (with id 0) that is the OBC of Tram 2. Indeed, in our experiments Tram 1 is stuck waiting for the signal to open, and thus could only increase the state space without introducing any behaviour. Furthermore, the second IXL to whom VTC 2 belongs is out of reach by the trams and thus is not depicted in Fig. 23, and VTC 2 is only used to test ϕ_{21}.

Table 2. Set-up of Constant Integer Parameters

Constant	Value
freq	5
maxfreq	7
msgRate	2
Pmsg	$10^6 - 1$
failRate	2
restoreRate	3
TmaxSafeOCC	45
freqOCCconnection	15
NmaxOCC	45
NmaxPosOCC	15
PLrefreshRate	2
PLprob	$10^9 - 1$
minPL	0
maxPL	10
worstCasePL	20
TmaxAck	15
TmaxConnect	15
TmaxSafe	15
Lsafemargin	500

VTC 0 has interval `TCV_0.a=250`, `TCV_0.b=350` (m) and belongs to IXL 0. VTC 1 has interval `TCV_1.a=450`, `TCV_1.b=550` and belongs to IXL 0. VTC 2 has interval `TCV_2.a=650`, `TCV_2.b=750` and belongs to another IXL 1. Note that this junction area is modelled as rather short to improve verification speed. The condition for releasing the route, which are the parameters of instantiation of `IXL_Disconnect`, is `releaseOccupied=false`, and VTC 1 is the one to be traversed for releasing the route. The virtual tag for entering the junction area managed by the interlocking is placed in the interval `TV_0.a=100`, `TV_0.b=150`. We note that these values are experimental as no real implementation of this system exists yet.

The time upper bound is of 150 (i.e. `bound=150` s that is enough for the tram to cover 750 m (the space upper bound, i.e. `spacebound=750`) and traverse the whole junction area. In all the experiments discussed in this section, it will be assumed that trams are travelling at a constant speed of 5 m/s (18 km/h), i.e. `OBC_Drive.speed=5`, with length of 50 m (i.e. `l=50`). We note that the model accounts also for the possibility of having trams with different length and different speed.

First set-up The first set-up is summarised in Table 2, and is related to normal conditions, with no inflation. More importantly, the system is assumed to be working in nominal operation conditions in the sense of Stanford Diagram [ESA18, LBC+15] (i.e. the alert limit is considered to be greater than the protection level and the protection level greater than the position error), so excluding unavailability conditions and/or misleading information provision. Note that these are all constant parameters (apart from those described above), and they have been validated by our industrial partner. We refer to Sect. 6 for their meaning. Finally, the formulae introduced in the previous section are designed to detect hazards, hence their value is expected to be close to zero (with the only exception of ϕ_{22} whose value is expected to be close to one).

A summary of the results and performances of the experiments for all formulae described in the previous section, using parameters in Table 2, is reported in Table 3. Note that some formulae have been evaluated with a lower probability uncertainty ε to increase the number of simulations performed by the tool. All results meet the expectations, thus confirming that the model satisfies the given requirements. Recall that the probability is given as an interval (see Sect. 2).

Second set-up We now proceed by inflating some parameters to observe occurrence of rare events and mitigation in action. Indeed, in Table 2 it is reported that there is a low probability of losing messages and having worse values of protection level (10^{-6} and 10^{-9}). Since statistical model checking is not based on full state space exploration, with the current parameter set-up of the model checker such rare events are not observed. Instead of tuning the model checker parameters (which would result in an unfeasible number of simulations), we will simply proceed by making the rare events more probable. Moreover, to avoid disturbances, the system will not be restored once a degraded mode is entered. In particular, we proceed with `Pmsg=1,minPL=0,maxPL=100`. Thus, now the probability of losing messages is high and the values for PL are not reliable. Table 4 reports the experiments performed with this second "inflated" set-up.

Table 3. Evaluation of Properties and Performances (* evaluation performed with probability uncertainty $\varepsilon = 0.005$)

