A Refinement-based Formal Development of Cyber-physical Railway Signalling Systems

Yamine Aït-Ameur, Sergiy Bogomolov, Guillaume Dupont, Alexei Iliasov, Alexander Romanovsky, Paulius Stankaitis

A Refinement-based Formal Development of Cyber-physical Railway Signalling Systems

YAMINE AÏT-AMEUR, INPT–ENSEEIHT, France
SERGIY BOGOMOLOV, Newcastle University, United Kingdom
GUILLAUME DUPONT, INPT–ENSEEIHT, France
ALEXEI ILIASOV, The Formal Route Limited, United Kingdom
ALEXANDER ROMANOVSKY and PAULIUS STANKAITIS, Newcastle University, United Kingdom

3

For years, formal methods have been successfully applied in the railway domain to formally demonstrate safety of railway systems. Despite that, little has been done in the field of formal methods to address the cyber-physical nature of modern railway signalling systems. In this article, we present an approach for a formal development of cyber-physical railway signalling systems that is based on a refinement-based modelling and proof-based verification. Our approach utilises the Event-B formal specification language together with a hybrid system and communication modelling patterns to developing a generic hybrid railway signalling system model that can be further refined to capture a specific railway signalling system. The main technical contribution of this article is the refinement of the hybrid train Event-B model with other railway signalling sub-systems. The complete model of the cyber-physical railway signalling system was formally proved to ensure a safe rolling stock separation and prevent their derailment. Furthermore, the article demonstrates the advantage of the refinement-based development approach of cyber-physical systems, which enables a problem decomposition and in turn reduction in the verification and modelling effort.

CCS Concepts: • **Software and its engineering → Software verification;**

Additional Key Words and Phrases: Formal verification, hybridised Event-B, cyber-physical railway signalling systems

This work is partially supported by an iCASE studentship (funded by EPSRC/UK and Siemens Rail Automation), EPSRC STRATA platform grant (No. EP/N023641/1) and Air Force Office of Scientific Research under Award No. FA2386-17-1-4065. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the United States Air Force. This work is also supported by the DISCONT Project of the French National Research Agency (Grant No. ANR-17-CE25-0005, The DISCONT Project, https://discont.loria.fr).
Authors' addresses: Y. Aït-Ameur and G. Dupont, INPT–ENSEEIHT, 2 Rue Charles Camichel, Toulouse, France; emails: {yamine, guillaume.dupont}@enseeiht.fr; S. Bogomolov, A. Romanovsky, and P. Stankaitis, Newcastle University, Science Square 1, Newcastle upon Tyne, United Kingdom; emails: sergiy.bogomolov@newcastle.ac.uk, paulius.stankaitis@ncl.ac.uk, {alexander.romanovsky, paulius.stankaitis}@newcastle.ac.uk; A. Iliasov, The Formal Route Limited, 32A Woodhouse Grove, London, United Kingdom; email: alexei.iliasov@formal-route.com.

0934-5043/2023/01-ART3
https://doi.org/10.1145/3524052

ACM Reference format:
Yamine Aït-Ameur, Sergiy Bogomolov, Guillaume Dupont, Alexei Iliasov, Alexander Romanovsky, and Paulius Stankaitis. 2023. A Refinement-based Formal Development of Cyber-physical Railway Signalling Systems. *Form. Asp. Comput.* 35, 1, Article 3 (January 2023), 24 pages.
https://doi.org/10.1145/3524052

1 INTRODUCTION

Over the past few decades, developments in communication and computing technologies have made it possible for railway engineers to develop novel railway signalling systems. These modern railway signalling systems (e.g., **European Train Control System (ETCS)** [17] or **Communication-based Train Control (CBTC)** [23]) are safety-critical cyber-physical transportation systems that aim to improve the capacity, interoperability, and reliability of railway networks. In these modern signalling systems, a train continuously receives a permitted travelling distance via a wireless communication, and an on-board train computer calculates the fitting speed profile. The calculated speed profile of a train is then dynamically followed and supervised by a driver or a computer-based speed controller. The increased dependability on the on-board computers and algorithms for computing, controlling, and monitoring the speed profile of a train requires the highest level of system assurance and poses new challenges to signalling engineers on ensuring the safety of the cyber-physical railway signalling systems.

To achieve such a high-level of safety assurance of those complex signalling systems, scenario-based testing methods are far from being sufficient, despite that they are still widely used in the industry. Alternatively, formal methods have been successfully applied in the railway domain [6, 9] and seem quite fitted for dealing with railway signalling systems. However, little has been done in the field of formal methods to address the cyber-physical nature of modern railway signalling systems. One of the main challenges of model-based cyber-physical signalling system development stems from the necessity to capture and formally reason at a system-level as, for example, safety of these systems heavily depends on correct interactions between interlocking, communication centres and rolling stock [7]. Therefore, the formal modelling language that would be used in the model-based development should be expressive enough to support a system-level modelling and verification. Furthermore, cyber-physical systems have both discrete and continuous behaviours that are best captured by hybrid models. A formal verification of hybrid systems have been a major challenge and an active research area. The challenges of formally verifying hybrid systems arise mainly due to real-valued state variables and systems with non-linear dynamics [3, 30]. The algorithmic verification of hybrid models with available model checking tools are limited even under severe restrictions. An alternative proof-based verification approach that is not limited by the state-space and combined with computer algebra systems can deal with non-linear dynamics [27]. Despite that an automated hybrid system deductive verification is still a major challenge and for an industrial application, an interactive proof effort should be dramatically reduced.

In this work, we present a formal development approach of cyber-physical railway signalling systems that makes it possible to capture and reason about safety of railway signalling systems at a system-level. The proposed approach also supports a verification of properties that are related to continuous rolling stock behaviour. Our introduced approach is based on the Event-B modelling language [2], which supports a refinement-based model development and proof-based model verification. The main contribution of this work is an Event-B model of a cyber-physical railway signalling system that is developed by utilising Event-B patterns for modelling hybrid [15] and communication [32] systems. The cyber-physical railway signalling system model is built upon our previous paper [31] in which we formally developed a speed controller that ensures a train

stays within a permitted travelling distance. In this work, we refine the speed controller Event-B model by formally introducing other railway signalling sub-systems that are responsible for computing and communicating safe travelling distances to the rolling stock. The other sub-systems and signalling communication protocol are formally modelled by following the communication modelling patterns that were introduced in our previous work [32]. This article also evaluates our proposed formal development methodology of cyber-physical systems and demonstrates the key benefits of its refinement-driven modelling for a system complex development.

Related Work. In the work by Berger et al. [10], the authors use a real-time modelling approach to developing and verifying the ERTMS model. A multifaceted formalism is introduced in Reference [22] to reason about real-time systems with a case study on railway crossings. In the work by Cimatti et al. [14], the authors propose a different logic based on the temporal logic with regular expressions and use it for requirement validation of hybrid railway systems. Their verification approach, based on a state-exploration, is used to demonstrate the desired and safe behaviour of the model. Halchin et al. [19, 20] propose a certified translation from B formal language to HLL for developing railway software. In this work, the Isabelle/HOL theorem prover is used to check the correctness of the translation process, and the train localization in a CBTC system is used to illustrate the overall approach and the developed tool B2HLL. Even though, the model-checking approaches are desirable due to their push-button verification advantage, the approach rarely scales for realistic scenarios, particularly in the hybrid domain. In the other strand of works, an alternative to model checking—proof-based modelling—is used to verify European and Chinese Train Control Systems Level 3 [8, 28, 34]. These studies are more related to our work as the authors not only formally model rolling stock but also capture other sub-systems of the signalling system (only level crossings [22]) and validate systems through proofs. However, in contrast to these papers, our work considers a more complex and realistic physical model of a train and other signalling sub-systems are modelled at a lower abstraction level. Last, we note that our approach is based on stepwise refinement, which reduces proof effort and makes it possible to systematically refine the model to a specific signalling configuration (or a protocol).

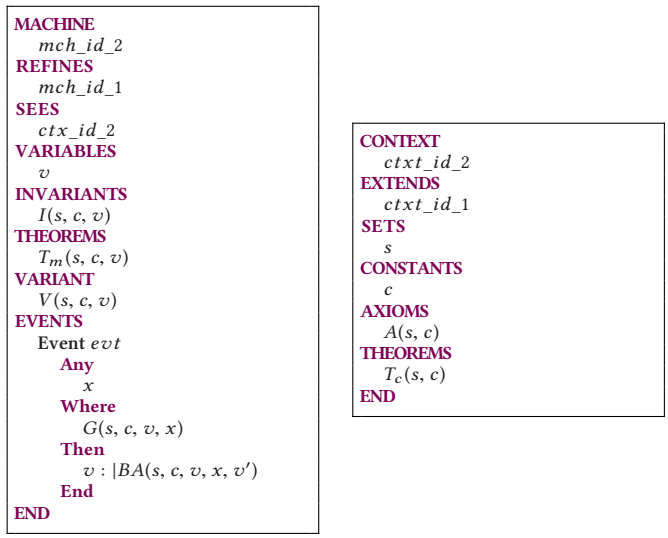
Article Structure. In Section 2, we provide preliminary information about the Event-B specification language and Event-B modelling patterns of hybrid systems. Section 3 describes the proposed method for a formal development of cyber-physical railway signalling systems. Section 4 semi-formally specifies the communication-based railway signalling system, which will be formally developed with Event-B. In this section, we provide a description of each signalling subsystem, signalling communication protocol and rolling stock model specifications. In Section 5, we discuss the process of formally modelling and verifying communication-based railway signalling system Event-B model. The final section discusses modelling and verification challenges of this work, and outlines future work directions.

