Formal Analysis of the UNISIG Safety Application Intermediate Sub-Layer

Applying Formal Methods to Railway Standard Interfaces

Davide Basile[0000−0002−7196−6609][1], Alessandro Fantechi[0000−0002−2930−6367][1,2], and Irene Rosadi[2]

[1] ISTI–CNR, Pisa, Italy
[2] University of Florence, Florence, Italy

Abstract. The combined use of standard interfaces and formal methods is currently under investigation by Shift2Rail, a joint undertaking between railway stakeholders and the EU. Standard interfaces are useful to increase market competition and standardization whilst reducing long-term life cycle costs. Formal methods are needed to achieve interoperability and safety of standard interfaces and are one of the targets of the 4SECURail project funded by Shift2Rail. This paper presents the modelling and analysis of the selected case study of the 4SECURail project: the Safe Application Intermediate sub-layer of the UNISIG RBC/RBC Safe Communication Interface. The adopted formal method is Statistical Model Checking of a network of Stochastic Priced Timed Automata, as provided by the Uppaal SMC tool. The main contributions are: (i) rigorous complete and publicly available models of an official interface specification already in operation, (ii) identification of safety and interoperability issues in the original specification using Statistical Model Checking, (iii) quantification of costs for learning the adopted formal method and developing the carried out analysis.

1 Introduction

Despite the large number of successful applications of formal methods in the railway domain [14], no universally accepted technology has emerged. Indeed, if applicable standards (e.g. CENELEC EN 50128 for the development of software for railway control and protection systems) mention formal methods as highly recommended practices [13], they do not provide clear guidelines on how to use them in a cost-effective way. The absence of a clear idea of which benefits can result from the adoption of formal methods is one of the aspects that act as an obstacle to the widespread use of formal methods [16,17]. This is witnessed also by the current efforts undertaken by Shift2Rail.

The Shift2Rail Joint Undertaking was established in 2014 under Horizon 2020 R&I program for pursuing research and innovation activities in the railway domain. As mentioned in the technology demonstrator TD2.7 "Formal methods and standardisation for smart signalling systems", Shift2Rail "has identified the

use of formal methods and standard interfaces as two key concepts to enable reducing the time it takes to develop and deliver railway signalling systems, and to reduce costs for procurement, development and maintenance. Formal methods are needed to ensure correct behaviour, interoperability and safety, and standard interfaces are needed to increase market competition and standardization, reducing long-term life cycle costs" [24]. One of the two workstreams of the 4SECURail (FORmal Methods and CSIRT for the RAILway sector) project deals with investigating the benefits of a formal method approach to the specification of standard interfaces. Moreover, 4SECURail aims to perform a costs and benefits analysis for the adoption of formal methods in the railway environment.

In this paper we present recent efforts in the context of the 4SECURail project. We present the formal modelling and analysis of the selected case study of the project that is the UNISIG Subset-098 - RBC/RBC (Radio Block Centre) Safe Communication Interface [25], and in particular the Safe Application Intermediate (SAI) sub-layer, concerning the protection against specific threats identified by CENELEC standards [12]. We exploit formal methods to build a fully defined mathematical model, in particular a network of Stochastic Priced Timed Automata. We verify the specified protection mechanisms against safety requirements identified by CENELEC standards, that are formalized using temporal logics. Our model enhances the existing standard interface with unambiguous, fine-grained modelling of the natural language requirements. The model can be exploited as starting point for other model-based activities such as model-based development or model-based testing. The identified benefits are also represented by a series of issues that emerged from the formal verification, mostly due to undefined or ambiguous aspects, tampering both safety and interoperability of the system. We also traced the costs in terms of man-hours needed to learn formal methods and to develop the model and analysis presented. Such data can be validated against publicly available documents and regulations [27]. All models and logs of experiments are publicly available at [22].

Related work A subset of authors have experience in applying Uppaal to study the upcoming ERTMS/ETCS Level 3 specification in the context of the Shift2Rail ASTRail project in [2,3,5,6]. Whilst those papers are exploring new requirements of an envisioned system, here we verify an official specification (dated 2012) already realised. Concerning the Shift2Rail 4SECURail project, in [4] the design of a formal methods demonstrator is discussed, which is based upon behavioural UML models. Here we provide a further contribution by adopting Uppaal SMC. In [19] the handover in Communication Based Train Control systems is analysed with Uppaal SMC and a novel method of probability evaluation. Similar to our work, they consider probabilistic failures (with probability weights set to 10^{-5}), assuming the presence of probabilistic communication failures. They analyse the scenarios of handover request and border point crossing from the front and rear end of a train, to show that the overall probability of a successful handover is high (0.99985). In [9] the RBC/RBC handover is analysed using run-time monitoring algorithms enforcing modal sequence charts. They report a concrete accident scenario where two trains collided. The accident was due to incorrect

interactions between interlocking and RBC, where basically the trains movement authority were independently generated by both the RBC (using Level 3) and the track circuits (using Level 2), thus unfortunately allocating the same track portion to two different trains. Compared to these papers, we study the current ERTMS/ETCS Level 2 handover. We do not focus on the exchange of application messages (e.g. crossing the border), but on a lower level, ensuring protection against threats due to open radio communications. Moreover, we do not focus on providing a precise evaluation of the probability of the handover to fail, nor to provide run-time monitors. We provide a qualitative analysis of the requirements in [25], identifying both safety and interoperability issues in the current operating specification. We use SMC to scale to the real-world case study size of our model, and probabilities of communications errors are inflated to drive the simulations toward faulty scenarios to analyse the protections.

Structure of the paper Uppaal SMC and the case study are briefly introduced in Section 2. The model and the analysis are in, respectively, Section 3 and Section 4. Section 5 concludes the paper.

