Modelling, Visualisation and Proof of an ETCS Level 3 Moving Block System

Michael Leuschel[1(✉)] and Nader Nayeri[2]

[1] Institut für Informatik, Universität Düsseldorf, Universitätsstr. 1, 40225 Düsseldorf, Germany
michael.leuschel@hhu.de

[2] Ground Transportation Deutschland GmbH, Stuttgart, Germany
nader.nayeri@urbanandmainlines.com

Abstract. This work aims to formally ensure the safety of modern moving block systems. For this a proof model was developed in Event-B which captures several safety critical aspects. The new model identifies several key concepts, that are at the heart of the mathematical safety proof and which should later be at the heart of the safety case for a moving block system with trackside train detection. Some of the key concepts were inspired by earlier CBTC models and adapted for ETCS moving block, and a few novel key concepts were developed to deal safely with delays of train position reports and trackside train detection.

The invariants of the proof model have proven mathematically with the Rodin toolset, thereby establishing safety properties of the modelled system. The proof model can also be animated and visualised using the ProB validation tool. By necessity, the proof model abstracts away from irrelevant details and still has some restrictions in scope (such as linear topology). Nonetheless, even with current restrictions, the key concepts already proved valuable when reasoning about safety of moving block systems. In the article we also present our modelling and tooling methodology, outlining the importance of complementing proof with animation. We also explain the importance of inductive properties and argue that a train-centric approach is more promising for proof of a moving block system than a track-centric approach.

1 Introduction

B was originally developed by Jean-Raymond Abrial in the 1990s and has now been used for over 25 years in the railway sector [9] (see also [19,36]). Currently the B Method is used in three distinct ways for safety critical systems:

– B for software (classical B [2]): here an abstract specification of a component is successively refined until one reaches an implementation level (called B0), where automatic code generators can be applied. Examples of this are the *Paris Métro Line 14* [16], the New York Canarsie Line [17] and around 95 installations of Alstom's U400 CBTC (Communication-Based Train Control) system, which contain code generated from verified classical B models.

B. Milius et al. (Eds.): RSSRail 2023, LNCS 14198, pp. 193–210, 2023.
https://doi.org/10.1007/978-3-031-43366-5_12

– B for system modelling (Event-B [3]): here B is used to model an entire system, not just an individual component. The goal is to verify critical properties at the system level and understand why a system behaves correctly.
– B for data validation: here properties about data and system configurations are specified in B. These properties can then be checked automatically (and reliably) on concrete data, resulting in validation reports.

For this work, we focus on the second application, using Event-B for safety cases for railway applications. Several railway examples can be found using Event-B for system modelling, notably the academic interlocking model in Chapter 17 of Abrial's book [3] on Event-B, and a much more detailed model [8] including feature management. On the industrial side, ClearSy has used Event-B in at least two previous projects [13,35] (Flushing Line for New York City Transit Authority and Octys for the Parisian Autonomous Transport Administration RATP) to perform a safety analysis of CBTC metro systems. Safety analyses of Alstom's U400 Zone Controller have also been carried out [14]. ClearSy is using this approach also for mainline systems, namely the new Hybrid ERTMS Level 3 signalling on the Marseille to Ventimiglia line.[1]

Within Thalesthe approach was previously used for Hybrid Level 3 [1] fixed virtual block systems [22]. There a formal model was developed that helped uncover over 30 errors in the original EUG specification and to various real-life demonstration by running the model in real-time using ProB [30]. The formal model was used as a demonstrator and presented at the 2018 Innotrans trade fair.However, the purpose of the model was not to conduct a formal proof; the errors were uncovered using animation and model checking. Other models for Hybrid Level 3 with Event-B were developed [15,20,33], within the scope of the ABZ 2018 case study [25]. While [33] was able to prove several properties using Event-B, some important aspects were not proven. Indeed, we believe that the Hybrid Level 3 [1] specification is not well suited for a formal correctness proof in general, nor for an inductive correctness proof within the B-method in particular (we return to this in Sect. 3 below). In this article, we focus on modelling a European Train Control System (ETCS) level 3 *full moving block* system with Trackside Train Detection (TTD), as specified in [38]. The modelling is done in Event-B, inspired by the previous successful applications of Event-B to CBTC systems in [13,14,35]. Motivated by the successful use of animation and visualisation in [22], we also strive to make the formal models amenable to animation. One goal of this work is to answer these key questions:

– Can we develop Event-B models for both proof and animation and is it beneficial to do so? Which methodology should be used to develop such models?
– Can the CBTC knowledge and insights be transferred to a moving block ETCS system?

[1] https://www.clearsy.com/en/ertms-en/.

2 Modelling Methodology

Before explaining specifics of the formal proof model, we elaborate on a few general principles we have employed in this and other projects.

2.1 Verification Techniques: Proof Is Essential

Verification of formal models can be done in three ways: 1) formal proof (e.g., [3]), 2) explicit model checking [26], or 3) symbolic model checking [5,11].

While model checking can scale for verifying concrete instances of interlocking systems, (see, e.g., [6,7,23]), it does not yet scale for moving block systems. One reason is that train positions need to be modelled much more precisely, leading to a considerably larger state space that has to be verified. Setting aside scalability, ideally one wants to establish a safety case independent of the topology. This can only be done with proof. As such we see **proof** to be essential for establishing the safety of moving block systems in particular and safety critical systems in general.

