Modelling hybrid train speed controller using proof and refinement

Paulius Stankaitis*, Guillaume Dupont†, Neeraj Kumar Singh†, Yamine Aït-Ameur†,
Alexei Iliasov*, Alexander Romanovsky*
*School of Computing, Newcastle University, Newcastle upon Tyne, UK
{p.stankaitis, alexei.iliasov, alexander.romanovsky}@newcastle.ac.uk
†INPT–ENSEEIHT, 2 Rue Charles Camichel, Toulouse, France
{guillaume.dupont, neeraj.singh, yamine }@enseeiht.fr

***Abstract*—The modern radio-based railway signalling systems aim to increase network's capacity by enabling trains to run closer to each other. At the core of such systems is train's on-board computer (discrete) responsible for computing and controlling the speed (continuous) of the train. Such systems are best captured by hybrid models, which capture discrete and continuous system's aspects. Hybrid models are notoriously difficult to model and verify, in our research we address this problem by applying hybrid systems' modelling patterns and stepwise refinement for developing hybrid train speed controller model.**

***Index Terms*—hybrid systems, railway signalling, Event-B, refinement, proofs**

I. Introduction

In the last few decades, modern signalling systems (e.g. [1] and [2]) have been proposed and progressively refined as a way to safely increase networks' capacity. In the radio-based signalling systems, trains regularly receive data about the distance they are permitted to travel (*movement authority*); an on-board computer then calculates the fitting speed profile for the train to comply to this authority. The whole system relies extensively on algorithms and computation (for calculating the speed profile), which being a part of the safety-critical system requires the highest level of trust. To achieve such a high level of safety assurance for those complex speed controlling systems, scenario-based testing methods are far from being sufficient, although very much used in the industry. On the other hand, formal methods have been successfully applied to the railway domain (e.g. [3], [4]) and seem quite fitting for dealing with programs.

For the highest level of assurance, trains should be modelled as *hybrid systems*, that is, systems that integrate both discrete and continuous types of behaviour. Modelling and certifying hybrid systems is known to be a difficult task [5] as standard formal methods often deal with purely discrete models. To overcome this, we chose to use the Event-B method, which proved successful in handling continuous concepts [6]–[8] while providing a convenient and well-adapted methodology, framework and tool to model those systems.

In this work we aim to develop a generic train speed controller model with a continuous train dynamics in the Event-B [9] formal modelling framework. The Event-B is a proof and stepwise based modelling method for developing *correct-by-construction* systems which has been used widely for modelling railway systems (e.g. [10]–[12]). However, continuous railway system aspects could not be modelled due to inability to extend the Event-B specification language with the additional theories. To overcome this issue the *Theory plug-in* [13] was developed which allowed developers to include additional theories. In a paper by Dupont et al. [7] authors utilized *Theory plug-in* and developed a modelling framework together with theories for modelling hybrid systems in Event-B. This paper presents an application of this method for developing hybrid railway signalling models. The generic hybrid railway signalling model is primarily based on specifications and requirements for European Train Control Systems [1].

Related Work In the work by Berger et al. [14] the authors use a real-time modelling approach to develop and verify the ERTMS model. A multifaceted formalism is introduced in [15] to reason about real-time systems with a case study on railway crossings. In the work by Cimatti et al. [16] 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. [17], [18] 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 [19]–[21]. These studies are more related to our work as the authors consider most of the signalling parts (only level crossings [15]) and validate systems through proofs. We would like to note here that our approach is based on the stepwise refinement, which reduces the proof effort and also enables to further refine the model for a specific signalling configuration (or a protocol).

The following section provides background information

on the Event-B specification language and theory extension process. In the Section III we recall the main modelling features of the modelling framework for hybrid systems in Event-B developed by Dupont et al. [7]. Section IV overviews the developed formal hybrid railway signalling model. The final sections discusses modelling and verification challenges, and outlines future work directions.