2 Background

Statistical Model Checking and Uppaal Statistical Model Checking [1,20] (SMC) is concerned with running a controlled number of (probabilistically distributed) simulations of a system model to obtain a statistical evaluation (with a predefined level of statistical confidence) of some formula φ. The Monte Carlo estimation with Chernoff-Hoeffding bound executes $N = \lceil (ln(2) - ln(\alpha))/(2\epsilon^2) \rceil$ simulations ρ_i, $i \in 1...N$, to provide the interval $[p' - \epsilon, p' + \epsilon]$ with confidence $1 - \alpha$, where $p' = (\#\{\rho_i \,|\, \rho_i \models \varphi\})/N$, i.e., $\texttt{Pr}(|p' - p| \leq \epsilon) \geq 1 - \alpha$ where p is the unknown value of φ being estimated statistically [20]. 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. Indeed, the parameter N is *independent* from the size of the state-space. Moreover, the required simulations can easily run in parallel. This comes at a price: contrary to exhaustive model checking, exact results are out of reach, especially for formulae evaluated with very low probability, called rare events. 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 [11] extends Uppaal [7], a well-known toolbox for the verification of real-time systems modelled by (extended) timed automata. Uppaal SMC models are network of Stochastic Priced 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. Uppaal SMC allows to check (quantitative)

properties over simulation runs of a Uppaal SMC model. These properties must be expressed in a dialect of the Metric Interval Temporal Logic (MITL) [8]. In particular, all formulae φ_i evaluated in Section 4 follow a specific form, which is the probability that the configuration identified by the propositional formula `conf` is reached before `bound` units of time, written `Pr[<=bound](<>conf)`.

RBC/RBC Safe Communication Interface The selected case study is the RBC/RBC handover interface as specified by UNISIG Subset-098 – RBC/RBC Safe Communication Interface in [25], which provides a public standardized interface that specifies the requirements for the handover protocol between neighbouring RBCs in natural language.

Each RBC supervises all the trains moving within its responsibility area. The handover procedure is used to manage the interchange of train supervision between two neighbouring RBCs. This protocol is based on a layered structure. The higher layer corresponds to an application process that addresses high-level functionalities, such as the generation and the reception of information to communicate with peer RBC entities, or the re-establishment of the safe connection when it is lost due to errors in lower layers. This layer communicates with an underlying layer, the Safety Functional Module (SFM) which specifies the requirements related to the safety of the communications. The SFM layer consists of two distinct sub-layers, the SAI (Safe Application Intermediate) sub-layer and the Euroradio SL (Euroradio Safety Layer), and their combination provides a safe protection strategy for the open transmission system. The SAI layer provides adequate protection against the threats identified by CENELEC and specified in the EN 50159 European Standard [12], specifically: repetition (a message already sent is sent again in the message stream); deletion (a message is removed from the message stream); insertion (an additional message is implanted in the message stream) and re-sequencing (the ordering of messages in a stream is changed). The Euroradio SL protects the system against corruption, masquerade and insertion threats. The SAI sub-layer protection is achieved with a sequence number for deletion, re-sequencing and repetition threats. Basically, it consists of inserting a consecutive number to each message and computing the difference between such numbers. The delay defence technique is achieved with the TTS (Triple Time Stamp) procedure, consisting in storing in each message three timestamps information for checking that the transmission delay is within a computed offset. An alternative delay defense technique in [25], namely, the execution cycle, is not addressed in our model. Due to lack of space, we refer to [12, 20, 25] for more details on, respectively, SMC, Subset-098 and EN 50159.

3 The model

We now discuss the model of the case study. Due to lack of space, some aspects will not be detailed. The model is defined through template automata. Each template automaton may have a set of parameters and local declarations of constants, variables, user functions and clocks. Global declarations are instead accessible from all the templates and can include clocks, constants, variables,

functions and channels. The templates are parameterised with an identifier `id` to identify which device each template belongs to, i.e. the Initiator or the Responder device. The system is defined as a network of processes that interact with each other in parallel; a process is instantiated from a template where all its parameters, if any, are set. The synchronizations between different processes of the system are obtained through broadcast channels, which are required to perform statistical model checking. All the safety service primitives specified in the requirements are modelled as arrays of channels where the indexing allows to identify the synchronized process. Since Uppaal channel synchronization does not support value passing, this is encoded with the use of global shared variables. State invariants are used to ensure that the communications through shared variables used for the value-passing are atomic. This implies that signals are always received and never lost, overcoming the undetected loss of messages.

The system is composed of two communicating devices, an Initiator device that sends the request to establish a connection and a Responder device that receives the connection request. When referring to the partner device, we consider the Responder device as the partner of the Initiator device, and vice versa. In Figure 1, the overall architecture of the system is shown. Only one component (i.e. the initiator or the responder) is displayed whilst the other is specular. Each device is modelled using three modules: the SAI User, the SAI and the Euroradio SL modules. The SAI User and SAI modules are adjacent and can communicate with each other. The same applies also to the SAI and Euroradio SL modules. Both the Initiator and the Responder devices are composed of all these three modules. Finally, the Euroradio SL modules of both devices can receive failure notifications from the Communication System module, a component of the system that abstracts both the Euroradio SL lower layers and the physical transmission system. In particular, this component models the occurrence of a disruptive connection release communicated to both the Initiator and the Responder devices, as specified in the requirements.

The communications between adjacent modules of the same device are modelled using channels synchronizations. Instead, the two partner devices interact asynchronously using two queues of messages. Their interactions are affected by stochastic delays, simulating the transmission delays that characterize the radio communications. Uppaal does not natively support asynchronous communication through queues, which are implemented in the model using arrays. Probabilistic failures are used to simulate communication errors and are implemented by functions modifying data in the arrays (e.g., removing, swapping elements). These injected faults are not to be confused with the low-level channels synchronizations that are guaranteed by invariants to be received.