2.2 Complementing Proof with Model Checking

Proof is rarely fully automatic and it is easy to make mistakes either in the proof itself or in the formulation of the properties in the model. Even experienced formal methods practitioners take time to analyse a failed proof, trying to decide whether the model needs changing or whether it is just the proof that needs to be done differently.

Hence **model checking** is a good complement to automatically check a system on small instances and find issues quickly. Indeed, if the model checker finds an error we know the model needs to be changed and we also obtain a counterexample that can be used to diagnose the problem.

2.3 Combining Proof and Animation

When doing mathematical proof it is essential that the axioms from which proof starts are consistent. Otherwise anything can be proven and a proof is worthless.

In classical B for software, an inconsistency in the axioms (aka PROPERTIES clause of B) would eventually be detected when implementing a system or checking the axioms during data validation [31]. While this is quite late (and can thus be expensive), at least the issue would be uncovered. In system modelling, however, there is not necessarily a refinement chain from the high-level specification to some concrete implementation. As such, inconsistencies in the axiomatisation may not be detected at all!

Hence, particularly for systems modelling, **animation** is essential for its role in ensuring that the axioms are consistent. Indeed, an animator like ProB will check the axioms in the initial setup phase, before performing the first animation steps. In our experience, this check has detected numerous inconsistencies in the axioms both in academic and industrial models.

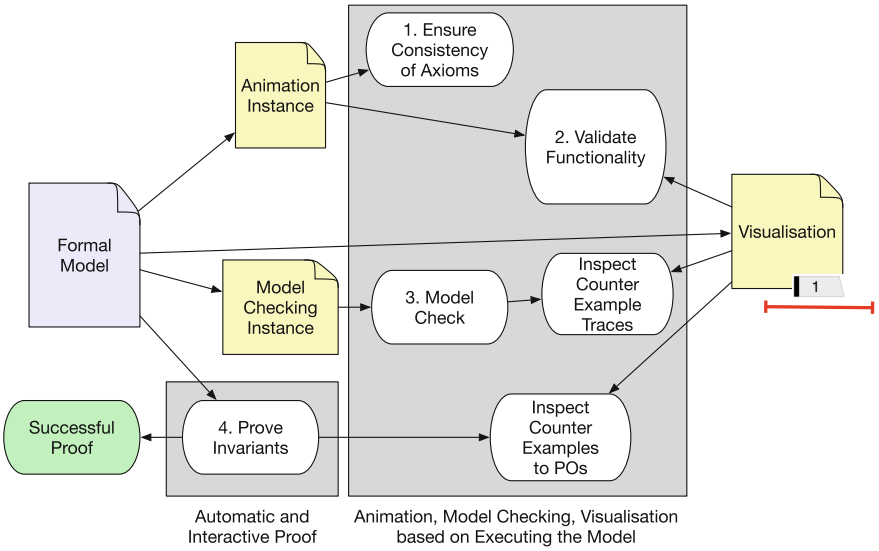

Fig. 1. Tooling Methodology

Animation is also a good complement to model checking, which may give false confidence (one should be skeptical when the model checker reports that your model is correct too quickly) and will only look for the very specific errors you have thought of. In fact, an animator will allow a user to interactively explore the behaviour of a model (i.e., controlling the non-determinism in the user interface). A such one can check that the model dynamics meet expectations, and one often uncovers issues that one did not think of before (and hasn't expressed formally yet as invariants or temporal logic formulas).

Adapting a quote from Leslie Lamport ("Formal mathematics is nature's way of letting you know how sloppy your mathematics is" [28, page 2]) we state that:

> Animation is nature's way of letting you know how sloppy your formal mathematics is.

2.4 Challenges in Combining Proof, Animation and Model Checking

There are also considerable challenges when combining proof and animation. Formulas that help conducting proof (e.g., universal quantifications over large or infinite domains) can make animation difficult or impossible. Still, thus far we have always managed to overcome these issues. Complicated formulas can be moved to the theorems (aka ASSERTIONS) section, which are not checked during constant setup. We have also added a feature to mark some properties with a "prob-ignore" pragma. The ProB animator will skip such properties, but it will mark them as "not fully checked". One should thus still use the tool

to validate such properties for a finite subset of parameters. In our experience, there are only a few of those "difficult" properties.

There used to be a considerable debate in the formal methods community whether specifications should be executable [21] or not [24]. While [24] is certainly right to point out the tension between proving and execution, some of the examples and arguments in [24] relate to imperative or functional languages, not to the constraint-logic programming foundation of tools such as ProB. E.g., ProB executes the "non-executable" perfect square example from [24] with ease, finding all perfect squares in 1..10000 in about 0.002 s. While the article [24] makes the plea that "the positive advantages of specification should not be sacrificed to the separable objective of prototyping" our plea is that "the positive advantages of animation should not be sacrificed to the separable objective of proving". One could argue that sometimes it is impossible to write a natural model that is amenable to proof and animation. For example, sometimes a proof model may use an infinite or very large domain (e.g., for positions of the train). This means that the model per se may not be animatable, but one can often reduce the size for animation by providing an instance or refinement of the model. This is what we did for our model, by providing several concrete track topologies for animation (see Fig. 1). Thus far we have never encountered a situation that was not "fixable".