	Runs	Evaluation	Verification/kernel/elapsed time used	Resident/virtual memory usage peaks
ϕ_1	1199	[0, 0.0575063]	20.234 s/0.14 s/20.467 s	9160 KB/30,264 KB
ϕ_2	597*	[0, 0.00998576]	20.454 s/0.266 s/23.346 s	10,132 KB/32,212 KB
ϕ_3	1199	[0, 0.05]	20.125 s/0.953 s/26.373 s	9124 KB/30,156 KB
ϕ_4	597*	[0, 0.00998576]	24.625 s/0.672 s/30.981 s	9396 KB/30,704 KB
ϕ_5	597*	[0, 0.00998576]	23.765 s/0.25 s/24.122 s	10,596 KB/33,296 KB
ϕ_6	1199	[0, 0.05]	16.844 s/0.203 s/17.175 s	9936 KB/31,580 KB
ϕ_7	1199	[0, 0.05]	13.25 s/0.172 s/13.519 s	10,648 KB/33,004 KB
ϕ_8	1199	[0, 0.05]	12.625 s/0.234 s/14.211 s	11,408 KB/34,524 KB
ϕ_9	1199	[0, 0.05]	13.297 s/0.156 s/14.167 s	10,392 KB/32,764 KB
ϕ_{10}	597*	[0, 0.00998576]	5.234 s/0.172 s/5.874 s	11,524 KB/35,228 KB
ϕ_{11}	597*	[0, 0.00998576]	5.266 s/0.125 s/5.697 s	11,500 KB/35,164 KB
ϕ_{12}	597*	[0, 0.00998576]	4.219 s/0.172 s/5.316 s	10,560 KB/33,408 KB
ϕ_{13}	1199	[0, 0.05]	14 s/0.25 s/14.891 s	9996 KB/31,696 KB
ϕ_{14}	1199	[0, 0.05]	28.547 s/0.391 s/32.759 s	9192 KB/30,040 KB
ϕ_{15}	1199	[0, 0.05]	13.828 s/0.156 s/14.265 s	9076 KB/29,908 KB
ϕ_{16}	597*	[0, 0.00998576]	4.688 s/0.25 s/5.072 s	10,244 KB/32,332 KB
ϕ_{17}	597*	[0, 0.00998576]	2.875 s/0.203 s/3.087 s	10,080 KB/32,164 KB
ϕ_{18}	1199	[0, 0.05]	12.516 s/0.266 s/12.821 s	9012 KB/29,864 KB
ϕ_{19}	597*	[0, 0.00998576]	4.375 s/0.234 s/4.67 s	9276 KB/30,376 KB
ϕ_{20}	1199	[0, 0.05]	19.062 s/0.047 s/19.172 s	10,864 KB/33,684 KB
ϕ_{21}	597*	[0, 0.00998576]	4.5 s/0.234 s/4.751 s	11,964 KB/36,240 KB
ϕ_{22}	1199	[0, 95, 1]	1.234 s/0.188 s/1.43 s	9092 KB/30,136 KB
ϕ_{23}	597*	[0, 0.00998576]	4.547 s/0.156 s/4.696 s	10,408 KB/32,804 KB
ϕ_{24}	1199	[0, 0.05]	12.547 s/0.094 s/12.7 s	9072 KB/30,108 KB
ϕ_{25}	1199	[0, 0.05]	14.922 s/0.203 s/15.222 s	9064 KB/30,104 KB
ϕ_{26}	1199	[0, 0.05]	11.891 s/0.157 s/12.173 s	9976 KB/31,848 KB
ϕ_{27}	597*	[0, 0.00998576]	4.782 s/0.203 s/5.074 s	11,020 KB/34,120 KB

As expected now the probability of entering a degraded mode is high in all components of the model (i.e. properties ϕ_5, ϕ_{12} and ϕ_{27}), whilst all other properties, except for ϕ_7, are still within the values in Table 3.

We now show the functionalities provided by UPPAAL for analysing a trace on which the evaluated hazard occurs. Figure 24 displays the concrete simulator functionality of UPPAAL. Basically, once the property ϕ_7 has been verified, a trace is stored showing a simulation where ϕ_7 holds true. This is indeed one of the main advantages of model checking: the stored trace is composed of around 30 discrete steps (that are displayed in the bottom left box of Fig. 24). The generation of such trace is completely automatised, and no manual simulation is required, which would be very hard to perform considering the high number of involved steps and the high number of hazards to evaluate. The trace is also depicted using a Message Sequence Chart (MSC), and by clicking on a specific configuration on the MSC it is possible to see the values of the variables for that configuration. In particular, in the configuration in Fig. 24 (emphasised by a horizontal line in the MSC) the tram has position `Lv[0]=26`, protection level `PL[0]=74`, thus `OBCS_0` detects to have reached the first virtual tag, issues the connection request to the IXL, and moves to location `Connecting`. Continuing the analysis of the trace, we note that two consecutive connection requests communications fail, and `IXLC_0` remains in state `Waiting`. Finally, in Fig. 25, it is possible to see that after some step, a new protection level is computed with value 24. At this point, the hazard modelled by property ϕ_7 is detected to hold, because `Lv[0]+PL[0]+(l[0]/2) < TVa[0]` (i.e. the tram has not reached the virtual tag), `failOBC[0]=0` (i.e., the tram has not failed) and it is in state `Connecting`. Basically, this hazard was triggered by an inflated protection level leading the On-Board Computer to try to connect to the IXL before the virtual tag was reached. This is indeed a further evidence indicating how, although the property being a negation of the modelled logic should trivially not hold in the model, this is not the case due to the complexity of the interactions between various automata.

Continuing the analysis of this scenario, Fig. 26 displays the probability distribution of ϕ_7 for this experiment. Indeed, once the property has been verified, UPPAAL allows to visualize plots concerning the evaluated probability, for a fine-grained analysis. In the plot it is possible to observe that the peaks are at time 5 and 10, which are exactly the instants where `OBC_TV` emits a signal to read a virtual tag that is received by `OBC_IXLConnectionSupervision` instance. At time `TmaxConnect`, 15 s in the experiment, the system fails due to the high probability of message loss. We will comment more on these results after having shown the next experimental set-ups.