While both the SAI User module and the Euroradio module functionalities are implemented through single templates, the SAI module is split into multiple sub-modules to reduce the complexity of each of them. Indeed, the responsibilities of both the SAI User and the Euroradio modules were abstracted away and only the interface with the SAI is implemented. They are the external entities interacting with the SAI module target of our model, whose functionalities

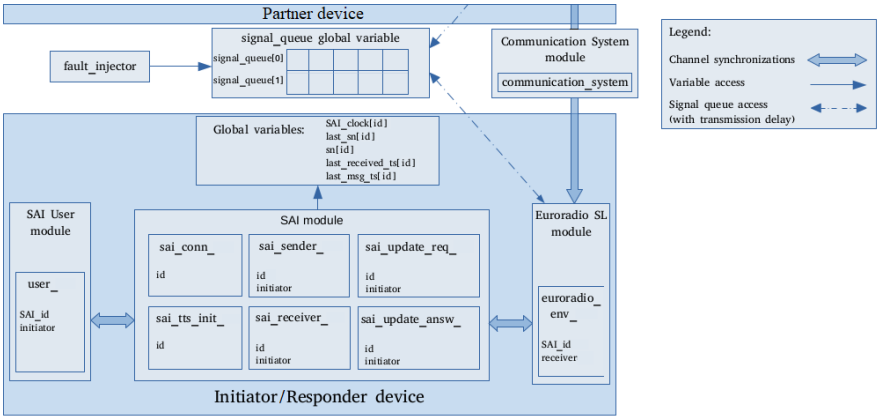

Fig. 1. The model architecture, the partner device is specular to the one displayed

are instead completely modelled. The SAI module is divided into different submodules. The TTS technique is implemented in the TTS initialisation and the TTS update procedure modules. The following list shows the templates that make up a single device:

The SAI_User template abstracts the SAI User module behaviour, implementing the triggering of a connection and periodically sending Application messages. It is instantiated specifying the `SAI_id` parameter and the `initiator` parameter;

The Euroradio_SL_Env template abstracts the Euroradio SL module behaviour implementing the stochastically delayed message exchange with the partner device. It is instantiated by specifying the `id` parameter and the `receiver` parameter;

The SAI_Conn_Ini/SAI_Conn_Res templates model the connection establishment according to the role of the device. The templates are instantiated specifying the `id` parameter corresponding to the device they belong to;

The SAI_TTS_Init_Ini/SAI_TTS_Init_Res templates model the TTS initialisation, depending on the role of the device. They exchange messages to estimate the minimum and maximum offset delay of messages. Their parameter is the same as SAI_Conn_Ini/SAI_Conn_Res;

The SAI_Update_Req templates implement the offset estimations update requests, to update the minimum and maximum offset delay. Its parameters are the same as the SAI_Sender template;

The SAI_Update_Answ template models the TTS update procedure by sending the offset estimations update answers to SAI_Update_Req. Its parameters are the same as the SAI_Sender template;

The SAI_Sender template models the SAI defence techniques, by inserting into each message the three timestamps of TTS and the sequence number,

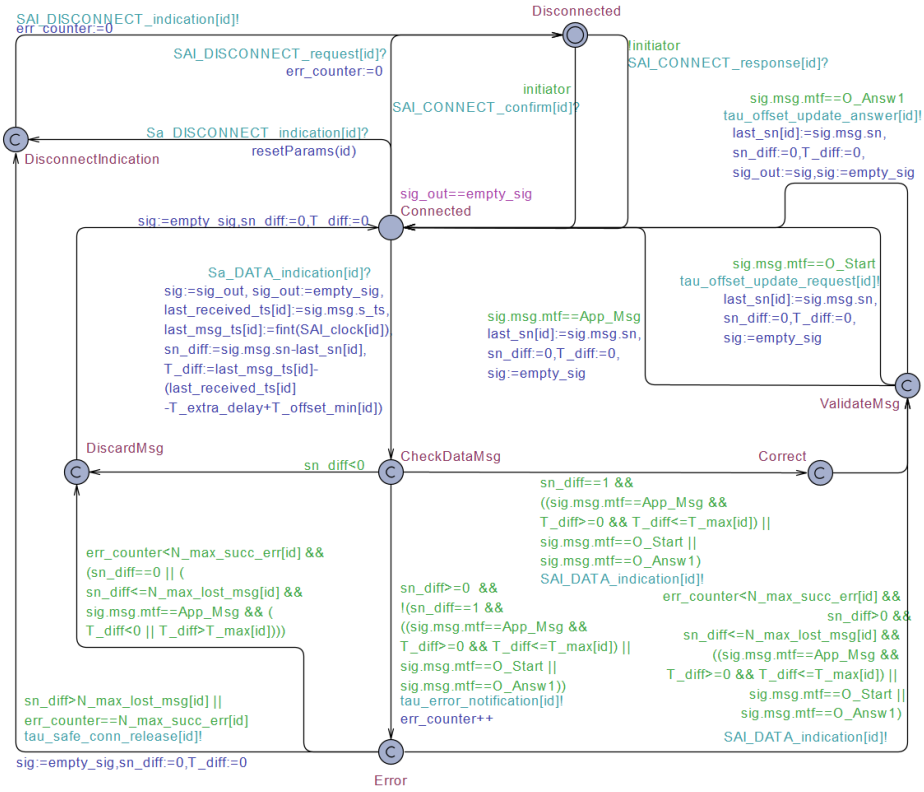

Fig. 2. The SAI_Receiver template

implementing the message sending procedure. It is instantiated with the `id` and the `initiator` parameters identifying its device and role;

The SAI_Receiver templates implement the check procedure for all the incoming messages. It has the same parameters as the SAI_Sender.

We discuss in details the template implementing the check of messages for the protections and the fault injection implementing the various CENELEC threats.