3 Principles of the Moving Block Model

Scope. The aim of this model is to establish the safety principles of a full moving block train control system for ETCS. We are thus not aiming for hybrid level 3 in the middle of Fig. 2, where track sections are divided into virtual subsections of fixed size and location. We are aiming to analyse a full moving block system, where there is no fixed granularity of zones that can be allocated to trains. We still want to cater for the possible existence of trackside train detection devices (**TTD**s), which can detect the presence of trains on certain fixed sections of track, see [38, page 10]. We, however, assume that our trackside system called TrackSide, encompasses both train protection and route protection functions as in traditional signalling systems. It should be pointed out that no routes are modelled.

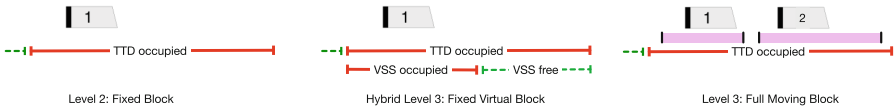

Fig. 2. ETCS Level 2, Hybrid Level 3 and Level 3

Finding Key Properties. Conducting a formal proof of a system often relies on finding and formulating certain **key properties** of the system.

These key properties must be strong enough to establish safety of the system, but weak enough so that they can be proven. Such proofs tend to be done by induction (see, e.g., Sect. 3.2 of [13]):

- we have to establish that the property holds initially
- we have to prove that when the property holds for some time point, that it holds after any possible next event.

We call a property that satisfies these two points **inductive**. Note that typical safety properties themselves do **not** satisfy the latter requirement and are thus not inductive: i.e., the fact that two trains are not in collision at time t does in no way guarantee that they will not collide immediately thereafter.

HL3 VSS Status Is Not Inductive. Finding the key properties is thus difficult and often requires many iterations of modelling and proof. Maybe surprisingly, the key concepts in the hybrid level 3 [1] specification are not inductive. [1] has a "track-centric" approach and relies on four different kinds of status (free, occupied, unknown, ambiguous) for the various virtual subsections (VSSs) of the track. For example, [1] contains these two possible track statuses:

- "unknown" when there is no certainty about whether the VSS is "occupied" or not.
- "ambiguous" when the VSS is known to be occupied by a (connected) train, but when it is unsure whether another (not connected) train is also present on the same VSS.

However, the knowledge about the status of each subsection of the track is **not** sufficient to always correctly determine the status in the next cycle of the system. For example, given the knowledge that a VSS is "ambiguous" and the knowledge that a connected train leaves the VSS, we do not know whether the VSS stays "ambiguous" (if another connected train is on the section) or switches to "unknown". The track status knowledge is not amenable to an inductive correctness proof; the VSS status information is too weak to allow to preserve it upon updates.

The hybrid level 3 specification [1] solves these issues by additional timers, the value of those timers contains additional knowledge and help determining the correct updates of the status of a VSS. Dealing with such timers is very error-prone, and it is difficult to provide a completely formal description of the meaning of a status value of a VSS in terms of possible train positions. We believe this explains the number of issues found by formal modelling [22].

3.1 Key Concepts Explained

Thus, in our model we do not use track-centric key concepts based on track status, but a train-centric approach. In other words, our model (and TrackSide) keeps track of a variety of **protection zones** which are associated with individual trains and not with track sections.

The overall safety of the system follows from the fact that the TrackSide keeps safe train images covering at all time points the real train position of all trains.

The last point is very important: an essential safety concept is the train image, and not the status of track sections. As such the modelling is quite different than, e.g., the one performed in the hybrid level 3 principles paper [1].

Key Concept 1: trainPZ At the heart of the modelling is the train protection zone (trainPZ) concept. Every train, registered or not, has an associated trainPZ. However, this is a virtual proof concept as the trackside does not directly know the trainPZ of the trains. The TrackSide only knows of the existence of registered trains and does not know how many unregistered trains there are.

The trainPZ always encompasses the associated train's physical location on the track. This is captured in these safety invariants in Table 1. However, the trainPZ covers more than just the current location of the train: it covers *all the controlled or uncontrolled movements* that the train can make in the future, without any further communication between train and TrackSide. In other words, the TrackSide cannot prevent the train from moving backwards or forwards within its trainPZ. These movements cover the following practical events:

- shortening or lengthening of trains due to acceleration, deceleration or slopes,
- rollback and roll forward,
- controlled moving forward to the movement authority (MA).

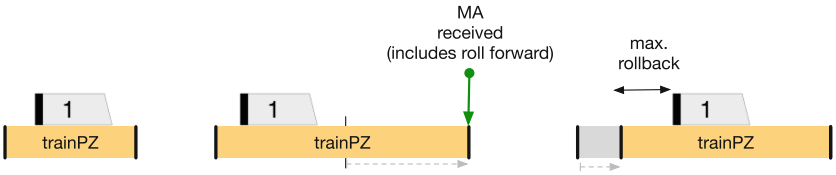

Fig. 3. Movement of a registered train within its trainPZ

On the one hand, the proof model assumes that a train will never leave its trainPZ. On the other hand, the proof model allows a train to freely move within its trainPZ. The proof model makes some assumptions about these possible movements of a train within its trainPZ. More precisely, we assume that rollback and roll forward are bounded by a maximal value. These assumptions are important, e.g., to adequately treat trackside train detection (ttd) devices, as we explain below. Indeed, these assumptions are also used in practice by the TrackSide to infer upper and lower bounds for the trainPZ (even though the TrackSide does not know the precise trainPZ of trains). Indeed,

Table 1. Some invariants related to TRAINPZ

`@safetyPZ`
$\forall t1, t2.t1 \in activeTrains \wedge t2 \in activeTrains \wedge t1 \neq t2 \Rightarrow trainPZ(t1) \cap trainPZ(t2) = \emptyset$
All TRAINPZs are pairwise disjoint.
`@trainPZ_safe_rear`
$\forall tr.tr \in activeTrains \Rightarrow trainPZ\_rear(tr) \leq train\_rear(tr)$
`@trainPZ_safe_front`
$\forall tr.tr \in activeTrains \Rightarrow train\_front(tr) \leq trainPZ\_front(tr)$
All trains are always fully contained in their TRAINPZs.