2 BACKGROUND

In this section, we provide preliminary information about the Event-B modelling language, which is at the core of our proposed formal development method of cyber-physical railway signalling systems. This section also describes Event-B modelling patterns of hybrid systems that were used in a development of a hybrid train model.

2.1 Event-B

The Event-B mathematical language used in the system development and analysis is an evolution of the classical B method [1] and Action Systems [5]. The formal specification language offers a fairly high-level mathematical language based on a first-order logic and Zermelo-Fraenkel set



Listing 1. Event-B Model Structure.

theory. The formalism belongs to a family of state-based modelling languages where a state of a discrete system is simply a collection of variables and constants whereas the transition is a guarded variable transformation.

A cornerstone of the Event-B method is the step-wise development that facilitates a gradual design of a system implementation through a number of correctness preserving refinement steps. The model development starts with a creation of a very abstract specification and the model is completed when all requirements and specifications are covered. The Event-B model is made of two key components—machines and contexts that, respectively, describe dynamic and static parts of the system (see Listing 1). The context contains modeler declared constants and associated axioms that can be made visible in machines. The dynamic part of the model contains variables that are constrained by invariants and initialised by an action. The state variables are then transformed in guarded events using a ***before-after predicate*** **(BAP)** linking the state of the variable before and after the event. The event's guard determines when the event may be triggered.

The Event-B method is a proof driven specification language where model correctness is demonstrated by generating and discharging proof obligations—theorems in first-order logic and set theory. Table 1 shows the key proof obligations associated to any Event-B model, and constructed from that model's **axioms** (A), **theorems** (T), **invariants** (I), **guards** (G), **variants** (V), and BAP. The model is considered to be correct when all proof obligations are discharged.

Rodin [33] is an open source, Eclipse-based, **integrated development environment (IDE)** for Event-B model development. The Rodin is a core set of plug-ins for project management, formal development, syntactic analysis, proof assistance and proof-based verification. Moreover, it also allows for extension points to support a range of additional plugins to provide different functionalities and features related to model checking, animation, code generation, additional proof capabilities (including calls to SMTs or external provers such as Why3 or Isabelle), composition and decomposition, refactoring framework, and model editors.

Even though, the Event-B mathematical language is expressive enough for a lot of useful mathematical concepts it is still desirable to allow users extending the language. For that reason a theory extension process has been developed and realized as a Rodin platform plug-in. With the theory extension approach, new theories, which include datatypes, operators, and proof rules, can be defined and proved to be sound through generated proof obligations.

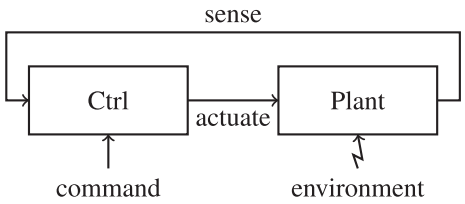


Fig. 1. Generic hybrid system representation.

Table 1. Event-B Proof Obligations

Theorems	$A(s,c) \implies T_c(s,c)$
Invariant Preservation	$A(s,c) \land I(s,c,v) \land G(s,c,v,x) \land BAP(s,c,v,x,v') \implies I(s,c,v')$
Event Feasibility	$A(s,c) \land I(s,c,v) \land G(s,c,v,x) \implies \exists v' \cdot BAP(s,c,v,x,v')$
Variant Progress	$A(s,c) \land I(s,c,v) \land G(s,c,v,x) \land BAP(s,c,v,x,v') \implies V(s,c,v') < V(s,c,v)$

2.2 Modelling Hybrid Systems in Event-B

Event-B is particularly adapted to the development of discrete systems, thanks to its semantics. Since its expression language is based on first-order logic and set theory, however, it is possible to express continuous behaviours, as high-level mathematical constructs, and handle them in Event-B models.

The approach of Dupont et al. [15, 16] takes advantage of this to propose a generic and reusable *modelling framework* for designing hybrid systems in Event-B.

2.2.1 Continuous Variables. Discrete variables in Event-B models are associated with instantaneous, point-wise assignment (the *before-after predicate*). Continuous variables, however, represent a continuous behaviour on a specific time interval, rather than on a single point. For this reason, continuous variables (denoted x_p in the following) are modelled using *functions of time*, and they are updated using the special ***continuous before-after predicate*** **(CBAP)** operator:

$$x_p :|_{t \to t'} \mathcal{P}(x_p, x_p') \;\&\; H \equiv \; [0,t[\lhd x_p' = [0,t[\lhd x_p \qquad (PP)$$
$$\land\; \mathcal{P}([0,t] \lhd x_p, [t,t'] \lhd x_p') \qquad (PR)$$
$$\land\; \forall t^* \in [t,t'], x_p(t^*) \in H. \qquad (LI)$$

Informally, the continuous before-after predicate allows to describe a change in the behaviour of a continuous variable so that:

- the variable's "past," i.e., its value on $[0,t[$, is preserved (*PP*),
- the variable's behaviour on $[t,t']$ is given by a predicate (*PR*),
- the variable remains in an evolution domain H (*LI*).

2.2.2 Embedding Continuous Features in Event-B. Even though the Event-B mathematical language is expressive enough for a lot of useful mathematical concepts, it is still desirable to allow users extending the language. For that reason an extension process has been proposed, under the form of *theories* [13]. With the theory extension approach, new theories, which include data types, operators, and proof rules, can be defined and their correctness proved by discharging generated proof obligations.

Theories are used to define continuous constructs, required for hybrid system design, and, in particular, differential equations, as well as the continuous before-after predicate defined in Section 2.2.1. Listing 2 gives an excerpt of the theory defined and used throughout the Event-B development.

THEORY DiffEq
TYPE PARAMETERS E, F, ...
DATATYPE
DE(F) **constructors** **ode**($f \oftype \mathbb{R} \times F \pfun F, \eta_0 \oftype F, t_0 \oftype \mathbb{R}$), ...
OPERATORS
solutionOf predicate ($D \oftype \mathbb{P}(\mathbb{R})$, $\eta \oftype \mathbb{R} \pfun F$, $\mathcal{E} \oftype$ **DE**(F))
well-definedness condition $D \subseteq \mathrm{dom}(\eta)$
Solvable predicate ($D \oftype \mathbb{P}(\mathbb{R})$, $\mathcal{E} \oftype$ **DE**(F))
direct definition $\exists \eta \cdot \eta \in \mathbb{R} \pfun F \land D \subseteq \mathrm{dom}(\eta) \land$ **solutionOf**($D, \eta, \mathcal{E}$)
$:|_{t \to t'}$ predicate ($t, t' \oftype \mathbb{R}$, $x_p, x_p' : \mathbb{R} \pfun F$, $\mathcal{P} \oftype \mathbb{P}((\mathbb{R} \pfun F) \times (\mathbb{R} \pfun F))$, $H \oftype \mathbb{P}(F)$)
...

Listing 2. Differential equation theory excerpt.

In particular, it defines the following operator and expressions:

- **ode**(F, η_0, t_0) is an ordinary differential equation (ODE) $\dot{\eta}(t) = F(t, \eta(t))$ with initial condition $\eta(t_0) = \eta_0$.
- **DE**(S) is a set of differential equations with solutions valued in S (the continuous state space).
- **solutionOf**(D, η, eq), with $D \subseteq \mathbb{R}$, $\eta \in D \to S$ and $eq \in$ **DE**(S), is a predicate indicating that η is a solution of eq (on domain D).
- **Solvable**(D, eq) is a predicate indicating that there exists a solution to equation eq on domain D.

2.2.3 Generic Hybrid System Model. The core of the methodology for designing hybrid system models using Event-B is a so-called *generic* model, encompassing both the discrete and the continuous part of the model, following the diagram presented in Figure 1. In this pattern, the discrete controller (*Ctrl*) controls a physical phenomenon (*Plant*). It may detect variations of the plant's state through *sensing*, and may influence its behaviour with *actuations*. As a physical object, the plant is subject to its *environment*; meanwhile, the controller may be piloted through *user commands*.

The characteristics of this model are abstracted using *event parameters* (ANY clause). They can be given at any time during *refinement* using witnesses (instantiation).

Variables and State Spaces. A hybrid system is modelled through two variables: its discrete state x_{st} and its continuous state x_p. Discrete state x_{st} is taken in the set of possible discrete states, STATES, which usually correspond to the possible modes of the controller. Continuous state x_p is a function of time and valued in *state space S* (usually a real vector space). As we will need it in subsequent proofs or properties, we also model time with a single read-only variable (t). By convention, this variable evolves in the set of positive or null reals ($\mathbb{R}^+$), and starts at 0. Listing 3 gives the header of the generic model.

Discrete Behaviour. The behaviour of hybrid systems consists of a discrete and a continuous part. Discrete behaviour is captured by discrete events, that represent changes in the system's discrete state (i.e., of the discrete variables). Transition events model internal controller changes, while Sense events model controller changes induced by the evolution of the continuous part, as denoted by a guard involving the continuous state x_p (see Listing 4).

Continuous Behaviour. Continuous behaviour is described by continuous events. These events update the system's continuous variables using the *CBAP* operator (see Section 2.2.1), and add whole time intervals to the system's trace.

MACHINE Generic
VARIABLES t , x_{st} , x_p
INVARIANTS
inv_1 : $t \in \mathbb{R}^+$
inv_2 : $x_{st} \in \text{STATES}$
inv_3 : $x_p \in \mathbb{R} \nrightarrow S$
...