SAI_Receiver The SAI_Receiver template implements the protection against the repetition, deletion, resequencing and delay threats that can occur in a transmission system, commanding itself the connection release if certain unsafe conditions are met, and it is shown in Figure 2. It performs functionalities related to the protection against delay, by checking the timestamps of messages. In state `Connected`, if a data message from the partner device is received (i.e. `Sa_DATA_indication[id]?`), the `last_received_ts` and `last_msg_ts` variables are updated. Then, the sequence number difference (referred to as `sn_diff`) between the sequence number received in the message and the last

sequence number stored in the last_sn variable is computed. Also the freshness of the received message (referred to as T_diff) is computed as the difference between the timestamp at the message reception (the last_msg_ts value just updated) and the estimation of the message time transmission in term of the SAI_Receiver clock, i.e. the sum of the last received timestamp (the last_received_ts value just updated) and the minimum offset estimation computed by the Receiver device during the TTS initialization, with the extra delay subtracted. The sn_diff computed values determine the SAI_Receiver behaviour according to the three outgoing transitions from state CheckDataMsg. If sn_diff==1 and $0 \leq$ T_diff $\leq$ T_max for an Application Message (i.e. a message to the SAI User), the received data message is not affected by sequencing errors and the transmission delay is acceptable. Assuming that also the conditions for no delay errors occur, the SAI_Receiver notifies its adjacent SAI_User of the correct message reception through the SAI_DATA_indication, moves first to the Correct location and then to the ValidateMsg location. Here, according to the message type field of the received message, the SAI_Receiver can forward a signal to the SAI_Update_Answ or the SAI_Update_Req templates respectively when concerning offset update request or offset update answer messages. Otherwise, in case of an Application message, the Connected location is entered without further actions. If sn_diff$<$0, the message is discarded (location DiscardMsg) without notifying the SAI_User. If sn_diff $\geq$ 0 and the previous conditions are not verified (i.e. sn_diff $\neq$ 1 or the transmission delay is not acceptable, i.e. either T_diff $<$ 0 or T_diff $>$ T_max), the SAI_Receiver notifies the SAI_User, updates its error counter and then enters the Error location. Here, if the maximum number of either successive errors N_max_succ_err or lost messages N_max_lost_msg is reached, the SAI_Receiver immediately sends a safe connection release (i.e. tau_safe_conn_release) to the SAI_Sender, which in turn sends a disconnect request to the peer entity, and from location DisconnectIndication a SAI_DISCONNECT_indication is sent to the SAI_User to command the release of the connection. Instead, if the maximum number of successive errors is not reached and the received message is a repetition of the last accepted message (i.e. sn_diff==0), or its transmission delay is not acceptable, the message is discarded. Anyway, the message can be validated if both its transmission delay and the number of lost messages are acceptable (i.e. $0 \leq$ T_diff $\leq$ T_max and 1$<$sn_diff $\leq$ N_max_lost_msg).

Moreover, from state Connected the SAI_Receiver synchronizes with the Sa_DISCONNECT_indication signal from the Euroradio_SL_Env. When a disconnect indication is received, the SAI_Receiver forwards the communication to the SAI_User before entering the Disconnected location and also all the other SAI templates of the same device that by synchronizing through the channel SAI_DISCONNECT_indication move to the Disconnected locations.

Fault_Injector The Fault_Injector template shown in Figure 3 models all the possible threats that can affect the communication system. This template acts as a fault injector in the signal queue of the two communicating devices, determining the occurrence of communication errors or simulating transmission problems

that could lead to unacceptable delays. This could also be considered as a model of an external attacker artificially injecting failures [12]. This template provides a minimum waiting time to model the probability of occurrence. Two probabilistic branches decide whether the attempt to perform a fault injection is successful or not. By fine-tuning the `fault` and `noFault` weights associated with the probabilistic branches, the probability of the desired fault occurrence can be adjusted.

Only if the signal queue of the non-deterministically chosen device to perform the fault injection is not empty, all the devices are connected and the attempt is successful, a probabilistic branching decides which threat to perform. The possible threats are: *deletion threat* (the first signal of the queue is removed), *repetition threat* (the first signal is repeated if the queue is not full, otherwise no repetition is performed), *resequencing threat* (the first signal is shifted of one or two positions inside the queue, if at least another signal is present, otherwise no re-sequencing is performed), and *transmission delay threat* (the `msgDelay` variable of the selected device is updated with the `msgDelayInjected` value).

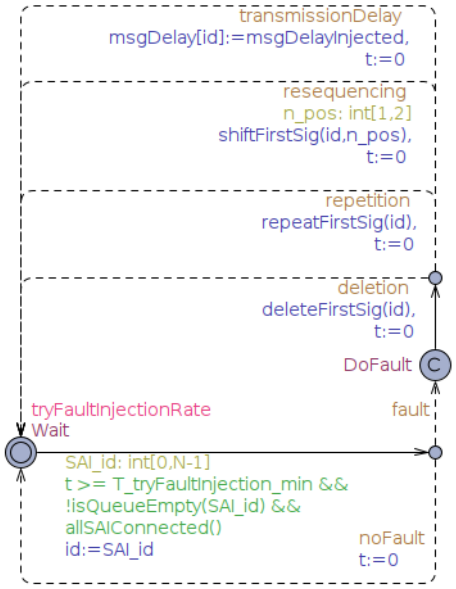

Fig. 3. The Fault_Injector template

The transmission delay is based on the update of the `msgDelay` variable, which defines the rate for the exponential distribution of the edge performing the signal dequeue. The injected rate is lower than the standard rate assigned to `msgDelay`, and during the sampling of the exact delay, the smaller the rate is specified, the longer the delay is preferred. Hence, with the injected rate it is possible to simulate a longer transmission delay, increasing the probability to exceed the validity time for the incoming messages.