II. Background

A. Event-B

The Event-B mathematical language used in the system development and analysis is an evolution of the classical B method [22] and Action Systems [23]. The formal specification language offers a fairly high-level mathematical language based on a first-order logic and Zermelo-Fraenkel set theory as well as an economical yet expressive modelling notation. 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.

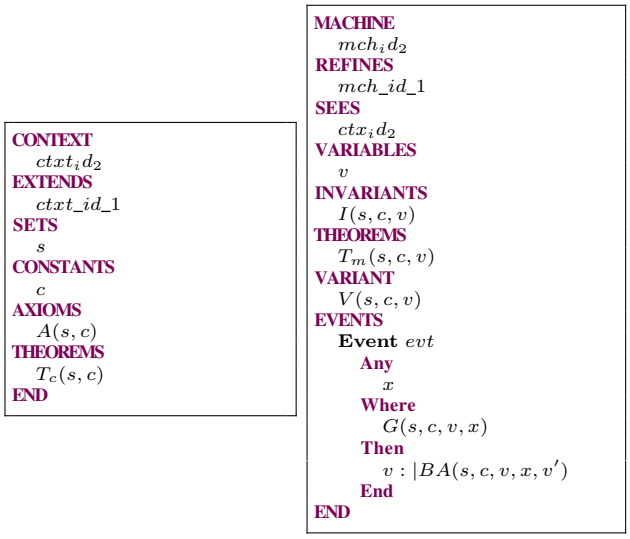

Fig. 1: Event-B Model Structure

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 which respectively describe dynamic and static parts of the system (see Fig. 1). The context contains modeler declared constants and associated axioms which can be made visible in machines. The dynamic part of the model contains variables which are constrained by invariants and initialised by an action. The state variables are then transformed by actions which are part of events and the modeler may use predicate guards to denote when event is triggered. Specifying a model is not sufficient, one must provide evidence about the correctness of the model as well. The Event-B method is a proof driven specification language where model correctness is demonstrated by generating and discharging proof obligations - theorems in the first-order logic. Table I shows the important proof obligations of the Event-B language. More details related to the modeling language and proof obligations can be found in [9]. The model is considered to be correct when all proof obligations are discharged.

Theorems	$A(s,c) \Rightarrow T_c(s,c)$ $A(s,c) \wedge I(s,c,v)$ $\Rightarrow T_m(s,c,v)$
Invariant preservation	$A(s,c) \wedge I(s,c,v)$ $\wedge G(s,c,v,x)$ $\wedge BA(s,c,v,x,v')$ $\Rightarrow I(s,c,v')$
Event feasibility	$A(s,c) \wedge I(s,c,v)$ $\wedge G(s,c,v,x)$ $\Rightarrow \exists v'.BA(s,c,v,x,v')$
Variant progress	$A(s,c) \wedge I(s,c,v)$ $\wedge G(s,c,v,x)$ $\wedge BA(s,c,v,x,v')$ $\Rightarrow V(s,c,v') < V(s,c,v)$

TABLE I: Proof Obligations

Rodin [24] 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 extension points for supporting a range of additional plugins to provide different functionalities and features related to model checking, animation, code generation, additional proof capabilities using SMTs and external theorem provers (i.e., Why3, Isabelle), UML-B, Theory plug-ins, composition and decomposition, refactoring framework, and model editors.

B. Extending Event-B with external theories

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[1]. 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.

A theory of real number is of particular interest for reasoning about hybrid systems properties. This work reuses the theory of dense reals originally developed by Abrial and Butler [25], and then extended by Dupont et al. [7] to support the theory of continuous functions, ordinary differential equations. etc.

III. Modelling Patterns for Hybrid Systems

In this section we overview the main modelling features of the framework for hybrid systems in Event-B. We first

[1]http://wiki.event-b.org/index.php/Theory_Plug-in#Standard Library

describe a theory for differential equations, then expose a generic hybrid model as well as a methodology for deriving a specific controller.