EVENTS
INITIALISATION
THEN
act_1 : $t := 0$
act_2 : $x_{st} :\in \text{STATES}$
act_3 : $x_p :\in \{0\} \rightarrow S$
END

Listing 3. Generic hybrid system model: Header.

Transition
ANY st
WHERE
grd_1 : $st \in \mathbb{P}1(\text{STATES})$
THEN
act_1 : $x_{st} :\in st$
END

Sense
ANY st , p
WHERE
grd_1 : $st \in \mathbb{P}1(\text{STATES})$
grd_2 : $p \in \mathbb{P}(\text{STATES} \times \mathbb{R} \times S)$
grd_3 : $(x_{st} \mapsto t \mapsto x_p(t)) \in p$
THEN
act_1 : $x_{st} :\in st$
END

Listing 4. Generic hybrid system model: Transition and sense.

Behave
ANY $\mathcal{P}$, t'
WHERE
grd_0 : $t' > t$
grd_1 : $\mathcal{P} \subseteq (\mathbb{R}^+ \nrightarrow S) \times (\mathbb{R}^+ \nrightarrow S)$
grd_2 : **Feasible**$([t, t'], x_p, \mathcal{P}, \top)$
THEN
act_1 : $x_p :|_{t \rightarrow t'} \mathcal{P}(x_p, x_p') \;\&\; \top$
END

Actuate
ANY $\mathcal{P}$, st , H , t'
WHERE
grd_0 : $t' > t$
grd_1 : $\mathcal{P} \subseteq (\mathbb{R}^+ \nrightarrow S) \times (\mathbb{R}^+ \nrightarrow S)$
grd_2 : **Feasible**$([t, t'], x_p, \mathcal{P}, H)$
grd_3 : $st \subseteq \text{STATES}$
grd_4 : $x_{st} \in st$
grd_5 : $H \subseteq S$
grd_6 : $x_p(t) \in H$
THEN
act_1 : $x_p :|_{t \rightarrow t'} \mathcal{P}(x_p, x_p') \;\&\; H$
END

Listing 5. Generic hybrid system model: Behave and actuate.

Continuous events are split in two categories: Behave events capture spurious changes and *perturbations* in the plant, and are used to model the environment. At this level of abstraction, Behave are unconstrained, as symbolised by the use of $\top$ (meaning the evolution domain is the whole space).

Actuate events model changes in the plant induced by the controller (i.e., actuation). Note, in particular, that actuation events are tied to a controller mode (see Listing 5).

To establish that the CBAP use is correct, we need to ensure predicate $\mathcal{P}$ is *feasible*, that is

$$\exists x_p' \cdot x_p' \in \mathbb{R}^+ \nrightarrow S \land [t, t'] \subseteq \text{dom}(x_p') \land ([0, t] \lhd x_p) \mapsto ([t, t'] \lhd x_p') \in \mathcal{P} \land \forall \hat{t} \cdot \hat{t} \in [t, t'] \Rightarrow x_p'(\hat{t}) \in H.$$

This is enforced by guard grd_2 using the **Feasible** predicate. Upon refining the event, *guard strengthening* will require that the provided parameters (H and $\mathcal{P}$) satisfy this predicate and thus that the instantiation provides feasible behaviours.

3 METHODOLOGY

This section describes a formal engineering methodology for the top-down development of cyber-physical railway signalling systems. In following sections, we describe two main steps of the methodology steps: system specification and a functional Event-B model development.

3.1 Methodology Overview

The proposed methodology is a two-stage formal development methodology with the Event-B formal specification language at its core (illustrated in Figure 2). The main outcome of the methodology is a formal Event-B model of a signalling system that is developed by refining a provided generic communication-based signalling Event-B model $\mathcal{M}$ with particular signalling system specifications (defined in the first step of the methodology). Safety system aspects are of the utmost importance and are ensured by mathematically proving an instantiated model with respect to user-defined safety requirements.

The proposed development methodology is built upon an established Event-B formal modelling environment. The development of the Event-B model of a cyber-physical signalling system is facilitated by the Rodin platform, which provides a graphical user interface to developing models, automated proof obligation generation, verification tools and other plug-ins. One of the essential Rodin extensions is the Theory plug-in [13], which makes it possible to extend the Event-B mathematical language with new mathematical theories. The developed Event-B model of a cyber-physical railway signalling depends on continuous system modelling theories and patterns introduced and implemented using the Theory plug-in by Dupont et al. [15]. One of our previous contributions [31] was developing an Event-B domain theory $\mathcal{T}$, which formally defines dynamics of rolling stock by utilising hybrid modelling theories [15] and the Theory plug-in. The communication modelling patterns were developed and implemented without using the Theory plug-in.

The second group of Rodin extensions our methodology relies upon are concerned with the verification and validation of a model. To support a deductive model verification, Rodin provides extensions to a number of automated provers (e.g., Reference [24]) that attempt to automatically discharge generated proof obligations. Models developed in the Rodin platform can also be validated and animated using the ProB toolset, which can be particularly useful in early modelling stages where it could be too onerous to deductively verify a model. Furthermore, deductively proving properties like deadlock-freedom or liveness in the Event-B setting can be challenging. Therefore, in some instances ProB can provide a more pragmatic model-checking-based solution for validating properties properties that are not concerned with safety. The current Rodin version does not provide a practical method for reasoning about stochastic system properties. To meet a requirement for our methodology, we utilised a well-known probabilistic model checker PRISM [21] and prove probabilistic properties outside of the Rodin environment.

By using the Event-B our methodology aims to demonstrate that the cyber-physical signalling system satisfies the formulated safety requirements.

3.2 System Specifications and Requirements

The generally accepted approach to any system formal development is starting a development process by informally describing system specifications and requirements. Therefore, the first development step in our methodology is defining system specifications and requirements concerned with the specific cyber-physical railway signalling system configuration.

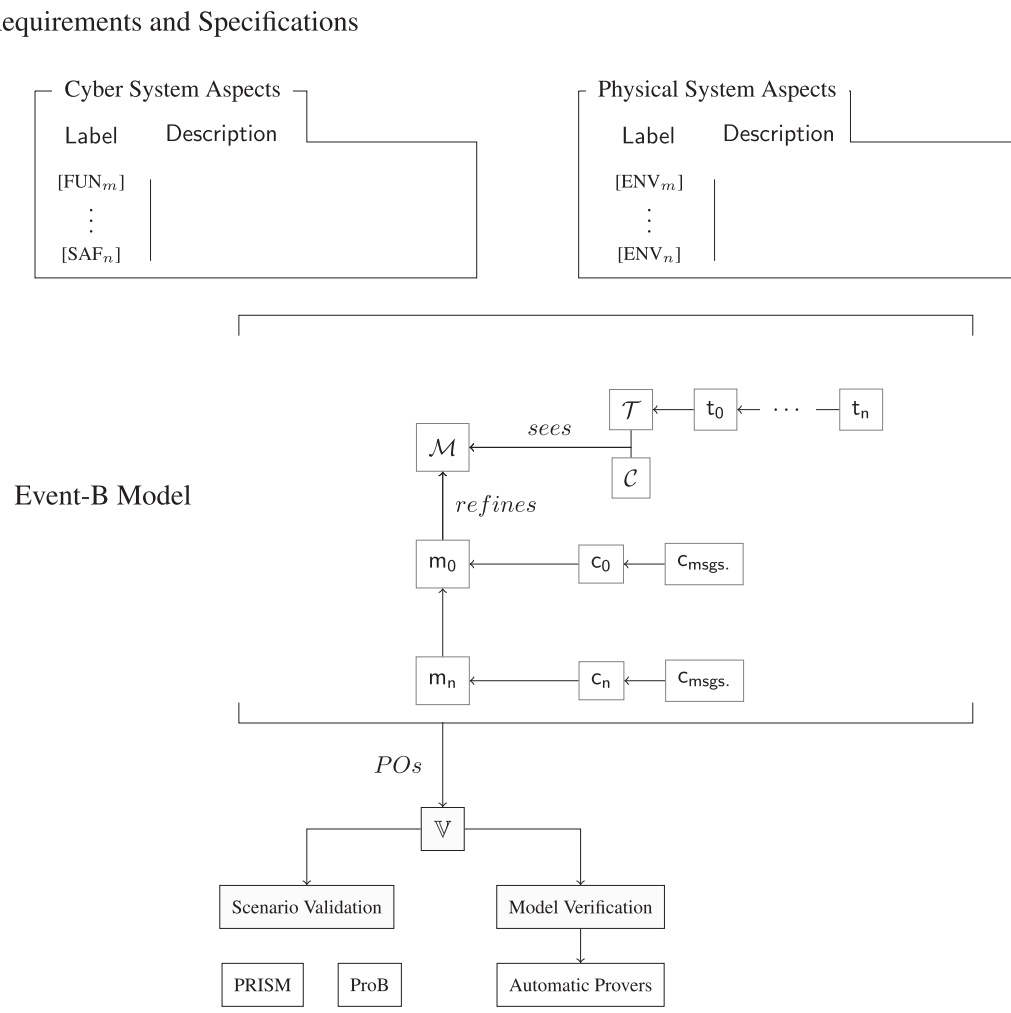


Fig. 2. Formal modelling and verification methodology of heterogeneous signalling systems.