4 The analysis

In this section we discuss the analysis of the model and the issues found. As required by Subset-039 [26] (ref. 4.2.1.2) only one RBC/RBC communication between a pair of RBCs must be active at one time. Thus, we focus on analysing a single pair of communicating devices. The specification suggests which parameters values to use with particular systems (for example highly-available systems), whereas the definition of other parameters is left to the specific application set-

tings. Our experiments consider a high probability of fault injection success, and diminishing the rates allows to model longer delays. This allows to observe faults with fewer simulations, thus quickly verifying the defence techniques against such faults. Finally, the configuration of certain parameters is done according to the need for a stable connection where both the probability of the SAI_User to send a disconnection request, the Communication_System to perform a disruptive connection release and the Euroradio_SL_Env to fail are very low. We remark that the purpose of our analysis is not the accurate quantification of measures such as performance or reliability. This would require a realistic, less extreme set-up of parameters with lower fault probabilities and time-expensive verification. We address the qualitative verification of the protection mechanisms. Nonetheless, threats for open systems [12] are also considering attackers artificially injecting faults into the communication system.

All the verified properties are related to the probability estimation of the occurrence of a specific hazard. If the probability of occurrence of the hazards is close to zero (i.e. $p' = 0$, see Sect. 2) the model satisfies its safety requirements with a certain degree of confidence $(1-\alpha)$ dictated by the parameters of the statistical model checker (probability of false negatives $\alpha = 0.0005$ and probability uncertainty $\epsilon = 0.005$). We recall that all evaluated formulae are of the form:

$$\varphi_i \texttt{ = Pr[<=bound](<>conf\_i).}$$

We set `msg_freq=8` time units (i.e. the period in which SAI_User attempts to send a new Application message) and `bound=1000` (i.e. trace length), thus allowing to perform faster simulations but still inclusive of a non-negligible message exchange. With this formula "template" only the logical conditions are left to be specified thus making easier the formalisation of the properties also for users not expert in temporal logics. In the following we only provide the specific configuration conditions `conf_i` of each formula indexed by i, and for improving readability we use the names of the templates, even though the names of the corresponding instantiations have been used in Uppaal.

Model checking We start by checking some properties that the model should meet and that are not related to the specification. Only for these formulae, we set α and ϵ to 0.05 to have a faster evaluation (N=738, see Sect. 2). Firstly, since the queue of messages is bounded, an excessive message delay could cause the queue to be filled if the size of the queue is not properly set. To ensure that this event does not occur, we measure the probability that there exists a full queue within 1000 time units using `conf_1 = exists(id: id_t) isQueueFull(id))`. The probability is evaluated to be close to zero (based on α and ϵ).

The next formula concerns the connection procedure, and in particular the parameter `T_conn_max` that is the maximum waiting time between two connection requests. The requirements specify that during the TTS initialization procedure, a `T_start_max` maximum waiting time for the incoming offset messages from the Responder device is provided. Thus it is important to issue new connection requests only if the TTS initialization procedure has exceeded its time limit. If this is not the case, a specific location `ConnectionFailure` is entered by the SAI_TTS_Init_Ini. Thus we set `T_conn_max` = `c`$*$`T_start_max` and we experimentally find the threshold value for the constant `c` beyond which the probability to

enter the `ConnectionFailure` location is not close to zero. This is evaluated with the formula `conf_2=SAI_TTS_Init_Ini.ConnectionFailure` performing several experiments at the varying of the constant `c`. The degradation of the model occurs for `c`$\leq$`3`, hence we set `c=4`. Concerning the possibility of undetected message loss, we recall that we used state invariants to ensure that all synchronous signals are received. Finally, we verify the high probability of fault injection that happens when the Fault Injector template enters the `DoFault` location with the formula `conf_3=Fault_Injector.DoFault`. The evaluation of φ_3 is evaluated with a value close to one, thus meaning that at least one fault is always injected in each simulation of the system.

Verifying Safety We now discuss the verification of the safety properties related to the effectiveness of the defence techniques described in the specification and implemented in our model. The hazards are modelled as "bad" configurations (expressed by `conf`) of the system and have been identified by a manual review of the model based on the requirements. We followed a schematic methodology where first the probability of occurrence for a particular threat, among those specified in the requirements, is evaluated. Then we evaluate the probability that the system does not behave as expected when that particular threat occurs. This second probability evaluation makes sense only if the first probability evaluation is non-negligible (i.e. $p' > 0.1$), thus meaning that the system is actually verified when the threat has a non-negligible probability to occur. Note that identifying the severity of each hazard is out of scope.

The first two analysed hazards concern the probability of either receiving a correct message that is considered erroneous or treating a message affected by some communication errors as a correct message. The φ_4 formula checks if the SAI_Receiver of both the Initiator and the Responder devices receives a correct message, i.e. the configuration where the sequence number difference with the previous message is 1 (`sn_diff==1`) and its delay is acceptable (`T_diff` $\geq$ `0` && `T_diff` $\leq$ `T_max`), and the system treats it as an error, i.e. the SAI_Receiver template enters the `Error` the `DiscardMsg` location. Formula φ_5 checks if the SAI_Receiver of both the Initiator and the Responder devices receives an erroneous Application Message, i.e. the sequence number difference with the previous message is not 1 (`sn_diff!=1`) or its delay is unacceptable (`T_diff` $<$ `0` || `T_diff` $>$ `T_max`), and the system considers it as a correct message, i.e. the SAI_Receiver enters the `Correct` location. Note that this location is necessary to distinguish the correct messages from the messages that are validated despite the sequence number difference with the previously accepted message is different from 1. Both φ_4 and φ_5 are evaluated with a value close to zero.