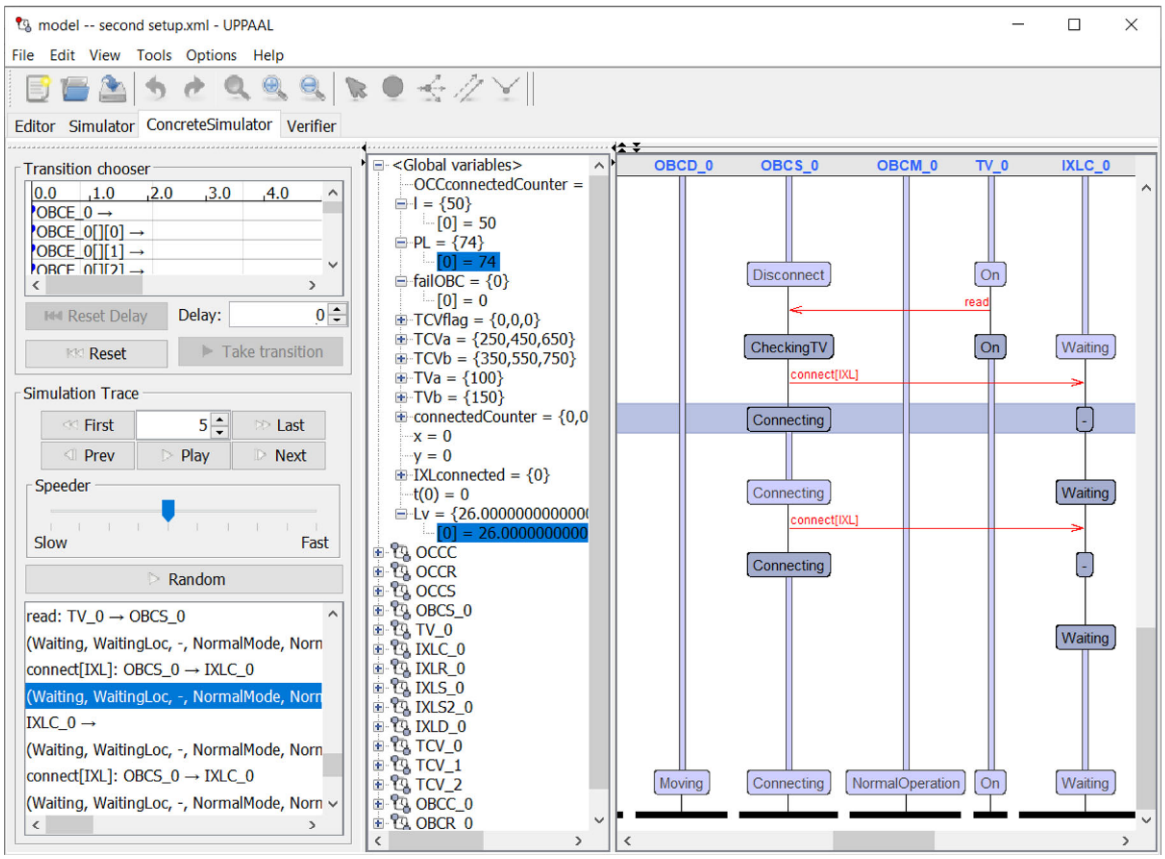

Fig. 24. Snapshot of UPPAAL showcasing a trace generated from the evaluation of ϕ_7 for the second set-up with both inflated protection level and probability of losing messages, in a configuration triggering the connection request of the tram

Table 4. Evaluation of Properties and Performances with inflated probability of losing messages and protection level

	Runs	Evaluation	Verification/kernel/elapsed time used	Resident/virtual memory usage peaks
ϕ_5	784	[0.529807, 0.629746]	3.703 s/0.203 s/3.969 s	12,324 KB/36,600 KB
ϕ_7	1199	[0.445413, 0.545413]	10.078 s/0.156 s/11.015 s	19,744 KB/51,616 KB
ϕ_{12}	57	[0.900222, 1]	0.125 s/0.062 s/0.193 s	12,392 KB/36,708 KB
ϕ_{27}	36	[0.902606, 1]	0.156 s/0.046 s/0.211 s	8760 KB/29,568 KB

Third set-up The result of the second set-up suggests that more hazards may occur in case only the protection level is degraded, whilst communications are reliable. Indeed, when there is a high probability of losing messages, the system will automatically enter a degraded mode after the timeout has expired, thus preventing to observe hazards that may occur after this expiration event. In the third set-up, only the protection level is inflated (as in the second set-up), whilst the probability of losing messages is low (as in the first set-up). Table 5 confirms our hypothesis: more hazards are observable by only inflating the protection level. In particular, properties ϕ_7, ϕ_{18}, ϕ_{25} and ϕ_{26} (and the corresponding hazards) now have a non-negligible probability of occurrence.

Fourth set-up We now change the condition in which a tram releases a route once it has traversed the junction. In particular, we set the condition for releasing the route to `releaseOccupied=true` (see Fig. 21), which is the tram is disconnected as soon as the last VTC involved in the route is occupied. In this fourth set-up we will only consider a slightly inflated protection level, i.e. `minPL=0,maxPL=40`, whilst the probability of losing messages is low. By changing the release condition we observe that the value of ϕ_{18} switches from a value similar to the one reported in Table 3 (with `releaseOccupied=false`) to being evaluated in the interval `[0.00921601,0.109216]` with confidence 0.995 using 1199 runs (with `releaseOccupied=true`).