– knowing that the *train location* lies within some interval is *not an inductive property* (i.e., without further knowledge about the train's speed, acceleration, movement authority, etc., we have no way of knowing where the train will be located next),
– while knowing that the TRAINPZ lies within some interval *is an inductive property*, and the extensions of the TRAINPZ can be monitored by the TRACKSIDE.

The proof model also distinguishes between unregistered and registered (connected) trains. For registered trains the TRAINPZ can change as follows:

– the TRAINPZ of a registered train can be **extended** at the front, this corresponds to receiving an MA. This is illustrated in the middle of Fig. 3.
– the TRAINPZ of a registered train is **reduced** at the back when a train moves forward. Note that the proof model contains a constant RollbackDistance for the "maximum rollback" and roll forward. The model enforces that the train's back location stays within that distance of the back of the TRAINPZ. This is illustrated on the right in Fig. 3.

Figure 4 illustrates the interplay of the TRAINPZ and the maximum rollback when TRACKSIDE processes TTD freeness. In Fig. 4 the TTD1 is marked as free (e.g., detected by axle counters or a track circuit). This, however, does **not** mean that the entire area of TTD1 is guaranteed to remain free, as the TRAINPZ of train 1 could still reach back into the area of TTD1; TRACKSIDE thus cannot prevent train 1 from rolling back onto TTD1 within its TRAINPZ. Howevever, TRACKSIDE **does** know that there is a maximum rollback and hence does know that there is a maximum reach of the TRAINPZ back onto TTD1. In other words, the gray area on the left in Fig. 4 is guaranteed not to contain the TRAINPZ of train 1.

Key Concept 2: REGPZ The next important concept is the TRACKSIDE protection zone REGPZ for every registered train. This is a zone managed and known

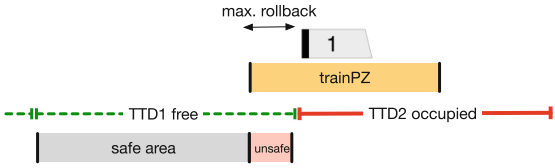

Fig. 4. TTD freeness in relation to trainPZ and maximum rollback

by the TrackSide. This zone completely covers the trainPZ of the associated train, and *not just* its current position. This is very important: as Fig. 4 shows, just requiring that a regPZ covers the train's current position would *not* be an inductive property necessary for a successful proof.

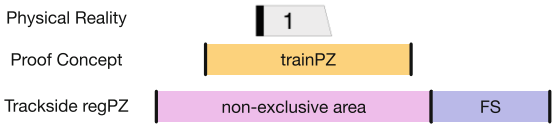

Fig. 5. The regPZ for registered trains in relation to the trainPZ

As Fig. 5 shows the regPZ is split into two parts:

- the non-exclusive area (nea), where the train is allowed to move at its own "risk", this area is not guaranteed to be free of other trains. This concept covers OS (On-Sight) and SR (Staff Responsible) modes. This area is necessary to cater for splitting or joining trains and also directly after registering of trains, when it is unclear whether other unregistered trains can be close to the train (e.g., on the same TTD).
- the FS (Full Supervision) area where the train can move forward without risking encountering any other train or obstacle.

A regPZ is always non-empty and must always fully cover the trainPZ of the associated registered train. The FS part, however, can be empty. The nea part contains at least one "unit" of measurement; this also enables joining two registered trains (the train behind can obtain an OS ma for the last "unit of measurement" to connect the trains).

An regPZ can be reduced at the back, based, e.g., either on TTD freeness or train position reports. The model requires to prove that such reduction still safely covers the train's trainPZ. A regPZ can extended at the front, either in an nea or a FS fashion.

There are also events that allow to extend the trainPZ of registered trains within the regPZ. These events correspond to the train receiving an ma from the TrackSide.

Fig. 6. The extension of a trainPZ due to an MA issued within the regPZ. On the left the TrackSide issues an MA, on the right the train actually receives the MA.

Fig. 7. The reduction of a regPZ due to a train position report or a free TTD report

Key Concept 3: Zones for Unregistered Trains. The next important concept are the non-identified protection zone niPZ. These zones are managed by TrackSide and are meant to cater for unregistered trains or other obstacles. Such zones are similar to the zones described in Fig. 1 of [13].

The TrackSide, by the invariants, knows that any part of the track not covered by a niPZ is free of unregistered trains. Together with the regPZ zones, the TrackSide can thus safely detect parts of the track are free, and that unless it issues MAs, these zones will stay free.