Concerning the threats that can occur in a communication system, we measure the probability of occurrence of six possible hazards caused by the CENELEC threats (formulae/subformulae with even id numbers), and we measure the probability of the corresponding protection to fail (with odd id numbers). Due to lack of space, we do not fully report the formulae. All the formulae are predicating over locations and variables of the SAI_Receiver (see Fig. 2). All threats probabilities were evaluated with a non-negligible probability of occur-

rence, confirming the fault injection. The corresponding probabilities of protection failure were all evaluated with values close to zero. Concerning the resequencing threat we found two hazards. Condition `conf_6` checks if a message earlier than the last accepted one is received (`sn_diff<0`); and `conf_7` checks if under this condition the SAI_Receiver does not discard this message (does not visit the location `DiscardMsg`). Another condition `conf_8` checks if a message with the same sequence number of the last accepted one is received, and `conf_9` checks if it is validated (location `ValidateMsg`). For the resequencing threat we found one hazard, and another one is obtained by combination with the delay threat. Condition `conf_10` (resp. `conf_12`) is satisfied if a message arrives with both an acceptable (resp. unacceptable) delay and with a sequence number difference between 2 and `N_max_lost_msg`. Condition `conf_11` checks if under `conf_10` the message is discarded or accepted (resp. locations `DiscardMsg` or `Correct`), whilst condition `conf_13` checks if under `conf_12` the error location is not entered by the SAI_Receiver. Note that for entering such location, it is required that both the delay is unacceptable and `sn_diff` is positive (see Fig. 2), thus the necessity of mixing the two threats. For the deletion threat condition `conf_14` checks if a message with a sequence number difference greater than `N_max_lost_msg` is received, and `conf_15` checks if it is discarded or validated. In this case it was also required the possibility of tolerating one communication error. Finally, for the delay threat we have one hazard: condition `conf_16` checks if a message with a correct sequence number (`sn_diff=1`) but with an unacceptable delay is received, and `conf_17` checks if it is valid (location `ValidateMsg`).

We verified three additional safety properties for further validation of the model. Condition `cond_18` is used for ensuring that the TTS initialization is completed before receiving any application message. Condition `cond_19` verifies that only correct messages are forwarded from the SAI to the user, and condition `cond_20` verifies if the SAI module correctly commands the release of the connection when reaching the maximum number of successive errors during the message exchange with the partner device. These last three formulae are evaluated in scenarios where the system can reach unsafe configurations due to the fault injection of communication errors, and are all satisfied. The results presented so far meet our expectations, augmenting our confidence that both the model and the adopted defence techniques are correct.

Issues detected We report the most relevant issues discovered with the analysis, which have been confirmed by our industrial partners in the 4SECURail project. Whilst some issues were already known and are due to negotiations among the UNISIG members for compatibility with their legacy solutions, others are new and will possibly lead to request for changes of the Subset-098.

Zero-crossing The first problem we met concerns the implementation of both the sequence number defence and TTS technique. Indeed, in the Subset-098 specification, for both protection techniques (sequence number and TTS) it is not specified how to behave in the presence of zero-crossing (i.e., overflow of the assigned bytes), and the specification only considers the case without overflow. We only detail the sequence number problem in the following. Through the

formula φ_{21}, we verify if the above unsafe scenario is reachable for the `SN_max` parameter (bound of the sequence number) set to a lower value, e.g. 100.

Condition `conf_21` checks if the SAI_Receiver enters either the `Error` location or the `DiscardMsg` location when receiving an incoming message (`sig !=empty_sig`) with sequence number 0 (`sig.msg.sn==0`) and its last accepted message had sequence number `last_sn[id]==SN_max`. This formula refers to the scenario in which the Sender device performs the zero-crossing of the sequence number. Even if the message stream is correct, the Receiver device behaves as if a communication error occurred, an undesired scenario. The formula is evaluated (where both the probability of false negatives α and the probability uncertainty ϵ are set to 0.05) with $p' = 0.847987$. An example of mitigation of this sequence number zero-crossing problem is to force the release of the safe connection from the SAI module when the maximum value for the sequence number is reached. We refer to domain experts for more efficient solutions to this problem. Note that without bounding the maximum number of consecutive lost messages, when approaching the zero-crossing it is not possible to distinguish between the reception of an earlier message or the loss of consecutive messages. The Subset-098 leaves this aspect of the protection open, which could potentially lead to unsafe scenarios in case of communication errors.

TTS initialisation We identified an undefined scenario of the system that could lead to possible unsafe configurations if no specific actions are implemented. Briefly, if during the TTS initialisation a specific notification message (i.e. OffsetStart) sent to the Responder device is lost, the Responder is stuck until a new connection request arrives from the Initiator. In the Subset-098, there is no mention of this scenario: it is not included in the SAI initial procedures at the error handling section (ref. 5.4.10.1.3 [25]) as no TTS initialization for the Responder is started yet. Moreover, no communication with the SAI User can be assumed, as the interactions between the SAI and the SAI User modules of the Responder device start after the successful TTS initialization. In our model, in this scenario the failed connection procedure is interrupted to restart a new one from the beginning. Obviously, this implies that the Initiator must send again a connection request. Another possible solution would be for the Responder to answer again to the connection request.

TTS offset update Another undefined aspect concerns the type fields of the offset update messages. It is only mentioned in ref. 5.4.8.7.3 and Figure 21 of [25] that two messages are exchanged. Since the two communicating devices can both start the update procedure and the two distinct update procedures can overlap, it could be the case that one device, after sending a request for offset update, cannot discern whether the received message is an answer to the previous request or a new request of the same type. In our model, we re-used the TTS initialization message type fields to distinguish the two types of update messages, and in particular the OffsetStart for the update request message and the OffsetAnsw1 for the update answer message. These implementation choices were made necessary to model a working offset update procedure despite the lack of details in the specification. The fact that different suppliers could implement these aspects in-

dependently could compromise the interoperability of their RBCs. Moreover, the Subset-098 [25] does not specify the actions to perform in case the timestamps of the offset update answer and request messages (used for relating them) do not correspond, a condition necessary to update the offset estimation. We identified two possible behaviours for the SAI module in this case: either it reissues a new request without waiting, or it can wait for the right answer, and reissues a new request only at the expiration of the timer. We opted for the second case. Indeed, the first case could cause a loop where the answering device keeps sending answer messages but they arrive when another request has already been sent, so the timestamps would not correspond again. The subset leaves both implementations possible, whilst only the second case should be allowed for avoiding unnecessary disconnections with errors.