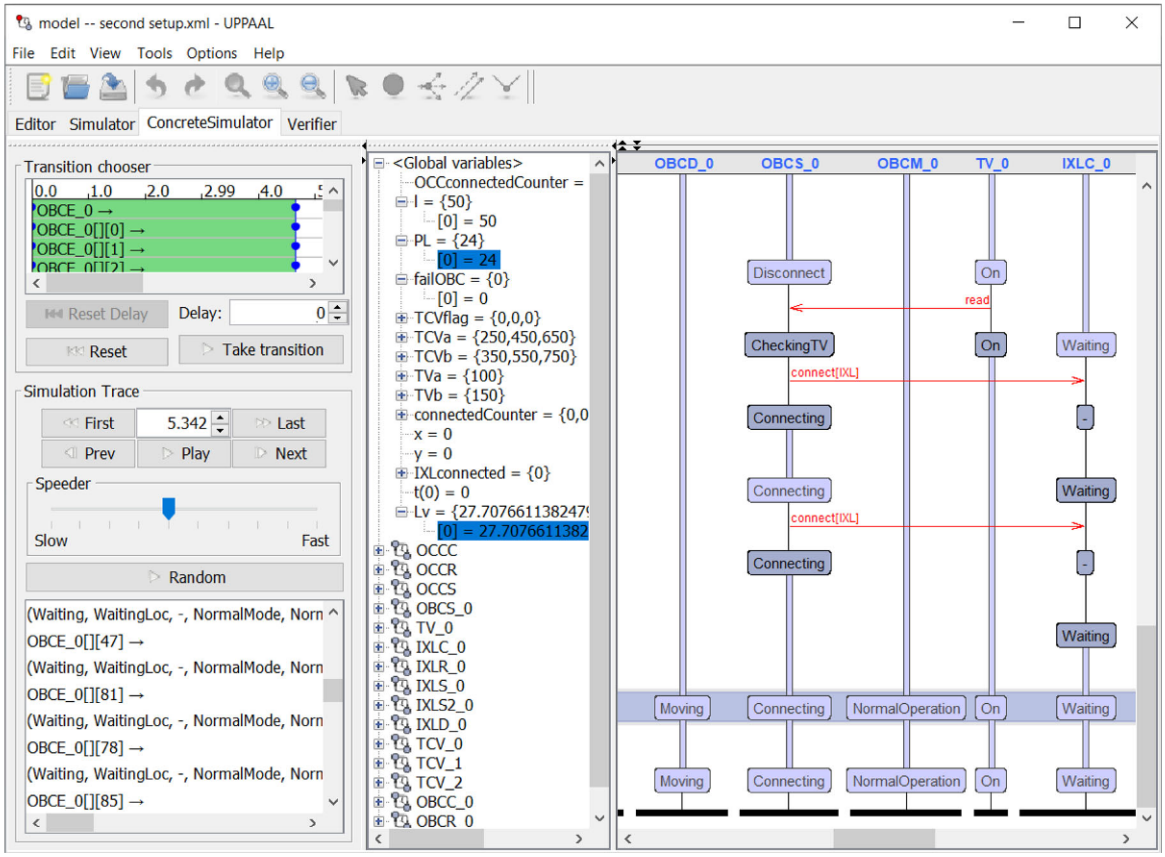

Fig. 25. Snapshot of UPPAAL showcasing a trace generated from the evaluation of ϕ_7 for the second set-up with both inflated protection level and probability of losing messages, in a configuration where ϕ_7 holds

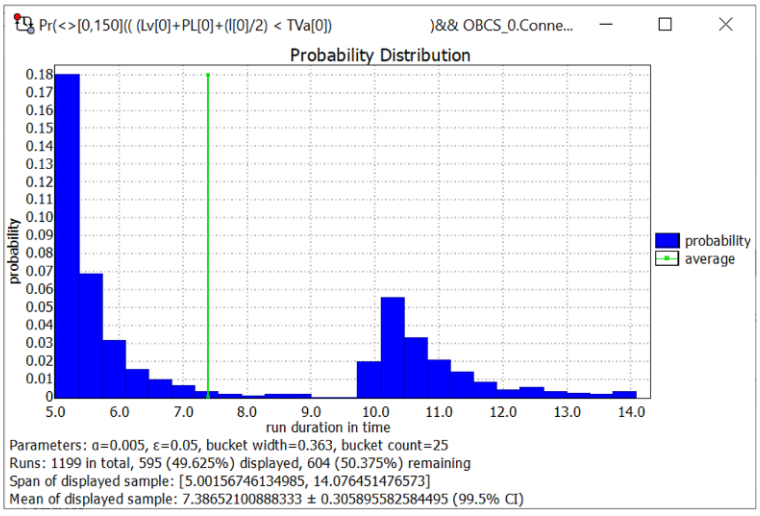

Fig. 26. The probability distribution for ϕ_7 in the degraded scenario

Table 5. Evaluation of Properties and Performances with inflated protection level only

	Runs	Evaluation	Verification/kernel/elapsed time used	Resident/virtual memory usage peaks
ϕ_7	1199	[0.248582, 0.348582]	17.625 s/1.062 s/22.148 s	19,488 KB/51,348 KB
ϕ_{18}	1199	[0.196872, 0.296872]	20.782 s/0.078 s/22.766 s	15,732 KB/49,136 KB
ϕ_{25}	1199	[0.00254379, 0.102544]	24.297 s/0.687 s/28.522 s	16,592 KB/50,868 KB
ϕ_{26}	1199	[0.0701001, 0.1701]	24.859 s/0.11 s/27.309 s	17,268 KB/52,264 KB

Hence, in the original specification `HAZ18` can happen when the release condition is set to occupied. Note that the violation of such hazard may result in a derailment. Indeed, in Fig. 23, once Tram 2 has been disconnected, the IXL will proceed to create the route to be assigned to Tram 1. Creating a route involves moving the switch (that Tram 2 is actually traversing) and setting signals. Since Tram 2 could be still traversing the junction, it could derail. The misalignment between physical position and virtual position is represented by depicting in grey the virtual positions of the trams and in black their real one. In particular, in Fig. 23 Tram 2 is behind its virtual position, but inside `PL`, a situation that would result in a derailment.

Fixing the specification An intuitive explanation of the causes of the hazards above follows: due to the introduction of `PL` in *LocationReferencing* (see Formula 1), the tram is "stretched" and "shrinked" in such a way that a virtual track circuit or virtual tag may be occupied and released even before the tram actually reaches it.