3.2 Major Challenge: Proving Preservation of Invariants in the Presence of Delays

We first concentrate on the reduction of zones at the back by the TrackSide when receiving new information. This information can either come from TTD's or from trains. The reduction at the back corresponds to the train having moved forward and we can free an associated zone (niPZ or regPZ) at the back. In our approach the actual duration of the delay is actually irrelevant, as long as the information can still be safely applied:

- if the information is still applicable, we simply reduce the zone at the back for niPZ and regPZ.
- if the information is no longer applicable it is discarded.

Below we show, how we decide whether information is still applicable and how we have formally proven that the zone reductions are safe.

Figure 8 shows two scenarios. At the start (on the left) TTD1 has become free in the physical reality but the message has not yet arrived at TrackSide. In scenario 1 the train rolls back and occupies TTD1 again and the free message is now processed by TrackSide: TrackSide reduces the regPZ, taking the roll-back into account (see Fig. 4). In scenario 2 the train is moving forward towards its MA (end of trainPZ). Even if the freeness information arrives very late,

the action taken by the TrackSide is identical: the same safe reduction of the regPZ occurs as in scenario 1. Also note that if the TTD freeness information gets lost, the TrackSide information remains safe.

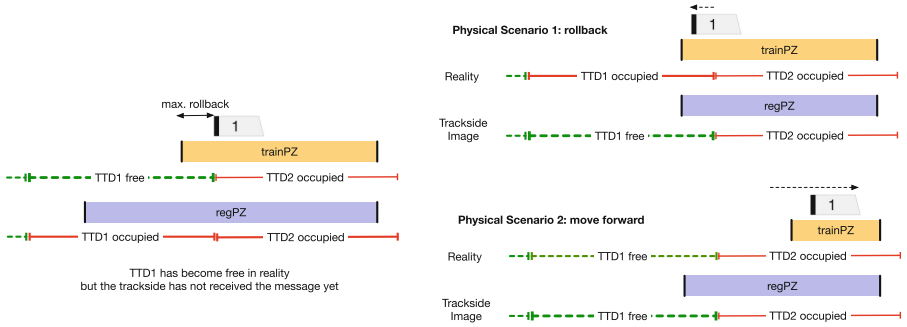

Fig. 8. The safe reduction of regPZ due to a free TTD report with two different delays

To conduct the successful proof in the model it is crucial that between a) the TTD freeness occurring and c) the arrival of the message the affected protection zone (regPZ or niPZ) was *not modified*. Figure 9 shows one scenario where a regPZ has been modified between the point that the physical freeness occurred and the TTD freeness message is processed. This leads to an erroneous reduction at the bottom of the figure, where the train is no longer covered by its regPZ.

The proof model uses this approach to dealing with delays:

- when a physical event like TTD freeness or sending of a TPR (train position report) occurs, the model executes a "virtual" event which marks existing zones (regPZ or niPZ) as safe for reduction
- the invariants of the model ensure that the potential reduction remains safe, as long as the message is in transit and the affected zone has **not been modified**,
- when the TTD freeness or TPR report arrives, the TrackSide can safely reduce the affected zones which have not been modified. This event contains in its guard a check that the zone is safe. Later implementations (aka refinements) of this event will have to prove that this guard holds. The guard is thus a documentation of the condition that ensures safety of any implementation. There are various ways this guard can be satisfied in practice, e.g., by remembering how long ago a zone was changed and by having time stamps for the TTD free and TPR events.

Thus in essence, a safe implementation of a TrackSide will need to ensure that a TTD freeness or TPR event occurred physically **after** the last time an affected zone was changed.

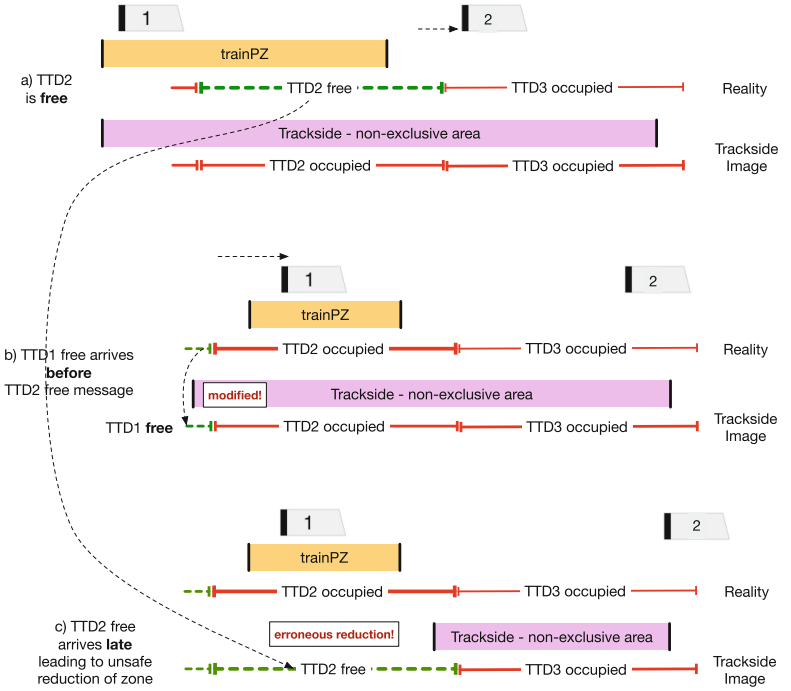

Fig. 9. Erroneous reduction of REGPZ due to delay in TTD2 arrival

3.3 Visualisation

Also, it is usually worthwhile to add **visualisation** to animation. While this adds some initial effort, it is quickly recovered by much more quickly spotting mistakes. For example, in the picture on the right of Fig. 2 one can immediately spot that train 2 is preceding train 1 and both are located on the same track section. This situation would be far from obvious when looking at a textual representation of the model's state (e.g., $trainFront = \{tr1 \mapsto 25, tr2 \mapsto 44\}, trainRear = \{tr1 \mapsto 22, tr2 \mapsto 39\}$, ..., $ttdLeft = \{ttd1 \mapsto 0, ttd2 \mapsto 20, ttd3 \mapsto 48\}, ttdRight = \{ttd1 \mapsto 19, ttd2 \mapsto 47, ttd3 \mapsto 69\}$).