A. A Theory for Differential Equations

To handle a hybrid system we need to employ both discrete and continuous concepts. If discrete parts are in essence "natively" supported by Event-B, it is not really the case as for continuous features. To that extent, we make use of the theory plug-in presented in Section II-B.

```
THEORY
  TYPE PARAMETERS E, F
  DATA TYPES
    DE(F)
    CONSTRUCTORS
      ode(fun : ℙ(ℝ × F × F), initial : F, initialArg : ℝ)
  OPERATORS
    solutionOf <predicate> (D_R : ℙ(ℝ), η : D_R → F, eq : DE(F))
    Solvable <predicate> (D_R : ℙ(ℝ), eq : DE(F))
      direct definition
        ∃x · x ∈ (D_R → F) ∧ solutionOf(D_R, x, eq)
    ...
END
```

Fig. 2: Differential Equation Theory Snippet

Listing 2 gives an excerpt of the theory defined and used through out the Event-B development. It defines in particular:

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

B. A Generic Hybrid Model in Event-B

The methodology used to design hybrid system has been first presented in [7]. It is mostly based on the diagram presented in Figure 3 and relies on a generic Event-B model, aimed at being refined.

```
VARIABLES t, x_p, x_s
INVARIANTS
  inv1 : t ∈ ℝ+
  inv2 : x_s ∈ STATES
  inv3 : x_p ∈ ℝ+ℝ → S
EVENTS
  ...
  Progress
  THEN
    act1 : t :| t < t'
```

a) Variables and State Spaces: A hybrid system is modelled through two variables: its discrete state x_s (usually corresponding to some mode in a mode automaton) and its continuous state x_p, which is a function of time and valued in some *state space* S (usually a real vector space).

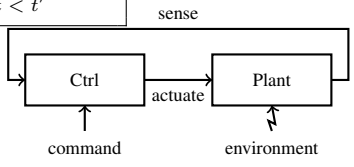

Fig. 3: Generic Hybrid System Representation

As we will need it in later proofs or properties, we also model time with a single read-only variable, t, which is "simulated" by the `Progress` event.

The behaviour of hybrid systems is then modelled through 4 types of events.

b) Transition Events: Model internal decisions of the controller. This typically corresponds to discrete changes happening in the program or decisions made by the user.

```
Transition
ANY s
WHERE
  grd1 : s ∈ ℙ1(STATES)
THEN
  act1 : x_s :∈ s
END
```

```
Sense
ANY s, p
WHERE
  grd1 : s ∈ ℙ1(STATES)
  grd2 : p ∈ ℙ(STATES×ℝ × S)
  grd3 : (x_s ↦ t ↦ x_p(t)) ∈ p
THEN
  act1 : x_s :∈ s
END
```

c) Sensing Events: Represent changes in the controller induced by changes in the plant, typically detected through sensors. This event is similar to `Transition` except its guards generally involve x_p.

d) Behave Events: Symbolize spurious changes in the plant, or *perturbations* in other words.

```
Behave
ANY e
WHERE
  grd1 : e ∈ DE(S)
  grd2 : Solvable([t, +∞[, eq)
THEN
  act1 : x_p :|
    x'_p ∈ ℝ+ → S∧
    [0, t[ ◁ x'_p = [0, t[ ◁ x_p∧
    solutionOf([t, +∞[, [t, +∞[ ◁ x'_p, e)
END
```

This particular event modifies the continuous state (x_p) which is a function of time. To ensure coherence, we need to state that the *past* of the system remains the same; that is, the function does not change on $[0, t\mathbb{R}[$.

The *future* of the system ($[t, +\infty\mathbb{R}[$) is then set to be a solution of the given equation.

```
Actuate
ANY e, s
WHERE
  grd1 : e ∈ DE(S)
  grd2 : Solvable([t, +∞[, eq)
  grd3 : s ⊆ STATES
  grd4 : x_s ∈ s
THEN
  act1 : x_p :| x'_p ∈ ℝ+ → S∧
    [0, t[ ◁ x'_p = [0, t[ ◁ x_p∧
    solutionOf([t, +∞[, [t, +∞[ ◁ x'_p, e)
END
```

e) Actuation Events: Models the action of the controller on the plant, achieved thanks to *actuators*.