Error tolerance Finally, we discuss the configuration choice of the maximum number of successive errors (`N_max_succ_err`) parameter, which [25] specified to have values either 1 or 2 (thus allowing a maximum tolerance of one error). By analysing the model, we noted that a transmission delay causes a rejection of the message, thus incrementing the error counter and triggering a subsequent sequence number error due to the discarding of the previous message. This means that the ability of the system to tolerate the occurrence of one communication error is tied to the type of the error being detected: if the error is a transmission delay, then the system is no longer able to accept the next incoming message as correct, even if the tolerance is set to do so.

The formal verification step helped in identifying the problems reported in this section, thus resulting very useful. It has been used to check the correctness of the model by debugging modelling errors during its development, as well as to formally validate the requirements of the system defined in the specification and discovering corner cases where both protections are ineffective.

Quantifying the learning and development efforts We provide a rough but indicative estimation of the costs for training and development. We considered the effort in terms of CFU (Crediti Formativi Universitari, corresponding to ECTS − European Credit Transfer and Accumulation System − credits) sustained by the third author for developing her Master thesis (containing the presented results) under the supervision of the first two authors. The training in formal methods has been provided during the *Software Dependability* university course in Florence University, provided by the second author, where the basis for the modelling aspects and the logic required to understand the model checking algorithm have been studied, which are almost 7 CFU [15].

Concerning the work presented in this paper, it can be quantified in the 24 CFU of the Master thesis. There is no clear division between the modelling and the verification phases because since the beginning of the modelling phase, a constant activity of verification of the model was made. Indeed, useful counterexamples from the verification allowed to debug modelling errors in the first model prototypes. Considering that each CFU conventionally corresponds to 25 working hours, we can estimate an effort of 775 working hours to reach the result we described in this paper, starting from no knowledge of formal methods,

which we report to adhere to the actual effort. Thus, an indicative division of the working hours required for each activity is: *Learning - 169 hours, Modelling and Verification - 606 hours*. These data can be validated by consulting the online regulations of the Degree Course [27] and the Master thesis [23]. Finally, assuming a fresh and already trained graduate as the third author, a fitting payment (in Italy) could be a professionalising grant (*assegno professionalizzante* [10]). In this specific case, she/he would have an annual gross cost of 21343 euro (as per the time of writing this paper), with a cost per hour of 15.56 euro, which roughly gives us a gross cost of 9400 euro for producing both the artifacts and the analysis described in this paper.

5 Conclusion

We have formalised and analysed an existing industrial specification already in operation: the UNISIG Subset-098 [25], and in particular the Safety Application Intermediate sub-layer, whose goal is to protect the system from the CENELEC 50159 [12] threats of open transmission systems. The analysis has discovered corner cases where the protections are not effective, due to unspecified scenarios or ambiguous requirements. We discussed simple mitigations to such issues, not detailed in the specification. Since different interpretations of these undefined aspects could be given as well by different suppliers (especially for newcomers, rather than the original members of the consortium that issued these subsets), this could lead to non-interoperability between RBCs developed and provided by different suppliers. This appears to be the current situation for this standard interface, as reported in [21], where interoperability issues during the handover procedure were encountered in the Milano-Bologna line containing three RBCs produced by different suppliers (Alstom and Ansaldo) using the same requirements analysed in this paper. Finally, considering that formal methods are still a subject of study in the field of railway industrial applications [14, 18], as witnessed by several Shift2Rail projects, this paper represents a further contribution to evaluate the usefulness in terms of costs of learning and development, as well as benefits deriving from the adoption of formal methods in this domain. The model and analysis for the various scenarios are publicly available in [22].

We argue that the benefits derived from the work described in this paper are not only limited to the identified safety and interoperability issues. Indeed, the provided formal model and the safety properties analysed can enrich the existing documentation. By simply tuning the parameters to realistic, less extreme values it is possible to have evaluations of other dependability aspects of the system such as performance, reliability. The presented model could also be the starting point for other model-based development approaches, e.g., by translating the SAI models into state machines (e.g., RT-UML) for code generation or model-based testing. A subset of the authors already provided a translation from RT-UML machines to Uppaal models in [2]. As future work it could also be of interest to enrich the model with full formalisation of the other layers as well as the execution cycle protection technique.

Acknowledgements This work has been partially funded by the H2020 Shift2Rail 4SECURail project, grant agreement No 881775 in the context of the open call S2R-OC-IP2-01-2019, programme H2020-S2RJU-2019.