At this formal development stage, the proposed methodology requires labelling system specifications and requirements. By labelling requirements and specifications, we intend to ensure a traceability between informal and formal methodology artefacts. The traceability aspect guarantees the completeness of the model, meaning, that all informal specifications and requirements have been captured by the formal model. On the formal Event-B artefact side, methodology requires to annotate events and invariants of the Event-B model with labelled references to a specific informal artefact (Figure 3). The Rodin platform tools like ProR [25] or others [18, 26] exist for ensuring traceability between formal and informal artefacts. However, both the ProR tool is longer supported and specification approach [18, 26] was not realised as the Rodin plug-in.

Furthermore, the cyber-physical railway signalling model we consider is a hybrid model, meaning, that system dynamics and physical aspects (e.g., rolling stock properties) must be defined. The generic theory of train dynamics we provide in a model theory $\mathcal{T}$ extension is parametrised and can be instantiated with specific (e.g., train mass, rail resistance) constants.

event_label $\widehat{=}$ $[specification_{id}]$
ANY
parameters
WHEN
guards
THEN
actions

INVARIANTS
invariant_label invariant $[requirement_{id}]$

Fig. 3. An example of an annotated Event-B model (event and invariant) in the Rodin tool.

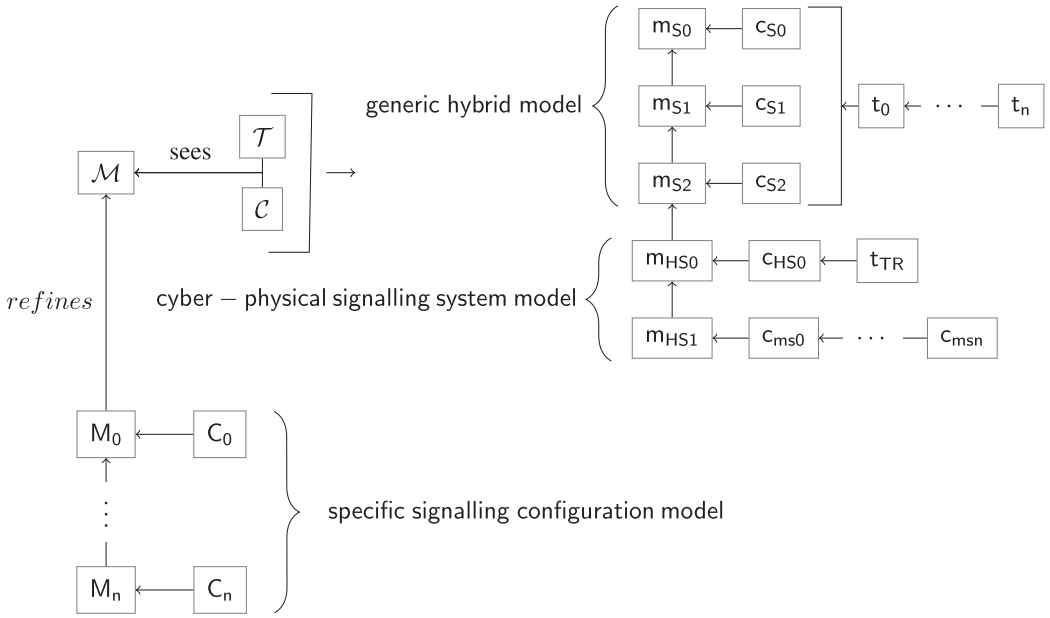


Fig. 4. The main activity of the proposed methodology is refining a generic signalling Event-B model $\mathcal{M}$ with cyber-physical signalling system modelling machines $M_{0..n}$ and context $C_{0..n}$ models.

3.3 Event-B Model

The formal development of a cyber-physical signalling system Event-B model is the central activity of the methodology. For the formal modelling, the methodology provides an Event-B model of cyber-physical railway signalling system, railway related Event-B *theories* and Event-B patterns for modelling protocols. To derive a cyber-physical signalling system Event-B model, a developer is required to refine a provided generic Event-B railway model with specific signalling system configurations by using the Event-B refinement mechanism (depicted in Figure 4).

The generic railway signalling model provided by our methodology includes a new Event-B theory ($\mathcal{T}$ in Figure 4), which defines continuous rolling stock dynamics with a parameterised first-order non-linear differential equation. Furthermore, the railway model also provides the Event-B context model (C in Figure 4) with a pre-defined train stopping distance function, constants related to the engine traction effort and parameters for the equation of train dynamics. The context model of the rolling stock can be further extended by specifying new parameters of the train physical model and updating train speed controller modes.

The proposed development methodology enables extending a generic signalling system Event-B context model that introduces signalling sub-systems and field elements (e.g., communication

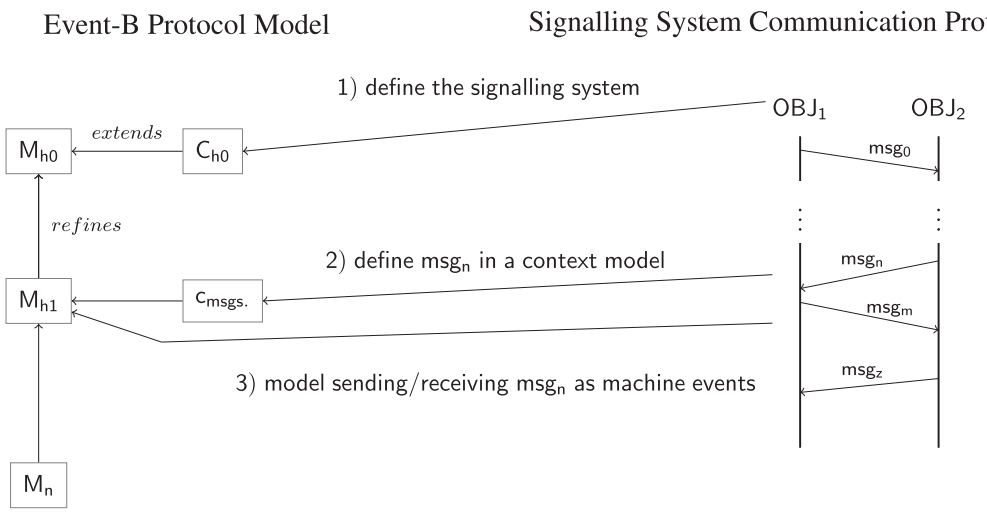


Fig. 5. The modelling process of a cyber-physical railway signalling communication protocol in Event-B.

centres, interlocking boxes, points) with signalling specifications defined in the first phase of the methodology. The system context model can be extended by including new axioms, for example, constraining the number of trains. The process of extending signalling context model with specific cyber-physical system aspects is visualised in Figure 5 as the first step.

To model an inter-subsystem communication of heterogeneous railway signalling systems, our methodology provides Event-B modelling patterns for a systematic modelling of protocols with Event-B. For example, patterns make it possible to introduce signalling protocol messages and capture message sending/receiving events in the Event-B model (steps 2 and 3 in Figure 5). Communication modelling patterns are templates for defining Event-B variables, machine events, and context models. In the actual Event-B model, templates are *instantiated*, or in other words, replaced with variables, events, and context models, which represent the actual system and adhere to the rules and structure of the template.

4 COMMUNICATION-BASED RAILWAY SIGNALLING SYSTEM

In this section, we describe an abstract hybrid railway signalling Event-B model, which can be refined to capture a specific signalling configuration. First, this section revisits a generalised communication-based railway signalling model, including major railway signalling sub-systems and their communication relations. The following section then describes continuous model aspects of the rolling stock that will be used to form a generic Event-B theory describing continuous behaviour of railway rolling stock.

4.1 Informal Communication-based Railway Signalling Model

As previously discussed, we base our railway signalling model on the radio-based communication and in-cab signalling systems, which generally contain three sets of objects: trains, interlocking boxes, and communication centres. On the infrastructure side, our railway model is made of railway tracks, which contain points (P1 in Figure 6) allowing trains to switch tracks and block markers ($M_{1..3}$ in Figure 6) for marking a spatial beginning and ending of railway sections (blocks).

The objective of the abstract railway signalling model is ensuring a safe spatial separation of trains and preventing train derailment by guaranteeing only locked point crossing. Our abstract

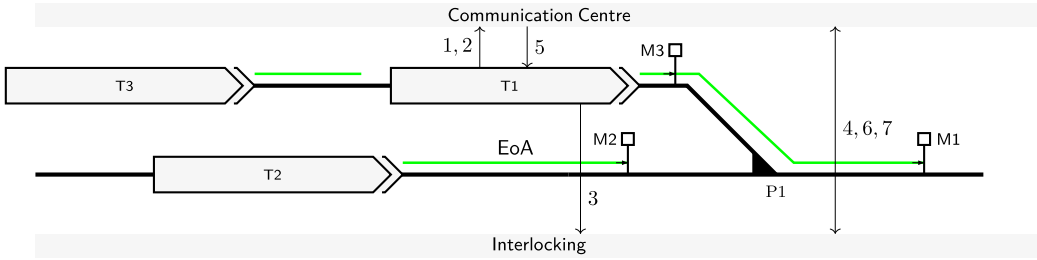


Fig. 6. An example of the abstract railway signalling model with three trains.