A first mitigation to the specification is to release a route only after the last VTC involved (in this case VTC 1) is completely traversed, and remove this scenario (i.e. `releaseOccupied` is always false). Note that this mitigation violates the assumption made by the initial specification (see Sect. 4), and in particular that the configuration parameters of the legacy system should remain unchanged in the new system. Indeed, all VTC must have release condition set to free. Concerning the hazards occurring when the `releaseOccupied` is always false, they are all related to a non-reliable protection level, i.e. in the interval `[0,100]` in our experiments, and we propose different alternative mitigations. First, occurrences of such hazards may be mitigated by pairing the satellite positioning with other positioning sensors, in such a way that the satellite position is ignored in case it is over a certain threshold (i.e. the alert limit). Indeed, when the protection level has exceeded the alert limit the position is ignored and a degraded mode is entered. In the first set-up, with a protection level in the interval `[0,10]` all properties were satisfied. A safe value for the alert limit in this example could be `10` (in previous experiments instead the alert limit was always greater than the protection level). A second alternative mitigation is to modify the specification of the location referencing function such that it also takes into account the direction of a tram, i.e. a virtual track circuit/virtual tag cannot be occupied and released from the same side. A third alternative mitigation is to substitute the alert limit with the protection level in the location referencing function (in automata `OBC_IXLConnectionSupervision` and `IXL_TCV`). Indeed, whilst the protection level `PL` varies dynamically (and this variation may potentially lead to hazards), the alert limit is fixed. Moreover, in case `PL` exceeds the alert limit the system will enter a degraded mode, where due operations will take place to restore normal conditions.

We have evaluated this third mitigation that has been implemented in the model using the third set-up (only inflated protection level). We used `100` as alert limit value. Moreover, `AL` has been substituted to `PL[0]` in properties ϕ_7, ϕ_{25} and ϕ_{26}. These three properties and ϕ_{18} are now evaluated to be close to zero, i.e., all hazards are successfully mitigated.

7.4.1. Remarks

The analysis described in this section has proved that the adoption of formal methods (in this particular case statistical model checking) may be of help for detecting in the early design phase of a system potentially critical bugs in the specification. By inflating the probability of losing connections and protection level we have been able to detect occurrences of some of the identified hazards. We also have evaluated a mitigation that has been proven to be effective.

It is important to note that a conventional model checking approach could not scale-up (as the state space of our model is very large) contrary to a statistical model checker that does not need to exhaustively visit the complete state space, but requires a probabilistic, stochastic model. Indeed, our aim was not a quantitative evaluation of dependability attributes of the proposed system but a qualitative, scalable verification. We used UPPAAL SMC as a fast tool to enlighten possible hazards in the design of a safety-critical system affected by uncertainty: the

obtained probability values for the hazards are not accurate because are obtained using inflated probabilities of occurrence for rare events such as communication failures. This was necessary to drive the statistical model checker towards those sensible scenario where hazards could be observed. However, a further, more accurate quantitative evaluation could be used to demonstrate that their probability of occurrence is well below any reasonable risk level, and for a possible refinement of the system specification. By simply using realistic set-ups and scenarios, our model can be used to provide a more costly reliable quantitative analysis of the measures of interest object of the analysis.

The adoption of a state-machine formalism has also been of help in sharing results with the industrial partners of this project. The automata modelling software components have a straightforward translation into state machines dialects used in Model-based Software Engineering tools, and can be ultimately refined to generate source code and tests for the pieces of software that have been modelled and analysed.

The studied hazards were related to the substitution of the track circuits with autonomous positioning, and did not include malfunctioning of legacy components, such as the IXL system, whose safety is considered already assessed. Indeed, the substitution principles studied on an example junction are actually quite independent from the junction itself and from its size. If deemed necessary, the experiments can be independently repeated for each junction of a line, with an execution time growing linearly with the number of junctions (e.g., let t be the worst time needed for verifying a junction, in case of k junctions one needs $k * t$ time to verify all of them). An approach similar to [GH21] for a network of distributed reconfigurable interlockings could also be adopted to separately verify each junction.

8. Related work

Uppaal Statistical Model Checker has been applied to various industrial case studies [LLN18], as for example [BFRM19, BtBFL19], and has also been used to verify the reliability of railway interlocking systems [CLS+17, HLF+20, LTH20], to verify railway timetables [HH19], and message protocols [LYZ20]. Uppaal Stratego, an extension of SMC with strategy synthesis and optimization, has been applied to a few other case studies belonging to the transport domain, such as traffic light controllers [TKR19], cruise control [LMT15], railway scheduling [KLFS19], autonomous driving [BtBL20] and navigation of maritime autonomous systems [SMVW20].

One of the main drawback of statistical model checking is the presence of rare events, i.e. events with small probability of occurrence. A technique called importance sampling [GI89] consists in modifying the model to reduce the variance of Monte Carlo estimator for rare event simulation. This technique has been applied to stochastic timed automata [JLL+16] for modifications performed on-the-fly during simulations; the algorithm in [BHP12] provides a confidence interval for the estimated property whilst [PFG18] uses a reinforcement learning technique for guiding simulations. Here we change the probability weights in the model to increase the occurrence of rare events (that are related to the possibility of loss of communications and interference in estimation of position), so to observe how the modeled mitigation behaves in presence of rare events, for a qualitative, quick verification.

Within the SISTER project [CBB+19], prior to this work, an Integer Linear Programming (ILP) model has been proposed for addressing route planning at run-time [BGG17], in the style of geographical interlocking systems [PKHP19]. Routes of all vehicles are guaranteeing safety against track disruptions and optimising the overall time-schedule. In the European Horizon 2020 Shift2Rail project ASTRail (SAtellite-based Signalling and Automation SysTems on Railways along with Formal Method and Moving Block Validation), Uppaal SMC has been used to model and analyse the ASTRail specification [BtBC18, BtBFL19]. Finally, other hybrid signalling systems using virtual fixed blocks with similarities to the one discussed in this paper have recently been modelled and analysed in [NGB+16, AKJ20, BLW18, CM20, MFTL20, DDPS20, HLK+20, BJL+18] with Promela/Spin, mCRL2, Electrum, and Event-B.