This event is shaped on `Behave` from which it differs by the presence of additional guards to relate it to the controller's state.

C. Deriving specific controlled systems

The generic model described in the previous section is used as the entry point of a method for designing hybrid systems. Indeed, hybrid models should instantiate this generic model through refinement, using the following steps:

1) Describe the set of discrete states, `STATES`. This usually corresponds to the places or modes of the hybrid automaton.

2) Describe the continuous state space, S. Generally, it is of the form $\mathbb{R}^n$ with $n \in \mathbb{N}$.
3) Define the plant's variables as functions of time.
4) Describe every event of the system as a refinement of one of the generic events.
5) Provide witnesses for disappearing event parameters. In particular, this is where the differential equations describing the system are given.

IV. Hybrid Railway Model in Event-B

This section describes the formal signalling model developed by instantiating the previously described hybrid model. We begin by formalizing requirements for the signalling model presented in Section 2. Further we present the instantiated events of the hybrid system modelling framework.

A. Modelling continuous system features

The model focuses on one train of acceleration, speed and position (resp.) ta, tv and tp. We control the engine's power, which *in fine* yields an input acceleration denoted f.

The trains obey a first order non-linear differential equation called the *Davis Equation* given in Equation 1. This equation is parameterized by constants a, b and c which are given and fixed (and generally depends on the rails, train mass, etc.).

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

The train is provided with an *end of movement authority* (EoA) which evolves in the time.

Additionally, the system is able to sense its distance to EoA, and in particular determine if, given current speed and acceleration, it can stop before it. As long as it can, the train is said to be in *free mode*, and it can chose arbitrary values for f. Whenever it cannot anymore, it goes into *restricted mode* and is then required to provide values for f so that it can stop before EoA.

The *stopping distance* computation is generally done by a complex algorithm on the on-board computer. It is abstracted by a $StopDist$ function which takes the current acceleration $ta(t)$ and speed $tv(t)$ as parameters.

B. Signalling model requirements

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: **at all times the train must remain within the issued movement authority.** Additionally one needs to prove properties about the plant. For example, one must prove that speed be always positive or traction force will remain within the minimum and maximum interval.

SAF_1	$\forall t \cdot t \in \mathbb{R}^+ \Rightarrow tp(t) \leq EoA$
PLT_1	$\forall t \cdot t \in \mathbb{R}^+ \Rightarrow tv(t) \geq 0$
PLT_2	$f \geq f_{min} \wedge f \leq f_{max}$

C. Hybrid signalling model in Event-B

Context and Theories. As the behaviour of train is generally common to several models, we decided to create a reusable *Trains* theory. This theory defines the Davis equation with initial condition $tv(t_0) = tv_0$, $tp(t_0) = tp_0$ (**DavisEquation**(a,b,c, f,t_0,tv_0,tp_0)). In the theory file we also define other related theorems on the mathematical properties of this equation that will be useful for completing proofs.

```
CONTEXT Signalling
EXTENDS ControlledSystemCtx
CONSTANTS
  free_move, restricted_move
  StopDist
  a,b,c
  f_min, f_max, f_dec_min
AXIOMS
  partition(STATES,{free_move
      },{restricted_move})
  StopDist ∈ (ℝ × ℝ) ⇸ ℝ
  StopDist(0 ↦ 0) = 0
  ...
END
```

Listing 1: Signalling Event-B Model

In addition we created a context in the model (see excerpt in Listing 1) which defines several constants of the system as well as constraints on them. In particular, we axiomatise the **StopDist** function, the Davis equation parameters and boundary for the engine's traction power: minimum power (f_{min}), maximum power (f_{max}) and minimum deceleration power (f_{dec_min}), i.e. the minimum strength with which the train can decelerate. And finally we declare train controller modes (*free_move* and *restricted_move*) with an enumerated set.

Machine Events. In the first refinement of the generic hybrid model we introduce several new events which instantiate generic events presented in the Section 4.2. However due to space limitations 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 continuously updated without specifying how at this level of abstraction.