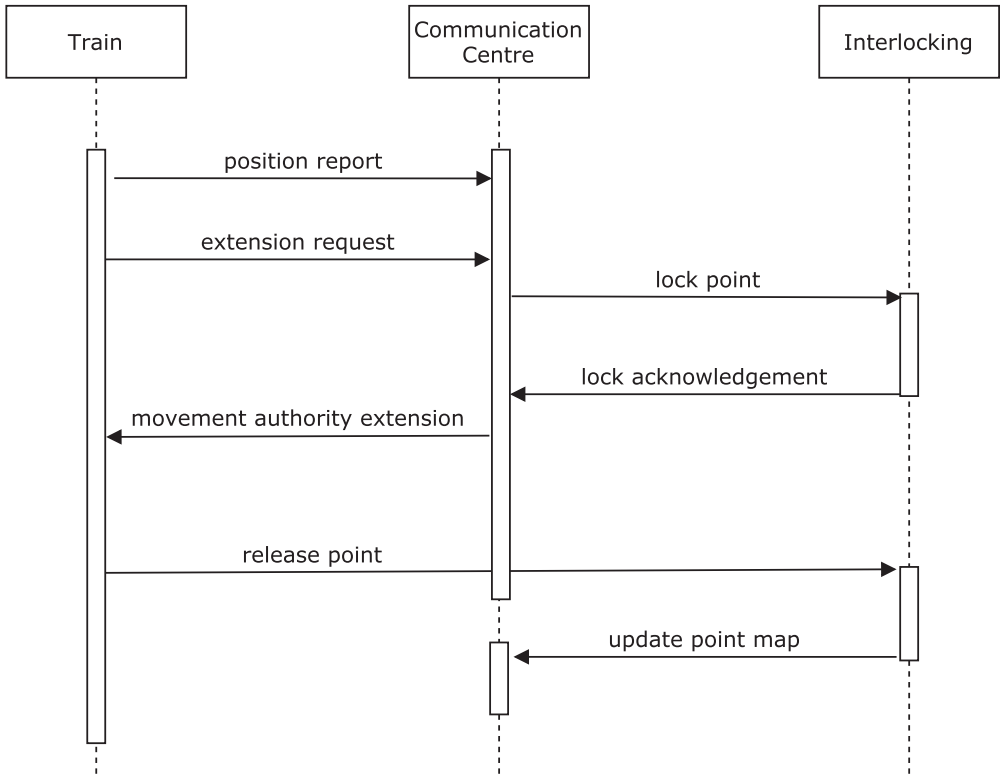


Fig. 7. Sequence diagram of the signalling model.

signalling model is based on a moving-block signalling principle with points protected with fixed-blocks to prevent a train derailment. In the following paragraphs, we specify the functionality of each object and communication relations with other objects (communication diagram of the signalling model is given in Figure 7).

Field Elements. Field elements are railway infrastructure elements that make it possible to detect and direct rolling stock as well as transmit information to the train drivers. Examples of field elements include:

— Points or switches (P_1 in Figure 7) are mechanical railway devices that make it possible for rolling stock to change tracks.

— Markers ($M_{1..3}$ in Figure 7) for marking a spatial beginning and ending of railway sections.
— Track clear detection systems (track circuits) are railway devices that make it possible to detect presence and absence of the trains.

Rolling Stock. In our signalling model trains are modelled by their physical (continuous) state as well as the mode (discrete) of their on-board computer. The physical state of a train evolves according to some differential equations with equation parameters controlled by the discrete mode of the on-board computer. The signalling model also assumes that a train is able to self-localise and communicate with other objects. In our abstract signalling model a train can send three types of messages:

(1) A position report message is sent periodically to the communication centre to update its position.
(2) A extension request message is sent to the communication centre when a train approaches the end of its movement authority (allowed travelling distance).
(3) Train sends a release message to the interlocking informing that it has left the junction area (crossed points).

Communication Centres. In the communication-based signalling systems, a communication centre is a pivotal object that manages a part of a railway network by interacting with rolling stock and interlocking boxes. A communication centre uses information received from trains and interlocking boxes to issue allowed travelling distances to individual trains. A centre contains and continuously updates an internal railway network map with junction locations (also their status: free or locked) and rolling stock positions. The model assumes that a communication centre knows the destination of each train, so points can be locked in the correct direction. The communication centre can send the following messages:

(1) When a communication centre receives an extension request message from a train with extension path requiring locking railway points. A communication centre sends a lock point message to the interlocking (if that point status is free) to set a point to the right direction and lock it.
(2) Once a point has been locked (or else it was not necessary) a communication centre sends a movement authority extension message to the train containing a permitted travelling distance, which is computed by considering other trains and point positions.

Interlocking Boxes. An interlocking is a safety-critical object responsible for authorising rolling stock and infrastructure movement of infrastructure (e.g., points). In our signalling model, the function of an interlocking box is guaranteeing safety by preventing trains crossing the same junction at the same time (e.g., P1 in Figure 6). An interlocking can send the following messages to a communication centre:

(1) An interlocking sends a lock acknowledgement message when it receives a lock point message, to inform the communication centre that a point has been adjusted and locked.
(2) An interlocking sends an update free map message to a communication centre when it received a releasepoint message from a train indicating that a train left the locked junction and now it can be set to free.

4.2 Continuous Rolling Stock Model

In this section, we describe the mathematical model of rolling stock continuous dynamics, which will be used as a basis to modelling hybrid train dynamics in Event-B. The section begins by deriving a realistic acceleration differential equation from fundamental laws of physics and models

Table 2. Safety and Physical Requirements of the Train

SAF_1 |	The train must remain within the issued movement authority at all times.
SAF_2 |	The train's speed must remain positive at all time.
SAF_3 |	A point must be reserved only to a single train and locked when train is passing over.
PHY_1 |	Traction force will remain within the minimum and maximum interval.

of rolling bodies. In the following paragraph, we describe a hybrid train speed controller model, which will be used in the generic railway signalling Event-B model.

A driver or an automated train operation system can only control the train engine power (tractive force f), which eventually yields an acceleration, and is bounded by two constants f_{min} and f_{max}. From Newton's second law, we know that acceleration is proportional to a net force (tractive engines force) applied to the mass of that object. The train must also overcome a resistance force, which acts in the opposite direction to engines traction force and thus a total engines tractive force can be expressed as the difference between two forces. The total rolling stock resistance is comprised of the mechanical and air resistances, and commonly expressed as a second-order polynomial (Davis Resistance equation $R_{tot.}(t)$ in Equation (1)), where a, b, c are fixed parameters and v(t) is the speed of a train at time t [29]:

$$\begin{cases} \dot{tv}(t) &= f - (a + b \cdot tv(t) + c \cdot tv(t)^2), \\ \dot{tp}(t) &= tv(t). \end{cases} \tag{1}$$

The train speed controller we consider is repeatedly issued with the **end of movement authority (EoA)**, which is updated discretely in the time by the communication centre. We assume that the speed controller is able to sense its distance to EoA and, in particular, determine if with a given current speed and acceleration it can stop before EoA. The stopping distance calculus is generally done by a complex algorithm on the on-board computer, whereas in our train model, we abstract the algorithm by a **stopping distance function (StopDist)**, which takes the current acceleration and speed as parameters, and returns the distance needed by the train to stop.

The model we developed can be split in two parts. The first part captures the role of the on-board train computer, responsible for the train speed supervision. Particularly in the moving block signalling systems, the absence of *head to tail* train collisions rely not only on correctly issued movement authority but also on-board speed controller. One can express the safety property as follows: **At all times the train must remain within the issued movement authority**. Additionally, the system also observes some specific physical properties; in particular, a train cannot go backward, meaning its speed must remain positive (or null) all time (refer to Table 2).

5 EVENT-B MODEL OF A CYBER-PHYSICAL RAILWAY SIGNALLING SYSTEM

This section describes the formal Event-B model development of a cyber-physical railway signalling. First, the section revisits our previous work [31] on a formal development of the hybrid Event-B model of the rolling stock model for which we used hybridised Event-B (described in Section 2.2). The following section describes the Event-B model of a cyber-physical railway signalling system that was developed by refining of the hybrid rolling stock Event-B model. The refined Event-B model formally models other signalling sub-systems (e.g., communication centres, interlocking boxes), infrastructure elements and a signalling communication protocol. In this refinement level, we prove that the issued movement authority distance to rolling stock is ensures safe rolling stock separation and prevents derailment.[1]

[1]The complete Event-B model can be found online at http://stankaitis.uk/2019/06/faoc/.

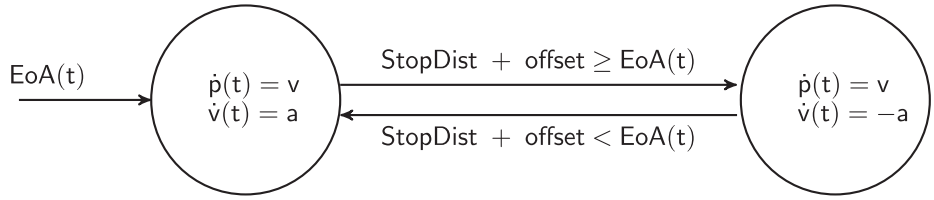


Fig. 8. An abstract train speed controller hybrid automata model with two modes.