Visualisation also makes the models amenable to validation by domain experts, unfamiliar with the formal notation used. This was the case for the HL3 models [22] that could be animated and inspected by domain experts.

In our case the visualisations were done with the VisB [37] plugin of PROB in a generic way. For examples trains and zones (protection zones, movement authorities, ...) were drawn using SVG (scalable vector graphics) polygons with coordinates derived from the formal model.

3.4 Tooling Decisions: Proof and Platform

For Event-B one can either use the commercial Atelier-B tool or the Rodin toolset [4]. The former has a textual syntax and more powerful proof interface, e.g., allowing the addition of new proof rules. On the other hand, the Atelier-B proving interface is more difficult to master. Rodin provides a more intuitive proof interface, where many actions can be carried out by clicking on symbols in the hypotheses or goals. On the downside, the modelling language of Rodin is more restricted as far as statements and components are concerned.

We have conducted initial modelling efforts in the more flexible Atelier-B/ProB syntax, and once the components and concepts had stabilised, we translated the model to a linear refinement chain in Rodin (see also [32]). Note that we could build on the experience of a preceding project, where we had developed a first moving block model.

The proof effort was reasonable. We constructed 10 iterations of the Rodin model over the course of 14 months. Iteration 5 for example had 271 proof obligations, 197 proven automatically, 71 manually. Three proof obligations related to initialisation were not proven but checked by ProB by starting the animation. Iteration 8 had 380 proof obligations, iteration 10 had 423. The last iteration has 7 refinement levels (i.e., one abstract machine and 6 refinements), as well as one additional refinement containing instantiations for animation, see Fig. 1.

The provers were sometimes a bit weak for expressions using intervals in combination with minimum and maximum values. Here we used ProB's Disprover [27] to uncover required hypotheses for proof to go through. We actually developed a new improved exportof proof obligations from Rodin so that the counterexamples can be inspected and visualised in detail (see bottom of Fig. 1). One sees the counterexample which invalidates the proof goal, satisfies all selected hypotheses but violates at least one global hypothesis.

4 Scope, Uses, Future and More Related Work

The proof model aims to formally analyse and ensure the safety of moving block systems. The model was developed in Event-B and its scope covers these aspects:

- Movement of multiple trains on a linear track section, including rollback and forward (Sect. 3.1).
- Joining and splitting of trains.
- Train registering and deregistering with TrackSide.
- TrackSide issueing movement authorities to registered trains
- TrackSide maintaining a safe image of the track's occupation using:
  - Identified zones for registered trains (Sect. 3.1),
  - Non-identified zones for unregistered trains and obstacles,
  - Safe processing of updates from trackside train detection (TTDs) and train position reports (TPRs), safely dealing with delays.
  - Invariants to guarantee that the image is a safe over-approximation of the real occupation at any time.

Thus far our Event-B model has been used internally for

- uncovering and visualisation of tricky danger scenarios and communication with domain experts in ETCS railway systems.
- The derived key properties provide guidelines for later implementation of moving block systems,
- the guards of the TrackSide events in the model contain the necessary operational conditions to ensure safety.

4.1 Possible Future Extensions

Below is a summary of possible ways to extend and improve our proof model.

- *Integrity loss* can be considered a special case of splitting a train. A corresponding event (to provoke integrity loss) is present in the current model, but is disabled. Correct processing of integrity loss will require adapting some invariants and adding train integrity status information to TPR messages.
- To support *non-linear topologies* new events and invariants will need to be added at higher refinement level. To be generic, we should also model that trains can appear at points and that trains can disappear at points (moving to and from other parts of the topology). This is already prepared in the latest version of our model by its support for joining and splitting trains, but will require updating invariants and proofs for niPZ zones.
- Allowing *bi-directional movement* is feasible and one would have to add a direction for every train and a direction for every zone.

In the future it would also be interesting to check whether we can produce a provably correct variation of Hybrid Level 3 [1] virtual fixed blocks by a form of data refinement, i.e. projecting our protection zones on the virtual sub sections.

4.2 Changes Compared to Previous Models

The proof model was inspired by an earlier CBTC proof model [35] developed by ClearSy for Thales Toronto, as well as a moving block model developed by the authors. Some of the key concepts were taken from [35], but there are significant changes:

- There is a *single* model for all proof aspects (not a union of various sub-models as in [35]),
- The model can be animated and visualised, avoiding inconsistencies or errors in the safety proofs,
- The TrackSide has no separate interlocking component,
- There are no complex physical movement functions for trains (the new proof model uses a simpler modeling approach, allowing trains to move freely in their trainPZ, see Sect. 3.1). We decided to try a simpler model of the train movement, to ease proof and maintenance of the model. The approach is more general and turned out to be sufficient for establishing safety. (It is, however, always possible to add more detailed movement descriptions as B refinements later.)