Transition_restricted_move event models the change in the speed controller by adjusting trains traction effort when train is in the restricted move mode. The event is simply guarded by a single predicate which enables event if and only if the status variable x_s is set to *restricted_move*. To control train's speed we created a variable f which denotes the traction force and is modified by the action such that the stopping distance would not overshoot the end of the movement authority. One must then prove an open proof obligation that such traction force value can be found.

```
Transition_restricted_move
REFINES Transition
WHERE
  grd1: x_s = restricted_move
WITH
  s: s = {restricted_move}
THEN
  act1: f :| tp(t) + StopDist(f' ↦ tv(t)) ≤ EoA
END
```

Listing 2: Event restricting train's movement (transition type)

Another internal controller event which changes controllers mode based on the input from the plant is sense event -

Sense_to_restricted. The one out of two sense events will change the train state variable x_s if the end of movement authority has not been extended and train must decelerate in order to remain within issued movement authority.

```
Sense_to_restricted
REFINES Sense
WHERE
  grd1: tp(t) + StopDist(ta(t) ↦ tv(t))) ≥ EoA
WITH
  s: s = {restricted_move}
  p: p = STATES×ℝ × {v* ↦ p* |
     p* + StopDist(f_dec_min ↦ v*) ≥ EoA}
THEN
     act1: x_s ≔ restricted_move
END
```

Listing 3: Event train controller's state (sense type)

As previously described the other set of events define the evolution of the plant. The two events provided below model the environment and the plant influenced by the controller. The first event modifies plant variables based on *some* solvable differential equation which describes the environment.

The dynamic protocol parts, such as messages exchanges, are modelled as variables and events computing next variable states and contained in a machine.

```
Behave REFINES Behave
ANY e
WHERE
  grd1: e ∈ DE(S)
  grd2: Solvable([t, +∞], e)
WITH
  x'_p : x'_p = bind(tv', tp')
THEN
  act1: tv, tp :| tv' ∈ ℝ+ → S ∧ tp' ∈ ℝ+ → S∧
    [0, t[ ◁ tv' = [0, t[ ◁ tv ∧ [0, t[ ◁ tp' = [0, t[ ◁ tp∧
    solutionOf([t, +∞[, [t, +∞[ ◁ bind(tv', tp'), e)
END
```

Listing 4: Event updating train plant (behave type)

Similarly the *Actuate_move* event updates plant variables tp, tv, tp. However, in this instance, variable are updated based on the Davis equation with traction force variable f value as one of the parameters.

V. Discussion

One of the main objectives of this research was to further demonstrate the general applicability of the proposed method [7] for modelling hybrid systems. 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 artifact, which could be further refined to capture a specific signalling configurations (e.g. by defining a specific railway schema) or modelling railway protocols (e.g. signalling handover).

As discussed by Alur in [5], 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

```
Actuate_move
REFINES Actuate
WHERE
  grd1: ⊤
WITH
  x'_p: x'_p = bind(tv', tp')
  e: e = DavisEquation(a, b, c, f, t, tv(t), tp(t))
  s: s = STATES
THEN
  act1: tv, tp :| tv' ∈ ℝ+ → S ∧ tp' ∈ ℝ+ → S∧
    [0, t[ ◁ tv' = [0, t[ ◁ tv ∧ [0, t[ ◁ tp' = [0, t[ ◁ tp∧
    solutionOf(
      [t, +∞[,
      [t, +∞[ ◁ bind(tv', tp'),
      DavisEquation(a, b, c, f, t, tv(t), tp(t))
    )
END
```

Listing 5: Event updating train's plant (actuate type)

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 II) and also similar works [7], [26], [27], the proof automation of hybrid Event-B models is still low and must be improved for a more practical applications. In spite of improved verification tooling, a refinement plan for hybrid models should be reconsidered, perhaps, a top-down approach is one suitable (particularly, for this model), where continuous system's aspects should be introduced as late as possible in the model.