THEORY Trains
OPERATORS
DavisResistance *expression* $(a : \mathbb{R},\ b : \mathbb{R},\ c : \mathbb{R})$
well–definedness condition $a \geq 0,\ b \geq 0,\ c \geq 0$
direct definition $(\lambda v \cdot v \in \mathbb{R} \mid a + bv + cv^2)$
DavisFunction *expression* $(a : \mathbb{R},\ b : \mathbb{R},\ c : \mathbb{R},\ f : \mathbb{R})$
well–definedness condition $a \geq 0,\ b \geq 0,\ c \geq 0$
direct definition
$(\lambda t \mapsto (v \mapsto p) \cdot t \in \mathbb{R}^+ \wedge (v \mapsto p) \in \mathbb{R}^2 \wedge v > 0 \mid f - \text{DavisResistance}(a, b, c)(v) \mapsto v) \cup$
$(\lambda t \mapsto (v \mapsto p) \cdot t \in \mathbb{R}^+ \wedge (v \mapsto p) \in \mathbb{R}^2 \wedge v \leq 0 \mid 0 \mapsto 0)$
DavisEquation *expression* $(a : \mathbb{R},\ b : \mathbb{R},\ c : \mathbb{R},\ f : \mathbb{R},\ t_0 : \mathbb{R}^+,\ v_0 : \mathbb{R},\ p_0 : \mathbb{R})$
well–definedness condition $a \geq 0,\ b \geq 0,\ c \geq 0$
direct definition ode(DavisFunction$(a, b, c, f), (v_0 \mapsto p_0), t_0)$
THEOREMS
. . .
DavisSolvability : $\forall a, b, c, f, t_0, p_0, v_0 \cdot a \in \mathbb{R} \wedge b \in \mathbb{R} \wedge c \in \mathbb{R} \wedge f \in \mathbb{R} \wedge t_0 \in \mathbb{R}^+ \wedge p_0 \in \mathbb{R} \wedge v_0 \in \mathbb{R} \wedge a \geq 0 \wedge b \geq 0 \wedge c \geq 0 \Rightarrow \textbf{Solvable}([t_0, +\infty[, \mathit{DavisEquation}(a, b, c, f, t_0, v_0, p_0))$
END

Listing 6. Extract of the Train domain theory.

5.1 Event-B Model of a Cyber-physical Railway Signalling System: Rolling Stock

The train speed controller controls the engine power (f) of the train, depending on its status (position, speed, acceleration) and, in particular, its distance to the EoA.

The train speed controller has two modes: free mode and restricted mode. If the stopping distance (plus a safety offset) of the train is shorter than EoA, then the train is said to be in free mode and it can choose arbitrary values for f. Whenever the stopping distance (+offset) of the train becomes greater than EoA, the train enters restricted mode, and the controller is required to provide values for f such that it can stop before EoA.

The train speed controller hybrid automaton model is visualised in Figure 8. In the following sections, we present a formal Event-B implementation of the informally presented hybrid signalling model.

5.1.1 Train Domain Theory. Common properties of the train are gathered in the Train *domain theory*, an extract of which is given in Listing 6. This theory mainly defines the *Davis equation* (DavisEquation) of coefficient a, b, and c and for a traction force of f, with initial condition $p(t_0) = p_0$ and $v(t_0) = v_0$. This equation corresponds to Equation (1).

CONTEXT TrainCtx
CONSTANTS
free_move, restricted_move
StopDist
$a, b, c, f_{min}, f_{max}, f_{dec\_min}$
AXIOMS
axm_1 : $a, b, c \in \mathbb{R}^+$
axm_2 : $f_{min}, f_{max}, f_{dec\_min} \in \mathbb{R}$
axm_3 : $\text{StopDist} \in (\mathbb{R} \times \mathbb{R}^+) \nrightarrow \mathbb{R}^+$
axm_4 : $\text{StopDist}(0 \mapsto 0) = 0$
axm_5 : partition(STATES, {free_move}, {restricted_move})
⋮
END

Listing 7. Extract of the TrainCtx context.

The theory also defines a few properties, that are useful when proving the models. In particular, a theorem is given that allows to deduce solvability of the Davis equation, which is crucial in establishing that the dynamics of the system are always *feasible*.

5.1.2 Train Model Static Informations. In addition to the train's dynamics, we gathered model-specific information in the TrainCtx context (see Listing 7). The context defines several constants of the system, as well as constraints on them. In particular, Davis coefficients are given (a, b, c), as well as the bounds on the train's traction power (f_{min}, f_{max}).

Furthermore, we define a train stopping distance function StopDist as a function of the current speed and acceleration with associated function constraining axioms. Finally, we define train controller modes free_move and restricted_move by refining the STATES set with an enumerated set.

5.1.3 Hybrid Rolling Model. In the first refinement of the generic hybrid model, we introduce several new events that instantiate generic events presented in Section 2.2. Because of the similarity of events, we only provide a single event for each of the generic event type. As specified in the previous subsection, in this refinement, we model the speed controller where the end movement authority is regularly updated, without specifying exactly how it is issued at this level of abstraction.

Machine header. Listing 8 presents the train model header. The train model features five variables in addition to time. The train itself is modelled using its position, speed, and acceleration (tp, tv and ta, respectively), as well as its traction power (f). Additionally, the end of authority is modelled by a real variable, EoA. Each variable is associated to a number of constraints (invariants 1 to 4), plus a gluing invariant that links the concrete and abstract continuous states (inv_5). In addition, both safety requirements proposed in Section 4.2 (invariants saf_1 and saf_2).

Discrete events. The Transition_restricted_move (see Listing 9) event models the change in the speed controller by adjusting trains traction effort when the train is in the restricted move mode. The event is simply guarded by a single predicate that enables event if and only if the status variable x_{st} is set to restricted_move. To control train's speed, we created a variable f that denotes the traction force and is modified by the action such that the stopping distance would not

MACHINE TrainMach **REFINES** Generic
VARIABLES t, x_{st} tp, tv, ta, f, EoA
INVARIANTS
inv_1: $tp, tv, ta \in \mathbb{R} \nrightarrow \mathbb{R}$
inv_2: $[0, t] \subseteq \text{dom}(tp), \ldots$
inv_3: $\text{EoA} \in \mathbb{R}^+$
inv_4: $f_{min} \leq f \wedge f \leq f_{max}$
inv_5: $x_p = [ta\ tv\ tp]^\top$
saf_1: $\forall t^* \cdot t^* \in [0, t] \Rightarrow tp(t^*) \leq \text{EoA}$
saf_2: $\forall t^* \cdot t^* \in [0, t] \Rightarrow tv(t^*) \geq 0$

Listing 8. Train model header.

Transition_restricted_move
REFINES Transition
WHERE
grd_1: $x_{st} = \text{restricted\_move}$
WITH
st: $st = \{\text{restricted\_move}\}$
THEN
act_1: $f :| tp(t) + \text{StopDist}(f' \mapsto tv(t)) \leq \text{EoA}$
END

Listing 9. Example of Transition event: calculating traction power f in restricted mode.

Sense_to_restricted
REFINES Sense
WHERE
grd_1: $tp(t) + \text{StopDist}(ta(t) \mapsto tv(t))) \geq \text{EoA}$
WITH
st: $st = \{\text{restricted\_move}\}$
p: $p = \text{STATES} \times \mathbb{R} \times \{v^* \mapsto p^* \mid p^* + \text{StopDist}(f_{dec\_min} \mapsto v^*) \geq \text{EoA}\}$
THEN
act_1: $x_{st} \coloneqq \text{restricted\_move}$
END

Listing 10. Example of Sense event: controller going into restricted mode.

overshoot the end of the movement authority. One must then prove an open proof obligation that such traction force value can be found.

Another internal controller event that changes controllers mode based on the input from the plant is sense event—Sense_to_restricted (see Listing 10). One of the two sense events will change the train state variable x_{st} if the end of movement authority has not been extended and train must decelerate to remain within issued movement authority.

Continuous Events. The Actuate_move event (see Listing 11) is the main continuous event of the model. It models the dynamics of the train, using the CBAP operator (see Section 2.2.1) together with the Davis equation defined in the Train theory (see Section 5.1.1). The proposed evolution

Actuate_move **REFINES** Actuate
ANY t'
WHERE
grd_1 : $tp(t) + \text{StopDist}(ta(t) \mapsto tv(t)) \leq \text{EoA}$
grd_2 : $t < t'$
WITH
x'_p : $x'_p = [ta\ tv\ tp]^\top$
$\mathcal{P}$: $\mathcal{P} = \ldots$
H : $H = \ldots$
st : $st = \text{STATES}$
THEN
act_1 : $ta, tv, tp :|_{t \to t'}$
solutionOf$([t, t'], [tv\ tp]^\top, \text{DavisEquation}(a, b, c, f, t, tv(t), tp(t))) \wedge$
$ta = \dot{tv}$
$\&tp + \text{StopDist}(ta \mapsto tv) \leq \text{EoA} \wedge tv \geq 0$
END

Listing 11. Event updating train's plant (actuate type).

domain ensures that (1) the train remains before the end of authority, and (2) the train's speed remains positive, in accordance with the system's safety invariants.

Note that the parameters of the generic model are refined using *witnesses* (`WITH` clause) as to match the CBAP of act_1 and the gluing invariant (inv_5).

5.2 Event-B Model of a Cyber-physical Railway Signalling System: Other Sub-Systems

To introduce other sub-systems and a communication protocol of the signalling system, the train speed controller model was further refined based on communication modelling patterns [32] (and diagram visualised in Figure 5) by introducing new context and machine models. According to the patterns, we first created a new context model (Listing 12) where new finite carrier sets representing trains (TRN), communication centres (CCS), interlocking boxes (IXL), and points (PNT) with their status were introduced. In the following paragraphs, we describe a part of the cyber-physical railway signalling model, which captures a train requesting movement authority extension and a communication-centre responding.

Following the message modelling pattern (see Section IV.B of Reference [32]), for each type of protocol message, an individual context will be introduced that defines the new message type with constant functions defining source, destination and *value* of the message. In context excerpt (Listing 13), the ma_extension message is defined, which is sent by a communications centre to a train (movement_authority_extension in Figure 7) with an updated movement authority. The constant (surjective) functions are used to select an appropriate message, which includes a source, destination and value of the message. In the case of movement_authority_extension message: ranges of source exts (axm_1) and destination extd (axm_2) functions are, respectively, communication centre and range train sets, while the range of value function extv is a set of real numbers (axm_3).