9. Conclusion and future work

In this paper a formal analysis of the system specification provided by the SISTER project has been carried out. The goal of the SISTER project is the substitution of physical positioning devices with virtual ones in tramway lines. Such devices are safety-critical, and the new system calls for a rigorous analysis with state-of-the-art techniques.

Starting from natural language requirements and operational scenarios provided by the industrial partners, we formalised the functioning of the SISTER system through stochastic timed automata and Uppaal SMC.

The result is a parametric formal model for a *generic application*, to be instantiated with parameters of a *specific application*. Such model is thus highly reusable and different operational scenarios can be evaluated.

Through the formal analysis it has been possible to detect problems in the original specification that have been fixed. The output of the analysis is a refined specification where proper mitigation are in place. Moreover, the new specification is described through a state-machine formalism, and it is ready to be used in the first stage of development of the system.

Future work This paper describes models of the informal specification that are partially translatable into software modules. We would like to explore the possibility of translating such models into some formalism amenable to automatic code generation. We are currently investigating the RT-UML and Papyrus [HDB17] framework and translation to UML-B and Event-B framework, in the style of [SMTVW20].

The approach in [AVW20] may be used in the future to generate code directly from the provided UPPAAL models, and to type-check the generated code using [KDPG16]. A further direction of research is investigating the possibility of having autonomously connected trains as described in [Fan19] and coupling our approach with tunnel control systems [OH19]. Finally, refining the automaton used for generating PL, which is now non-deterministic, into an automaton that computes PL starting from the error distribution φ_{0,σ^2} would allow to analyse realistic values of the alert limit. The synthesis capabilities of UPPAAL Stratego could be applied to automatically compute such alert limit to enforce safety whilst maximising the capacity of the railway network.

CRediT author statement D. Basile (first author): Conceptualization, Methodology, Writing - Original Draft, Software, Validation, Formal Analysis, Data Curation, Investigation. A. Fantechi: Writing - Review & Editing, Supervision, Project administration, Funding acquisition. L. Rucher: Project administration, Funding acquisition. G. Mandò: Project administration, Funding acquisition.

Acknowledgements

This work has been funded by the Tuscany Region project POR FESR 2014-2020 SISTER "SIgnaling & Sensing Technologies in Railway application".

References

[AKJ20] Arcaini P, Kofroň J, Ježek P (2020) Validation of the hybrid ERTMS/ETCS level 3 using SPIN. Int J Softw Tools Technol Transf 22(3):265–279

[AP18] Agha G, Palmskog K (2018) A survey of statistical model checking. ACM Trans Model Comput Simul 28(1):6:1–6:39

[AVW20] Arenis SF, Vujinovic M, Westphal B (2020) On implementable timed automata. In: Formal techniques for distributed objects, components, and systems—40th IFIP WG 6.1 international conference, FORTE 2020, held as part of the 15th international federated conference on distributed computing techniques, DisCoTec 2020, Valletta, Malta, June 15–19, 2020, proceedings, pp 78–95

[Bas21] Basile D (2021) Repository for reproducing the experiments. https://github.com/davidebasile/faoc2020

[BDL+06] Behrmann G, David A, Larsen KG, Håkansson J, Pettersson P, Yi W, Hendriks M (2006) UPPAAL 4.0. In: Proceedings of the 3rd international conference on the quantitative evaluation of sysTems (QEST'06). IEEE, pp 125–126

[BDL+13] Bulychev P, David A, Larsen KG, Legay A, Li G, Poulsen DB (2013) Rewrite-based statistical model checking of WMTL. In: Qadeer S, Tasiran S (eds) Runtime verification—revised selected papers of the 3rd international conference on runtime verification (RV'12), volume 7687 of lecture notes in computer science. Springer, pp 260–275

[BFRM19] Basile D, Fantechi A, Rucher L, Mandò G (2019) Statistical model checking of hazards in an autonomous tramway positioning system. In: Reliability, safety, and security of railway systems. Modelling, analysis, verification, and certification—third international conference, RSSRail 2019, Lille, France, June 4–6, 2019, proceedings, pp 41–58

[BGG17] Basile D, Di Giandomenico F, Gnesi S (2017) Dependable dynamic routing for urban transport systems through integer linear programming. In: Fantechi A, Lecomte T, Romanovsky AB (eds) Reliability, safety, and security of railway systems. Modelling, analysis, verification, and certification—second international conference, RSSRail 2017, Pistoia, Italy, November 14–16, 2017, proceedings, volume 10598 of lecture notes in computer science. Springer, pp 221–237

[BHP12] Barbot B, Haddad S, Picaronny C (2012) Coupling and importance sampling for statistical model checking. In: Flanagan C, König B (eds) Tools and algorithms for the construction and analysis of systems, volume 7214 of LNCS. Springer, pp 331–346

[BJL+18] Berger U, James P, Lawrence A, Roggenbach M, Seisenberger M (2018) Verification of the European rail traffic management system in real-time maude. Sci Comput Program 154:61–88

[BLW18] Bartholomeus M, Luttik B, Willemse T (2018) Modelling 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'18), volume 11119 of LNCS. Springer

[BM12] Beugin J, Marais J (2012) Simulation-based evaluation of dependability and safety properties of satellite technologies for railway localization. Transp Res C-Emerg 22:42–57

[Bou15] Boulanger JL (2015) Tool qualification. In: CENELEC 50128 and IEC 62279 Standards, chap. 9. Wiley, pp 287–308

[BtBC18] Basile D, ter Beek MH, Ciancia V (2018) Statistical model checking of a moving block railway signalling scenario with UPPAAL SMC—experience and outlook. In: Leveraging applications of formal methods, verification and validation. Verification—8th international symposium, ISoLA 2018, Limassol, Cyprus, November 5–9, 2018, proceedings, Part II, pp 372–391