References

1. Agha, G., Palmskog, K.: A survey of statistical model checking. ACM Transactions on Modeling and Computer Simulation **28**(1), 6:1–6:39 (2018). https://doi.org/10.1145/3158668
2. Basile, D., ter Beek, M.H., Ciancia, V.: Statistical model checking of a moving block railway signalling scenario with Uppaal SMC: Experience and outlook. In: Margaria, T., Steffen, B. (eds.) ISoLA. LNCS, vol. 11245, pp. 372–391. Springer (2018). https://doi.org/10.1007/978-3-030-03421-4_24
3. Basile, D., Fantechi, A., Rucher, L., Mandò, G.: Statistical model checking of hazards in an autonomous tramway positioning system. In: Collart-Dutilleul, S., Lecomte, T., Romanovsky, A. (eds.) RSSRail. LNCS, vol. 11495, pp. 41–58. Springer (2019). https://doi.org/10.1007/978-3-030-18744-6_3
4. Basile, D., ter Beek, M.H., Fantechi, A., Ferrari, A., Gnesi, S., Masullo, L., Mazzanti, F., Piattino, A., Trentini, D.: Designing a demonstrator of formal methods for railways infrastructure managers. In: Margaria, T., Steffen, B. (eds.) ISoLA. LNCS, vol. 12478, pp. 467–485. Springer (2020). https://doi.org/10.1007/978-3-030-61467-6_30
5. Basile, D., ter Beek, M.H., Ferrari, A., Legay, A.: Modelling and analysing ERTMS L3 moving block railway signalling with Simulink and Uppaal SMC. In: Larsen, K.G., Willemse, T. (eds.) FMICS. LNCS, vol. 11687. Springer (2019). https://doi.org/10.1007/978-3-030-27008-7_1
6. Basile, D., ter Beek, M.H., Legay, A.: Strategy synthesis for autonomous driving in a moving block railway system with Uppaal Stratego. In: Gotsman, A., Sokolova, A. (eds.) FORTE. LNCS, vol. 12136, pp. 3–21. Springer (2020). https://doi.org/10.1007/978-3-030-50086-3_1
7. Behrmann, G., David, A., Larsen, K., Håkansson, J., Pettersson, P., Yi, W., Hendriks, M.: UPPAAL 4.0. In: Proceedings of the 3rd International Conference on the Quantitative Evaluation of SysTems (QEST'06). pp. 125–126. IEEE (2006). https://doi.org/10.1109/QEST.2006.59
8. Bulychev, P., David, A., Larsen, K.G., Legay, A., Li, G., Poulsen, D.B.: Rewrite-based statistical model checking of WMTL. In: Qadeer, S., Tasiran, S. (eds.) RV. LNCS, vol. 7687, pp. 260–275. Springer (2013). https://doi.org/10.1007/978-3-642-35632-2_25
9. Chai, M., Wang, H., Tang, T., Liu, H.: Runtime verification of train control systems with parameterized modal live sequence charts. Journal of Systems and Software **177**, 110962 (2021). https://doi.org/https://doi.org/10.1016/j.jss.2021.110962
10. CNR: Assegni di ricerca, `https://www.urp.cnr.it/page.php?level=15&pg=1522`
11. David, A., Larsen, K.G., Legay, A., Mikučionis, M., Poulsen, D.B.: Uppaal SMC tutorial. International Journal on Software Tools for Technology Transfer **17**(4), 397–415 (2015). https://doi.org/10.1007/s10009-014-0361-y
12. European Committee for Electrotechnical Standardization: CENELEC EN 50159 – Railway applications – Communication, signalling and processing systems – Safety-related communication in transmission systems (2010), `https://standards.globalspec.com/std/14256321/EN50159`

13. European Committee for Electrotechnical Standardization: CENELEC EN 50128 – Railway applications – Communication, signalling and processing systems – Software for railway control and protection systems (2020), https://standards.globalspec.com/std/14317747/EN2050128
14. Fantechi, A.: Twenty-five years of formal methods and railways: What next? In: Counsell, S., Núñez, M. (eds.) SEFM. LNCS, vol. 8368, pp. 167–183. Springer (2013). https://doi.org/10.1007/978-3-319-05032-4_13
15. Fantechi, A.: Software Dependability course. University of Florence, https://www.unifi.it/p-ins2-2018-502809-0.html
16. Ferrari, A., Mazzanti, F., Basile, D., ter Beek, M.H., Fantechi, A.: Comparing formal tools for system design: a judgment study. In: Proceedings of the 42nd International Conference on Software Engineering (ICSE). pp. 62–74. ACM (2020). https://doi.org/10.1145/3377811.3380373
17. Ferrari, A., Mazzanti, F., Basile, D., ter Beek, M.H.: Systematic evaluation and usability analysis of formal tools for system design. arXiv:2101.11303 [**cs.SE**] (2021), https://arxiv.org/abs/2101.11303
18. Garavel, H., ter Beek, M.H., van de Pol, J.: The 2020 expert survey on formal methods. In: ter Beek, M.H., Ničković, D. (eds.) FMICS. LNCS, vol. 12327, pp. 3–69. Springer (2020). https://doi.org/10.1007/978-3-030-58298-2_1
19. Huang, J., Lv, J., Feng, Y., Luo, Z., Liu, H., Chai, M.: A novel method on probability evaluation of ZC handover scenario based on SMC. In: Qian, J., Liu, H., Cao, J., Zhou, D. (eds.) ICRRI. CCIS, vol. 1335, pp. 319–333. Springer (2020). https://doi.org/10.1007/978-981-33-4929-2_22
20. Legay, A., Lukina, A., Traonouez, L.M., Yang, J., Smolka, S.A., Grosu, R.: Statistical model checking. In: Steffen, B., Woeginger, G.J. (eds.) Computing and Software Science: State of the Art and Perspectives, LNCS, vol. 10000, pp. 478–504. Springer (2019). https://doi.org/10.1007/978-3-319-91908-9_23
21. Morselli, S.: Il nuovo servizio ferroviario ad Alta Velocità "Frecciarossa": analisi delle performance. Master's thesis, University of Bologna (2009), http://amslaurea.unibo.it/435/
22. Rosadi, I.: Repository for reproducing the experiments (2021), https://github.com/IreneRosadi/UppaalModels
23. Rosadi, I.: Analysing a safe communication protocol in the railway signaling domain with Timed Automata and Statistical Model Checking. Master's thesis, University of Florence (2021)
24. Shift2Rail: Annual Work Plan and Budget (2021), https://shift2rail.org/about-shift2rail/reference-documents/annual-work-plan-and-budget/
25. UNISIG: RBC-RBC safe communication interface, Subset-098, v3.0.0 (2012), https://www.era.europa.eu/sites/default/files/filesystem/ertms/ccs_tsi_annex_a_-_mandatory_specifications/set_of_specifications_3_etcs_b3_r2_gsm-r_b1/index063_-_subset-098_v300.pdf
26. UNISIG: FIS for the RBC/RBC handover, Subset-039, v3.2.0 (2015), https://www.era.europa.eu/sites/default/files/filesystem/ertms/ccs_tsi_annex_a_-_mandatory_specifications/set_of_specifications_3_etcs_b3_r2_gsm-r_b1/index012_-_subset-039_v320.pdf
27. University of Florence: Regulations of the M.Sc. degree, https://www.informaticamagistrale.unifi.it/vp-165-regulations.html