In the dynamic model (machine) part, we introduce communication channels for the new messages, which includes one for the movement_authority_extension message ext (see Listing 14). The model excerpt also contains another channel req for requesting movement authority extension messages and a variable reqt. The latter variable is simply used to track sent messages *locally*

CONTEXT CommunicationCtx
SETS
TRN, CCS, IXL, PNT, PSTATUS
CONSTANTS
FREE, RESERVED
AXIOMS
$axm_{1..4}$: finite(TRN, CCS, IXL, PNT)
axm_5 : partition(PSTATUS, {FREE}, {RESERVED})

Listing 12. Communication signalling model context.

CONTEXT ma_extension
SETS
EXT
CONSTANTS
exts, extd, extv
AXIOMS
axm_1 : $exts \in EXT \rightarrow\!\!\!\!\!\rightarrow CCS$
axm_2 : $extd \in EXT \rightarrow\!\!\!\!\!\rightarrow TRN$
axm_3 : $extv \in EXT \rightarrow\!\!\!\!\!\!\rightarrow \mathbb{R}^+$
axm_4 : $\forall s, d, v \cdot s \in CCS \land d \in TRN \land v \in \mathbb{R}^+ \Rightarrow \exists m \cdot exts(m) = s \land extd(m) = d \land extv(m) = v$

Listing 13. Movement authority extension message context.

MACHINE SignallingModel
SEES CommunicationCtx
VARIABLES
ext, req, reqt ...
INVARIANTS
inv_1 : $ext \subseteq EXT$
inv_2 : $req \subseteq REQ$
inv_3 : $reqt \in TRN \rightarrow \mathbb{P}(REQ)$
⋮
END

Listing 14. Movement authority channel variables for messages.

as messages are added and removed from the channel. Once communication channel variables are introduced, events can be refined or new events introduced to capture the process of message sending and receiving.[2]

The Trains_sense_to_restricted event (see Listing 15) which models the controller state change once the train enters the area where it must decelerate to stay within the movement authority. For a train to continue to travel, it must request the movement authority extension by sending a message to the communication centre. We model that by refining the Sense_to_restricted event by first adding additional event parameters denoting a train tr, communication centre rb, and a request message rq. Then, we include new guards $grd_{2..6}$, which define variable types and state

[2]Note: new continuous variables, replacing f, tp, ta, tv have been introduced to capture multiple trains in the system ntp, nta, ntv

Trains_sense_to_restricted
REFINES Sense_to_restricted
ANY
tr, cc, rq
WHERE
grd_1 : tr ∈ TRN
grd_2 : ntp(tr)(t) + StopDist(nta(tr)(t) ↦ ntv(tr)(t))) ≥ nEoA(tr)
grd_3 : cc ∈ CCS
grd_4 : rq ∈ (REQ \ req)
grd_5 : reqd(rq) = cc
grd_6 : reqs(rq) = tr
grd_7 : reqv(rq) = ntp(tr)(t)
THEN
act_1 : nx_s(tr) ≔ restricted_move
act_2 : req ≔ req ∪ {rq}
act_3 : reqt(tr) ≔ reqt(tr) ∪ {rq}
END

Listing 15. Refined event for requesting *EoA* extension.

that rq must be a new message with destination to specific communication centre rb and source of the train of interest rq. The message rq also includes the current position of the train at time t (grd_7), which can be accessed via train position function ntp(tr). New actions of the events $act_{2..3}$ add the new message to the request message channel req and saves a sent message receipt locally.

To model movement_authority_extension message sending, we apply a reply message modelling pattern that adds a new message to the channel and removes the replied message. To avoid having a single *super-event*, it was decided to split message sending into two events that, respectively, model movement authority extension over a line and over a point. Listing 16 describes an event for movement extension over a line. First, the guards ($grd_{1..6}$) check if an adequate request rq message has been received and new extension message ex will be sent to the source train grd_5. The value of the reply message contains a distance value, which is shorter than a distance to the next train and point ($grd_{8..9}$), but greater than the current end of the movement authority (grd_7). The event actions simply create a new extension message and remove the responded message. A similar event was created to capture the second branch when a point must be locked by communicating with an interlocking.

The final event in the message exchange cycle for requesting movement authority extension is the refined event Trains_transition_eoa (Listing 17). Originally, this event was introduced in modelling train speed controller to abstractly represent the internally updated movement authority. To introduce communication aspect to this event, we extend it with additional guards to check if an extension message ex has been received to the specific train tr ($grd_{3..5}$). The new end of the movement authority value (nEoA) is constrained by the value of extension message (grd_6). The action of the event simply update trains end of authority variable EoA(tr) and deletes the extension message ex.

5.3 Event-B Model of a Cyber-physical Railway Signalling System: Proof Statistics

The generic model is proved once and for all, and in this work, we refined the abstract hybrid controller model. The hybrid railway signalling model is itself a reusable artefact, which could be

communication_centre_extension_1
ANY
rb, rq, ex, tr
WHERE
grd_1 : rb ∈ RBC
grd_2 : rq ∈ req
grd_3 : ex ∈ (EXT \ ext)
grd_4 : reqd(rq) = rb
grd_5 : extd(ex) = reqs(rq)
grd_6 : exts(ex) = reqd(rq)
grd_7 : extv(ex) > nEoA(reqs(rq))
grd_8 : extv(ex) < ppos(pmap(ntp(reqs(rq))))
grd_9 : ∀tr · tr ∈ TR ∧ tr ≠ reqs(rq) ⇒ extv(ex) < ntp(tr)(t)
THEN
act_1 : ext := ext ∪ {ex}
act_2 : req := req \ {rq}
END

Listing 16. Event for modelling communication centre reply to the extension_request message (line part).

Trains_transition_eoa
REFINES Transition_eoa
ANY
tr, ex
WHERE
grd_1 : tr ∈ TRN
grd_2 : ex ∈ ext
grd_3 : extd(ex) = tr
THEN
act_1 : nEoA(tr) := extv(ex)
act_2 : ext := ext \ {ex}
END

Listing 17. Refined transition event with updated *EoA*.

further refined to capture a specific signalling configuration (e.g., by defining a specific railway schema) or modelling railway protocols (e.g., signalling handover).

As discussed by Alur in Reference [3], the verification of hybrid systems remains a challenge. The verification approaches based on the reachability analysis aim to provide a fully automatic verification approach, but due to the real-valued variables, these approaches are limited to linear systems. In this and other related works, authors have tried to address the verification scalability problem, by developing alternative proof-based verification approaches. Still, as our verification results suggest (Table 3) and also similar works [4, 12, 15], the proof automation of hybrid Event-B models is still low and must be improved for more practical applications. In spite of improved verification tools, a refinement plan for hybrid models should be reconsidered. Perhaps, a top-down approach (particularly, for this model), where continuous system's aspects are introduced in later refinement steps is more suitable.

Table 3. Proof Statistics of the Event-B Model of Cyber-physical Railway Signalling

Refinement	PO Type	|POs|	Auto.	Inter.
Speed Controller		55	36	19
	Well-Definedness	12	12	0
	Guard Strengthening	11	11	0
	Invariant Preservation	18	10	8
	Feasibility	8	0	8
	Simulation	6	3	3
Communication and Signalling		85	71	14
	Well-Definedness	31	31	0
	Guard Strengthening	12	7	5
	Invariant Preservation	42	33	9
	Feasibility	0	0	0
	Simulation	0	0	0
Total		140	119	21

An important feature of this hybrid system development approach is the requirement of explicitly stating the system's dynamic properties. In our opinion, this problem is often overlooked and could lead to miscommunication between, for instance, control engineers and software engineers. With our proposed approach, a formal hybrid artefact is created, which can be used between different types of engineers. However, the approach currently requires some understanding of formal methods (e.g., mathematical syntax) and could benefit with connection to more visual widely used tool like Simulink or other similar tools via Functional Mock-up Interface [11].

6 CONCLUSIONS AND FUTURE WORK

In this research, we attempted to address the challenges of applying formal methods for a model-based development of cyber-physical systems. In particular, the research described in this article focused on a formal development and the safety verification of heterogeneous cyber-physical railway signalling systems. The difficulty of a model-based cyber-physical system development stems from the complex nature of cyber-physical systems, which have deeply intertwined physical processes, computation, and networking system aspects that have to be captured by a system-level formal model.

The overall objective of this research was to address the challenge of applying formal methods to designing cyber-physical railway signalling systems by formulating a formal development methodology of cyber-physical railway signalling systems. It was essential that the methodology not only reduces the effort of applying formal methods but also facilitates a rigorous and systematic model-based signalling system development.

The main technical objective of our work was developing a generic hybrid signalling model that could be further refined to capture a specific signalling configuration. Or to enable modelling and verification of signalling communication aspects, for instance, protocols such as the one used for a radio-block handover. Our model can be divided into three main parts. The generic hybrid model is refined with train-/railway-specific attributed, including the new continuous theory, which defines the train plant by the Davis equation and a context for the other aspects such as stopping distance function. The dynamic part of the model refines the generic events to capture the train speed controller. As stated before, the main invariants we need to prove are that the train must remain within its movement authority and that the plant variables must remain within their limits. One of the future work directions is to address the verification problem by exploiting the previously

developed Rodin verification extension based on the Why3 umbrella prover [24]. To achieve that, a new *real* theory together with definitions of operator used in this work could be re-defined in a new Why3 theory. Another direction for the hybrid model validation we would like to explore is co-simulation of discrete-continuous models via Functional Mock-up Interface [11].