- There is a non-exclusive (NEA) area where safety invariants are relaxed to allow joining and splitting of trains (joining and splitting is probably not catered for in the CBTC models).
- The new proof model has, however, a more principled way of dealing with delays.

5 Conclusion

The essential goal of the proof model is to ensure that the moving block TrackSide computes a safe internal image of the external world. It is secondary what this safe image is used for, be it collision freeness, derailment or other safety hazards.

- The formal model contains a gluing invariant, which formulates very precisely the meaning of the image inside of TrackSide. It provides the exact relationship between the internal information and the physical position of trains and the physical status of the infrastructure.
- The model is very precise about the time points for which the internal information is valid. Delays between the image and the physical world are not "swept under the carpet". The formal model provides conditions under which the safety of the image is maintained by any update and action.

The proof model uses a new refinement technique to deal with arbitrary delays, by adding virtual events and temporary invariants that are preserved until the end of the delay. In summary, we have achieved the following:

- We have ported some of the ideas and concepts from CBTC models to a (simplified) generic moving block model.
- We were able to create a formal model suited for rigorous formal proof and animation and visualization.
- We successfully combined sub-models into a single overall model, that can be used as demonstrator or shown to domain experts.
- We established the absence of collisions (despite delays in position reports).
- Parts of the model were formally proven for all topologies. Train image and protection zones are vital concepts and scale to inductive formal proof.

The take away conclusions of this work are:

- We are confident that a proven formal model for moving block can be developed with sufficient precision for ETCS.
- Reusing experience from CBTC (and also HL3) is possible, key concepts and components are now known. We argue for a train-centric approach with inductive key properties rather than the track-centric approach of [1].
- The key to a good design and good safety case are inductive concepts, which have a clear and precise link to the physical reality, are robust wrt delays and which can be understood by stakeholders and (future) engineers alike.

– It is important to complement formal proof with animation and it is possible, given current tooling, to develop a formal Event-B model that can serve multiple purposes: as safety proof, as an executable reference specification understandable by domain experts, and as a demonstrator for experimentation and field tests.

Acknowledgement. We thank anonymous referees for their useful feedback.

References

1. Hybrid ERTMS/ETCS Level 3. Principles Ref: 16E042, Version: 1A, EEIG ERTMS Users Group, 123–133 Rue Froissart, 1040 Brussels, Belgium (2017)
2. Abrial, J.-R.: The B-Book. Cambridge University Press, Cambridge (1996)
3. Abrial, J.-R.: Modeling in Event-B: System and Software Engineering. Cambridge University Press, Cambridge (2010)
4. Abrial, J.-R., Butler, M., Hallerstede, S., Voisin, L.: An open extensible tool environment for event-B. In: Liu, Z., He, J. (eds.) ICFEM 2006. LNCS, vol. 4260, pp. 588–605. Springer, Heidelberg (2006). https://doi.org/10.1007/11901433_32
5. Biere, A., Kröning, D.: SAT-based model checking. In: Handbook of Model Checking, pp. 277–303. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-10575-8_10
6. Borälv, A.: Case study: formal verification of a computerized railway interlocking. Formal Aspects Comput. **10**(4), 338–360 (1998)
7. Breton, N., Fonteneau, Y.: S3: proving the safety of critical systems. In: Lecomte, T., Pinger, R., Romanovsky, A. (eds.) RSSRail 2016. LNCS, vol. 9707, pp. 231–242. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33951-1_17
8. Butler, M., et al.: Formal modelling techniques for efficient development of railway control products. In: Fantechi, A., Lecomte, T., Romanovsky, A. (eds.) RSSRail 2017. Lecture Notes in Computer Science, vol. 10598, pp. 71–86. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68499-4_5
9. Butler, M., et al.: The first twenty-five years of industrial use of the B-method. In: ter Beek, M.H., Ničković, D. (eds.) FMICS 2020. LNCS, vol. 12327, pp. 189–209. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58298-2_8
10. Su, W., Chen, J., Khan, S.: Insulin pump: modular modeling of hybrid systems using event-B. In: Butler, M., Raschke, A., Hoang, T.S., Reichl, K. (eds.) ABZ 2018. LNCS, vol. 10817, pp. 403–408. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91271-4_31
11. Chaki, S., Gurfinkel, A.: BDD-based symbolic model checking. In: Handbook of Model Checking, pp. 219–245. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-10575-8_8
12. Marques-Silva, J., Malik, S.: Propositional SAT solving. In: Handbook of Model Checking, pp. 247–275. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-10575-8_9
13. Comptier, M., Deharbe, D., Perez, J.M., Mussat, L., Pierre, T., Sabatier, D.: Safety analysis of a CBTC system: a rigorous approach with event-B. In: Fantechi, A., Lecomte, T., Romanovsky, A. (eds.) RSSRail 2017. Lecture Notes in Computer Science, vol. 10598, pp. 148–159. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68499-4_10