Model	|POs|	Auto.	Inter.
hybrid_model (4m. + 1c.)	61	23	38
communication_model (1m. + 8c.)	49	18	31
Total	110	41	69

TABLE II: Event-B protocol model proof statistics

An important feature of this hybrid system development approach is the requirement of explicitly stating system's dynamic properties. In our opinion, this problem is often overlooked and could lead to mis-communication between, for instance, control engineers and software engineers. With our proposed approach, a formal hybrid artifact is created, which can be used between different types of engineers. On the other hand, 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 [28].

VI. Conclusions and Future Work

In this paper we addressed the challenge of modelling and verification of hybrid systems. This work builds on the recent development of the Rodin Theory plug-in, which allows users to extend the Event-B specification language and the Event-B modelling framework for hybrid systems. The main technical objective of this research was to demonstrate that the modelling framework for hybrid systems could be used for modelling systems with more complex dynamics.

In the paper, we demonstrated how a generic hybrid model could be refined with train specific dynamics. The work included developing a new, reusable continuous theory, for the train plant, which is defined by the Davis equation as well as other associated Event-B context files for aspects such as stopping distance function. As stated before the main invariants we need to prove is 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 [29]. To achieved 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 [28].

Acknowledgements. The work is partially supported by an iCASE studentship (EPSRC/UK and Siemens Rail Automation) and the EPSRC/UK STRATA platform grant.

REFERENCES

[1] ERTMS User Group, "UNISIG: ERTMS/ETCS system requirements specification. version 2.2.2 (2002)."

[2] IEEE Std 1474.1-2004, "IEEE standard for Communications-Based Train Control (CBTC) performance and functional requirements," 2005.

[3] F. Badeau and A. Amelot, "Using b as a high level programming language in an industrial project: Roissy val," in ZB 2005: Formal Specification and Development in Z and B, H. Treharne, S. King, M. Henson, and S. Schneider, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2005, pp. 334–354.

[4] P. Behm, P. Benoit, A. Faivre, and J.-M. Meynadier, "Météor: A successful application of b in a large project," in FM'99 — Formal Methods, J. M. Wing, J. Woodcock, and J. Davies, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 369–387.

[5] R. Alur, "Formal verification of hybrid systems," in Proceedings of the Ninth ACM International Conference on Embedded Software, ser. EMSOFT '11. New York, NY, USA: ACM, 2011, pp. 273–278.

[6] W. Su, J.-R. Abrial, and H. Zhu, "Formalizing hybrid systems with Event-B and the rodin platform," Science of Computer Programming, vol. 94, no. Part 2, pp. 164 – 202, 2014, abstract State Machines, Alloy, B, VDM, and Z.

[7] G. Dupont, Y. Aït-Ameur, M. Pantel, and N. K. Singh, "Proof-based approach to hybrid systems development: Dynamic logic and event-b," in Abstract State Machines, Alloy, B, TLA, VDM, and Z, M. Butler, A. Raschke, T. S. Hoang, and K. Reichl, Eds. Cham: Springer International Publishing, 2018, pp. 155–170.

[8] ——, "Hybrid systems and Event-B: A formal approach to signalised left-turn assist," in New Trends in Model and Data Engineering. Springer International Publishing, 2018, pp. 153–158.

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

[10] T. S. Hoang, M. Butler, and K. Reichl, "The hybrid ertms/etcs level 3 case study," in Abstract State Machines, Alloy, B, TLA, VDM, and Z, M. Butler, A. Raschke, T. S. Hoang, and K. Reichl, Eds. Cham: Springer International Publishing, 2018, pp. 251–261.

[11] M. Butler, "A system-based approach to the formal development of embedded controllers for a railway," Design Automation for Embedded Systems, vol. 6, no. 4, pp. 355–366, Jul. 2002.

[12] T. Kiss and K. T. Jánosi-Rancz, "Developing railway interlocking systems with session types and event-b," in Proceedings of the IEEE 11th International Symposium on Applied Computational Intelligence and Informatics (SACI), May 2016, pp. 93–98.

[13] M. Butler and I. Maamria, Practical Theory Extension in Event-B. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 67–81.