REFERENCES

[1] J.-R. Abrial. 1996. *The B-book: Assigning Programs to Meanings*. Cambridge University Press, New York, NY.

[2] J.-R. Abrial. 2013. *Modeling in Event-B: System and Software Engineering*. Cambridge University Press, New York, NY.

[3] R. Alur. 2011. Formal verification of hybrid systems. In *Proceedings of the 9th ACM International Conference on Embedded Software (EMSOFT'11)*. ACM, New York, NY, 273–278. https://doi.org/10.1145/2038642.2038685

[4] G. Babin, Y. Aït-Ameur, S. Nakajima, and M. Pantel. 2015. Refinement and proof-based development of systems characterized by continuous functions. In *Proceedings of the International Symposium on Dependable Software Engineering: Theories, Tools, and Applications*. Springer, 55–70.

[5] R. J. R. Back. 1990. Refinement calculus, part II: Parallel and reactive programs. In *Stepwise Refinement of Distributed Systems Models, Formalisms, Correctness*, J. W. de Bakker, W. P. de Roever, and G. Rozenberg (Eds.). Springer, 67–93.

[6] Frédéric Badeau and Arnaud Amelot. 2005. Using B as a high level programming language in an industrial project: Roissy VAL. In *Proceedings of the Conference on Formal Specification and Development in Z and B (ZB'05)*, Helen Treharne, Steve King, Martin Henson, and Steve Schneider (Eds.). Springer, Berlin, 334–354.

[7] Radhakisan Baheti and Helen Gill. 2011. Cyber-physical systems. *The Impact of Control Technology* (2011), 161–166. www.ieeecss.org.

[8] Richard Banach, Michael Butler, Shengchao Qin, Nitika Verma, and Huibiao Zhu. 2015. Core hybrid event-B I: Single hybrid Event-B machines. *Sci. Comput. Program.* 105 (2015), 92–123.

[9] Patrick Behm, Paul Benoit, Alain Faivre, and Jean-Marc Meynadier. 1999. Météor: A successful application of B in a large project. In *Proceedings of the Conference on Formal Methods (FM'99)*, Jeannette M. Wing, Jim Woodcock, and Jim Davies (Eds.). Springer, Berlin, 369–387.

[10] Ulrich Berger, Phillip James, Andrew Lawrence, Markus Roggenbach, and Monika Seisenberger. 2018. Verification of the european rail traffic management system in real-time maude. *Sci. Comput. Program.* 154 (2018), 61–88. Formal Techniques for Safety-Critical Systems 2015.

[11] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clauß, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, D. Neumerkel, H. Olsson, J. v. Peetz, S. Wolf, Atego Systems Gmbh, Qtronic Berlin, Fraunhofer Scai, and St. Augustin. 2011. The functional mockup interface for tool independent exchange of simulation models. In *Proceedings of the 8th International Modelica Conference*.

[12] C. Bogdiukiewicz, M. Butler, T. S. Hoang, M. Paxton, J. Snook, X. Waldron, and T. Wilkinson. 2017. Formal development of policing functions for intelligent systems. In *Proceedings of the IEEE 28th International Symposium on Software Reliability Engineering (ISSRE'17)*. 194–204. https://doi.org/10.1109/ISSRE.2017.40

[13] Michael Butler and Issam Maamria. 2013. *Practical Theory Extension in Event-B*. Springer, Berlin, 67–81. https://doi.org/10.1007/978-3-642-39698-4_5

[14] A. Cimatti, M. Roveri, and S. Tonetta. 2009. Requirements validation for hybrid systems. In *Proceedings of the 21st International Conference on Computer Aided Verification (CAV'09)*. Springer-Verlag, 188–203.

[15] Guillaume Dupont, Yamine Aït-Ameur, Marc Pantel, and Neeraj Kumar Singh. 2018. Proof-based approach to hybrid systems development: Dynamic logic and Event-B. In *Abstract State Machines, Alloy, B, TLA, VDM, and Z*, Michael Butler, Alexander Raschke, Thai Son Hoang, and Klaus Reichl (Eds.). Springer International Publishing, Cham, 155–170.

[16] Guillaume Dupont, Yamine Aït-Ameur, Neeraj K. Singh, and Marc Pantel. 2021. Event-B hybridation: A proof and refinement-based framework for modelling hybrid systems. *ACM Trans. Embed. Comput. Syst.* 20, 4 (To appear 2021). https://doi.org/10.1145/3448270

[17] ERTMS User Group. 2002. UNISIG: ERTMS/ETCS: System Requirements Specification v. 3.4.0.

[18] Fahad Rafique Golra, Fabien Dagnat, Jeanine Souquières, Imen Sayar, and Sylvain Guerin. 2018. Bridging the gap between informal requirements and formal specifications using model federation. In *Software Engineering and Formal Methods*, Einar Broch Johnsen and Ina Schaefer (Eds.). Springer International Publishing, 54–69.

[19] Alexandra Halchin, Yamine Aït Ameur, Neeraj Kumar Singh, Abderrahmane Feliachi, and Julien Ordioni. 2019. Certified embedding of B models in an integrated verification framework. In *Proceedings of the 13th International Symposium on Theoretical Aspects of Software Engineering (TASE'19)*.

[20] Alexandra Halchin, Abderrahmane Feliachi, Neeraj Kumar Singh, Yamine Aït Ameur, and Julien Ordioni. 2017. B-PERFect—Applying the PERF approach to B-based system developments. In *Proceedings of the 2nd International*

Conference on Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification (RSSRail'17). 160–172.

[21] Andrew Hinton, Marta Kwiatkowska, Gethin Norman, and David Parker. 2006. PRISM: A tool for automatic verification of probabilistic systems. In *Tools and Algorithms for the Construction and Analysis of Systems*, Holger Hermanns and Jens Palsberg (Eds.). Springer, Berlin, 441–444.

[22] Jochen Hoenicke and Ernst-Rüdiger Olderog. 2002. CSP-OZ-DC: A combination of specification techniques for processes, data and time. *Nord. J. Comput.* 9, 4 (2002), 301–334.

[23] IEEE Std 1474.1-2004. 2005. IEEE standard for communications-based train control (CBTC) performance and functional requirements. https://standards.ieee.org/ieee/1474.1/3552/.

[24] Alexei Iliasov, Paulius Stankaitis, David Adjepon-Yamoah, and Alexander Romanovsky. 2016. Rodin platform why3 plug-in. In *Abstract State Machines, Alloy, B, TLA, VDM, and Z*, Michael Butler, Klaus-Dieter Schewe, Atif Mashkoor, and Miklos Biro (Eds.). Springer International Publishing, Cham, 275–281.

[25] Michael Jastram. 2010. ProR, an open source platform for requirements engineering based on RIF. In *Proceedings of the Systems Engineering Infrastructure Conference*.

[26] Michael Jastram, Stefan Hallerstede, Michael Leuschel, and Aryldo G. Russo. 2010. An approach of requirements tracing in formal refinement. In *Verified Software: Theories, Tools, Experiments*, Gary T. Leavens, Peter O'Hearn, and Sriram K. Rajamani (Eds.). Springer, Berlin, 97–111.

[27] A. Platzer. 2008. Differential dynamic logic for hybrid systems. *J. Autom. Reason.* 41, 2 (2008), 143–189.

[28] A. Platzer and J.-D. Quesel. 2009. European train control system: A case study in formal verification. In *Proceedings of the International Conference on Formal Engineering Methods*. Springer, 246–265.

[29] B. P. Rochard and F. Schmid. 2000. A review of methods to measure and calculate train resistances. *Proc. Inst. Mech. Eng., Part F: J. Rail Rapid Transit* 214, 4 (2000), 185–199. https://doi.org/10.1243/0954409001531306

[30] B. I. Silva, O. Stursberg, B. H. Krogh, and S. Engell. 2001. An assessment of the current status of algorithmic approaches to the verification of hybrid systems. In *Proceedings of the 40th IEEE Conference on Decision and Control*, Vol. 3. IEEE, 2867–2874.

[31] Paulius Stankaitis, Guillaume Dupont, Neeraj Kumar Singh, Yamine Ait-Ameur, Alexei Iliasov, and Alexander Romanovsky. 2019. Modelling hybrid train speed controller using proof and refinement. In *Proceedings of the 24th International Conference on Engineering of Complex Computer Systems (ICECCS'19)*. 107–113. https://doi.org/10.1109/ICECCS.2019.00019

[32] Paulius Stankaitis, Alexei Iliasov, Yamine Aït Ameur, Tsutomu Kobayashi, Fuyuki Ishikawa, and Alexander Romanovsky. 2019. A refinement-based method for developing distributed protocols. In *Proceedings of the IEEE 19th International Symposium on High Assurance Systems Engineering (HASE'19)*. 90–97.

[33] The RODIN platform. 2006. Retrieved from https://sourceforge.net/projects/rodin-b-sharp/files/Core_Rodin_Platform/.

[34] Liang Zou, Jidong Lv, Shuling Wang, Naijun Zhan, Tao Tang, Lei Yuan, and Yu Liu. 2014. Verifying chinese train control system under a combined scenario by theorem proving. In *Verified Software: Theories, Tools, Experiments*, Ernie Cohen and Andrey Rybalchenko (Eds.). Springer, Berlin, 262–280.

Received 25 December 2021; accepted 2 March 2022