[BtBFL19] Basile D, ter Beek MH, Ferrari A, Legay A (2019) Modelling and analysing ERTMS L3 moving block railway signalling with simulink and UPPAAL SMC. In: Formal methods for industrial critical systems—24th international conference, FMICS 2019, Amsterdam, The Netherlands, August 30–31, 2019, proceedings, pp 1–21

[BtBL20] Basile D, ter Beek MH, Legay A (2020) Strategy synthesis for autonomous driving in a moving block railway system with UPPAAL stratego. In: Formal techniques for distributed objects, components, and systems—40th IFIP WG 6.1 international conference, FORTE 2020, held as part of the 15th international federated conference on distributed computing techniques, DisCoTec 2020, Valletta, Malta, June 15–19, 2020, proceedings, pp 3–21

[CBB+19] Ceccarelli A, Basile D, Bondavalli A, Falai L, Fantechi A, Ferrari S, Mandò G, Nostro N, Rucher L (2019) The SISTER approach for verification and validation: a lightweight process for reusable results. In: Computer safety, reliability, and security—SAFECOMP 2019 workshops, ASSURE, DECSoS, SASSUR, STRIVE, and WAISE, Turku, Finland, September 10, 2019, proceedings, pp 185–197

[CLS+17] Cappart Q, Limbrée C, Schaus P, Quilbeuf J, Traonouez L-M, Legay A (2017) Verification of interlocking systems using statistical model checking. In: HASE. IEEE, pp 61–68

[CM20] Cunha A, Macedo N (2020) Validating the hybrid ERTMS/ETCS level 3 concept with Electrum. Int J Softw Tools Technol Transf 22(3):281–296

[DDPS20] Dghaym D, Dalvandi M, Poppleton M, Snook C (2020) Formalising the hybrid ERTMS level 3 specification in iUML-B and Event-B. Int J Softw Tools Technol Transf 22(3):297–313

[DLL+15] David A, Larsen KG, Legay A, Mikučionis M, Poulsen DB (2015) UPPAAL SMC tutorial. Int J Softw Tools Technol Transf 17(4):397–415

[E+15] Ericson CA et al (2015) Hazard analysis techniques for system safety. Wiley, Hoboken

[ERAMK18] El-Rahman S, Attiya A, Mamoud H, Kader H (2018) Passive rfid tag for railway application. Open J Antennas Propag 06:15–24

[ESA18] ESA (2018) Navipedia - Integrity. https://gssc.esa.int/navipedia/index.php/Integrity#Protection_Level, page last edited July

[Eur10] European Committee for Electrotechnical Standardization (2010) CENELEC EN 50159—railway applications—communication, signalling and processing systems—safety-related communication in transmission systems

[Eur11] European Committee for Electrotechnical Standardization (2011) CENELEC EN 50128—railway applications—communication, signalling and processing systems—Software for railway control and protection systems, 01 06

[Eur17a] European Committee for Electrotechnical Standardization (2017) CENELEC EN 50126-1—railway applications—the specification and demonstration of reliability, availability, maintainability and safety (RAMS)—part 1: generic RAMS process, 01 10

[Eur17b] European Committee for Electrotechnical Standardization (2017) CENELEC EN 50126-2—-Railway applications—the specification and demonstration of reliability, availability, maintainability and safety (RAMS)—part 2: systems approach to safety, 01 10

[Fan19] Fantechi A (2019) Connected or autonomous trains? In: Reliability, safety, and security of railway systems. Modelling, analysis, verification, and certification—third international conference, RSSRail 2019, Lille, France, June 4–6, 2019, proceedings, pp 3–19

[FNF18] Falco G, Nicola M, Falletti E (2018) 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). Institute of Navigation, pp 3463–3477

[GH21] Geisler S, Haxthausen AE (2021) Stepwise development and model checking of a distributed interlocking system using raise. Form Asp Comput 33(1):87–125

[GI89] Glynn PW, Iglehart DL (1989) Importance sampling for stochastic simulations. Manag Sci 35(11):1367–1392

[GJRS13] Groves PD, Jiang Z, Rudi M, Strode P (2013) 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). Institute of Navigation, pp 3231–3247

[GtBvdP20] Garavel H, ter Beek MH, van de Pol J (2020) The 2020 expert survey on formal methods. In: ter Beek MH, Ničković D (eds) Proceedings of the 25th international conference on formal methods for industrial critical systems (FMICS'20), volume 12327 of LNCS Springer, pp 3–69

[HDB17] Hili N, Dingel J, Beaulieu A (2017) Modelling and code generation for real-time embedded systems with uml-rt and papyrus-rt. In: Proceedings of the 39th international conference on software engineering companion, ICSE-C '17. IEEE Press, pp 509–510

[HEPGP21] Tomas Hotzel E, Ken P, Golightly D, Palacin R (2021) Modelling train driver behaviour in railway co-simulations. In Cleophas L, Massink M (eds) Software engineering and formal methods. SEFM 2020 collocated workshops. Springer International Publishing, Cham, pp 249–262

[HH19] Haxthausen AE, Hede K (2019) Formal verification of railway timetables - using the UPPAAL Model Checker. In: From software engineering to formal methods and tools, and back, volume 11865 of LNCS, pp 433–448

[HLF+20] Huang J, Lv J, Feng Y, Luo Z, Liu H, Chai M (2020) A novel method on probability evaluation of zc handover scenario based on smc. In: Qian J, Liu H, Cao J, Zhou D (eds) Robotics and rehabilitation intelligence. Springer Singapore, Singapore, pp 319–333