[14] U. Berger, P. James, A. Lawrence, M. Roggenbach, and M. Seisenberger, "Verification of the european rail traffic management system in real-time maude," Science of Computer Programming, vol. 154, pp. 61 – 88, 2018, formal Techniques for Safety-Critical Systems 2015.

[15] J. Hoenicke and E. Olderog, "CSP-OZ-DC: A combination of specification techniques for processes, data and time," Nord. J. Comput., vol. 9, no. 4, pp. 301–334, 2002.

[16] A. Cimatti, M. Roveri, and S. Tonetta, "Requirements validation for hybrid systems," in Proceedings of the 21st International Conference on Computer Aided Verification, ser. CAV '09. Springer-Verlag, 2009, pp. 188–203.

[17] A. Halchin, A. Feliachi, N. K. Singh, Y. A. Ameur, and J. Ordioni, "B-perfect - applying the PERF approach to B based system developments," in Reliability, Safety, and Security of Railway Systems. Modelling, Analysis, Verification, and Certification - Second International Conference, RSSRail 2017, Pistoia, Italy, November 14-16, 2017, Proceedings, 2017, pp. 160–172.

[18] A. Halchin, Y. A. Ameur, N. K. Singh, A. Feliachi, and J. Ordioni, "Certified Embedding of B Models in an Integrated Verification Framework," in 2019 13th International Symposium on Theoretical Aspects of Software Engineering (TASE), Jul. 2019.

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

[20] L. Zou, J. Lv, S. Wang, N. Zhan, T. Tang, L. Yuan, and Y. Liu, "Verifying chinese train control system under a combined scenario by theorem proving," in Verified Software: Theories, Tools, Experiments, E. Cohen and A. Rybalchenko, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2014, pp. 262–280.

[21] R. Banach, M. Butler, S. Qin, N. Verma, and H. Zhu, "Core hybrid event-b i: Single hybrid event-b machines," Science of Computer Programming, vol. 105, pp. 92 – 123, 2015.

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

[23] R.-J. Back, "Refinement calculus, part ii: Parallel and reactive programs," in Proceedings on Stepwise Refinement of Distributed Systems: Models, Formalisms, Correctness, ser. REX workshop. New York, NY, USA: Springer-Verlag New York, Inc., 1990, pp. 67–93.

[24] J.-R. Abrial, M. Butler, S. Hallerstede, T. S. Hoang, F. Mehta, and L. Voisin, "Rodin: An open toolset for modelling and reasoning in event-b," Int. J. Softw. Tools Technol. Transf., vol. 12, no. 6, pp. 447–466, Nov. 2010.

[25] M. Butler, J.-R. Abrial, and R. Banach, From Action Systems to Distributed Systems: The Refinement Approach, ser. Computer and Information Science Series. Chapman and Hall/CRC, Apr. 2016, ch. Modelling and Refining Hybrid Systems in Event-B and Rodin, pp. 29–42.

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

[27] C. Bogdiukiewicz, M. Butler, T. S. Hoang, M. Paxton, J. Snook, X. Waldron, and T. Wilkinson, "Formal development of policing functions for intelligent systems," in 2017 IEEE 28th International Symposium on Software Reliability Engineering (ISSRE), Oct 2017, pp. 194–204.

[28] T. Blochwitz, M. Otter, M. Arnold, C. Bausch, C. Clau, H. Elmqvist, A. Junghanns, J. Mauss, M. Monteiro, T. Neidhold, D. Neumerkel, H. Olsson, J. v. Peetz, S. Wolf, A. S. Gmbh, Q. Berlin, F. Scai, and S. Augustin, "The functional mockup interface for tool independent exchange of simulation models," in Proceedings of the 8th International Modelica Conference, 2011.

[29] A. Iliasov, P. Stankaitis, D. Adjepon-Yamoah, and A. Romanovsky, "Rodin platform why3 plug-in," in Abstract State Machines, Alloy, B, TLA, VDM, and Z, M. Butler, K.-D. Schewe, A. Mashkoor, and M. Biro, Eds. Cham: Springer International Publishing, 2016, pp. 275–281.