14. Comptier, M., Leuschel, M., Mejia, L.-F., Perez, J.M., Mutz, M.: Property-based modelling and validation of a CBTC zone controller in event-B. In: Collart-Dutilleul, S., Lecomte, T., Romanovsky, A. (eds.) RSSRail 2019. LNCS, vol. 11495, pp. 202–212. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-18744-6_13
15. Dghaym, D., Dalvandi, M., Poppleton, M., Snook, C.F.: Formalising the hybrid ERTMS level 3 specification in iUML-B and event-B. Int. J. Softw. Tools Technol. Transf. **22**(3), 297–313 (2020)
16. Dollé, D., Essamé, D., Falampin, J.: B dans le transport ferroviaire. L'expérience de Siemens transportation systems. Technique et Science Informatiques, **22**(1), 11–32 (2003)
17. Essamé, D., Dollé, D.: B in large-scale projects: the Canarsie line CBTC experience. In: Julliand, J., Kouchnarenko, O. (eds.) B 2007. LNCS, vol. 4355, pp. 252–254. Springer, Heidelberg (2006). https://doi.org/10.1007/11955757_21
18. Fantechi, A., Lecomte, T., Romanovsky, A.B.: (eds.) Proceedings RSSRail 2017, LNCS 10598. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68499-4
19. Ferrari, A., et al.: Survey on formal methods and tools in railways: the ASTRail approach. In: Collart-Dutilleul, S., Lecomte, T., Romanovsky, A. (eds.) RSSRail 2019. LNCS, vol. 11495, pp. 226–241. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-18744-6_15
20. Fotso, S.J.T., Frappier, M., Laleau, R., Mammar, A.: Modeling the hybrid ERTMS/ETCS level 3 standard using a formal requirements engineering approach. Int. J. Softw. Tools Technol. Transf. **22**(3), 349–363 (2020)
21. Fuchs, N.E.: Specifications are (preferably) executable. Softw. Eng. J. **7**(5), 323–334 (1992)
22. Hansen, D., et al.: Using a formal B model at runtime in a demonstration of the ETCS hybrid level 3 concept with real trains. In: Butler, M., Raschke, A., Hoang, T.S., Reichl, K. (eds.) ABZ 2018. LNCS, vol. 10817, pp. 292–306. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91271-4_20
23. Haxthausen, A.E., Nguyen, H.N., Roggenbach, M.: Comparing formal verification approaches of interlocking systems. In: Lecomte, T., Pinger, R., Romanovsky, A. (eds.) RSSRail 2016. LNCS, vol. 9707, pp. 160–177. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33951-1_12
24. Hayes, I., Jones, C.B.: Specifications are not (necessarily) executable. Softw. Eng. J. **4**(6), 330–338 (1989)
25. Hoang, T.S., Butler, M., Reichl, K.: The hybrid ERTMS/ETCS level 3 case study. In: Butler, M., Raschke, A., Hoang, T.S., Reichl, K. (eds.) ABZ 2018. LNCS, vol. 10817, pp. 251–261. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91271-4_17
26. Holzmann, G.J.: Explicit-state model checking. In: Handbook of Model Checking, pp. 153–171. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-10575-8_5
27. Krings, S., Bendisposto, J., Leuschel, M.: From failure to proof: the ProB disprover for B and event-B. In: Calinescu, R., Rumpe, B. (eds.) SEFM 2015. LNCS, vol. 9276, pp. 199–214. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-22969-0_15
28. Lamport, L.: Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers. Addison-Wesley, Boston (2002)
29. Limbrée, C., Cappart, Q., Pecheur, C., Tonetta, S.: Verification of railway interlocking - compositional approach with OCRA. In: Lecomte, T., Pinger, R., Romanovsky, A. (eds.) RSSRail 2016. LNCS, vol. 9707, pp. 134–149. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33951-1_10

30. Leuschel, M., Butler, M.J.: ProB: an automated analysis toolset for the B method. STTT **10**(2), 185–203 (2008)
31. Leuschel, M., Falampin, J., Fritz, F., Plagge, D.: Automated property verification for large scale B models with ProB. Formal Asp. Comput. **23**(6), 683–709 (2011)
32. Leuschel, M., Mutz, M., Werth, M.: Modelling and validating an automotive system in classical B and event-B. In: Raschke, A., Méry, D., Houdek, F. (eds.) ABZ 2020. LNCS, vol. 12071, pp. 335–350. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-48077-6_27
33. Mammar, A., Frappier, M., Fotso, S.J.T., Laleau, R.: A formal refinement-based analysis of the hybrid ERTMS/ETCS level 3 standard. Int. J. Softw. Tools Technol. Transf. **22**(3), 333–347 (2020)
34. Mazzanti, F., Basile, D.: A formal methods demonstrator for railways. ERCIM News **121**, 2020 (2020)
35. Sabatier, D.: Using formal proof and B method at system level for industrial projects. In: Lecomte, T., Pinger, R., Romanovsky, A. (eds.) RSSRail 2016. LNCS, vol. 9707, pp. 20–31. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-33951-1_2
36. ter Beek, M.H., et al.: Adopting formal methods in an industrial setting: the railways case. In: ter Beek, M.H., McIver, A., Oliveira, J.N. (eds.) FM 2019. LNCS, vol. 11800, pp. 762–772. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-30942-8_46
37. Werth, M., Leuschel, M.: VisB: a lightweight tool to visualize formal models with SVG graphics. In: Raschke, A., Méry, D., Houdek, F. (eds.) ABZ 2020. LNCS, vol. 12071, pp. 260–265. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-48077-6_21
38. X2Rail-3. Advanced signalling, automation and communication system (IP2 and IP5). Deliverable D4.2: Moving block specifications. Part 2 - System definition. Technical report (2020). https://projects.shift2rail.org