[HLK+20] Hansen D, Leuschel M, Körner P, Krings S, Naulin T, Nayeri N, Schneider D, Skowron F (2020) 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

[JLL+16] Jegourel C, Larsen KG, Legay A, Mikučionis M, Poulsen DB, Sedwards S (2016) Importance sampling for stochastic timed automata. In: Fränzle M, Kapur D, Zhan N (eds) Dependable software engineering: theories, tools, and applications. Springer International Publishing, Cham, pp 163–178

[KDPG16] Kouzapas D, Dardha O, Perera R, Gay SJ (2016) Typechecking protocols with mungo and stmungo. In: Proceedings of the 18th international symposium on principles and practice of declarative programming, PPDP '16. Association for Computing Machinery, New York, NY, USA, pp 146–159

[KLFS19] Karra SL, Larsen KG, FL, Srba J (2019) Safe and time-optimal control for railway games. In: RSSRail, volume 11495 of LNCS, pp 106–122

[LBC+15] Legrand C, Beugin J, Conrard B, Marais J, Berbineau M, El-Miloudi E (2015) Approach for evaluating the safety of a satellite-based train localisation system through the extended integrity concept. In: Proceedings of ESREL 2015—European safety and reliability conference

[LLN18] Larsen KG, Lorber F, Nielsen B (2018) 20 years of UPPAAL enabled industrial model-based validation and beyond. In: Leveraging applications of formal methods, verification and validation. industrial practice—8th international symposium, ISoLA 2018, Limassol, Cyprus, November 5–9, 2018, proceedings, Part IV, pp 212–229

[LLT+19] Legay A, Lukina A, Traonouez LM, Yang J, Smolka SA, Grosu R (2019) Statistical model checking. In: Steffen B, Woeginger GJ (eds) Computing and software science: state of the art and perspectives, volume 10000 of LNCS. Springer, pp 478–504

[LMT15] Larsen KH, Mikucionis M, Taankvist JH (2015) Safe and optimal adaptive cruise control. In: Correct system design, volume 9360 of LNCS, pp 260–277

[LTH20] Laursen PL, Trinh VAT, Haxthausen AE (2020) Formal modelling and verification of a distributed railway interlocking system using uppaal. In: Margaria T, Steffen B (eds) Leveraging applications of formal methods, verification and validation: applications. Springer International Publishing, Cham, pp 415–433

[LYZ20] Li R, Yin J, Zhu H (2020) Modeling and analysis of rabbitmq using uppaal. In: 020 IEEE 19th international conference on trust, security and privacy in computing and communications (TrustCom), pp 79–86

[MFTL20] Mammar A, Frappier M, Tueno Fotso SJ, Laleau R (2020) A formal refinement-based analysis of the hybrid ERTMS/ETCS level 3 standard. Int J Softw Tools Technol Transf 22(3):333–347

[NGB+16] Nardone R, Gentile U, Benerecetti M, Peron A, Vittorini V, Marrone S, Mazzocca N (2016) Modeling railway control systems in Promela. In: Artho C, Ölveczky PC (eds) Formal techniques for safety-critical systems—revised selected papers of the 4th international workshop on formal techniques for safety-critical systems (FTSCS'15), volume 596 of communications in computer and information science. Springer, pp 121–136

[OH19] Oortwijn W, Huisman M (2019) Formal verification of an industrial safety-critical traffic tunnel control system. In: Ahrendt W, Tapia Tarifa SL (eds) Integrated formal methods. Springer International Publishing, pp 418–436

[PFG18] Puch S, Fränzle M, Gerwinn S (2018) Quantitative risk assessment of safety-critical systems via guided simulation for rare events. In: Margaria T, Steffen B (eds) Leveraging applications of formal methods, verification and validation. Verification. Springer International Publishing, Cham, pp 305–321

[PKHP19] Peleska J, Krafczyk N, Haxthausen AE, Pinger R (2019) Efficient data validation for geographical interlocking systems. In: Reliability, safety, and security of railway systems. Modelling, analysis, verification, and certification—third international conference, RSSRail 2019, Lille, France, June 4–6, 2019, proceedings, pp 142–158

[RCN+13] Rispoli F, Castorina M, Neri A, Filip A, Di Mambro G, Senesi F (2013) Recent progress in application of GNSS and advanced communications for railway signaling. In: Proceedings of the 23rd international conference radioelektronika (RADIOELEKTRONIKA 2013). IEEE, pp 13–22

[Shi15] Shift2Rail Joint Undertaking (2015) Multi-Annual Action Plan, 26 November. http://ec.europa.eu/research/participants/data/ref/h2020/other/wp/jtis/h2020-maap-shift2rail_en.pdf.

[SMTVW20] Shokri-Manninen F, Tsiopoulos L, Vain J, Waldén M (2020) Integration of iUML-B and UPPAAL timed automata for development of real-time systems with concurrent processes. In: Raschke A, Méry D, Houdek F (eds) Rigorous state-based methods. Springer International Publishing, pp 186–202

[SMVW20] Shokri-Manninen F, Vain J, Waldén M (2020) Formal verification of colreg-based navigation of maritime autonomous systems. In: de Boer F, Cerone A (eds) Software engineering and formal methods. Springer International Publishing, Cham, pp 41–59

[tBGK18] ter Beek MH, Gnesi S, Knapp A (2018) Formal methods for transport systems. Int J Softw Tools Technol Transf 20(3)

[TKR19] Thamilselvam B, Kalyanasundaram S, Panduranga Rao MV (2019) Coordinated intelligent traffic lights using UPPAAL stratego. In: COMSNETS. IEEE, pp 789–794

Received 7 August 2020

Accepted in revised form 9 June 2021 by Jim Woodcock, Alessandro Fantechi and Anne Haxthausen