Merging Railway Standard Notations in a Formal DSL-Based Framework

Asfand Yar[1], Akram Idani[1(✉)], and Simon Collart-Dutilleul[2,3]

[1] Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000 Grenoble, France
asfand.yar@grenoble-inp.org, akram.idani@univ-grenoble-alpes.fr
[2] Institut de Recherche Technologique Railenium, 59300 Famars, France
[3] Univ. Lille Nord de France, IFSTTAR, 59666 Villeneuve d'Ascq Cedex, France
simon.collart-dutilleul@ifsttar.fr

Abstract. The design of a railway signalling system may be validated using three basic concepts: (1) functional standards, (2) domain specific notations, and (3) safety requirements checking. However, there is a lack of tools that merge these notions in a unified framework to be used by standardisation authorities, as well as domain experts and safety engineers. In this ongoing work we make the bridge between the three notions using Meeduse, a tool in which the B method is applied in order to formally reason on the correctness of domain specific languages (DSLs) and simulate their dynamic semantics using the ProB animator. The application context of this work is that of two well known standards in the railway field: RailTopoModel and ERTMS/ETCS. We propose a railway DSL framework whose static semantics are built on top of RailTopoModel and the underlying dynamic semantics conform to ERTMS/ETCS. The overall approach is assisted by the B method, which allows us to define, prove and animate safety-critical behaviors given domain-centric models.

Keywords: ERTMS/ETCS · RailTopoModel · B method · DSL

1 Introduction

In the railway field, there are several tools that propose Domain Specific Languages (DSLs) to model railroad networks such as RaIL-AiD[1] and SafeCap [7]. They allow the design of readable models thanks to domain specific notations. However, most of their DSLs are not formally defined and hence they do not apply formal verification techniques such as theorem proving or model-checking to guarantee the correctness of the underlying semantics. Furthermore, often the DSLs they provide are not directly derived from existing standards, such as RailTopoModel [8] and ERTMS/ETCS [3]. In order to circumvent these shortcomings, we are developing a formally proved railway DSL framework whose

[1] Railway Infrastructure and Layout Aided Designer (https://www.rail-aid.com).

H. Muccini et al. (Eds.): ECSA 2020, CCIS 1269, pp. 411–419, 2020.
https://doi.org/10.1007/978-3-030-59155-7_30

static semantics implement RailTopoModel and dynamic semantics comply with ERTMS/ETCS operating rules.

RailTopoModel is the International Railway Standard (IRS 30100[2]) developed by the UIC (International Union of Railways), with the contribution of several railway infrastructure managers and industrial companies, for the sake of optimizing communication between the various actors of the railway sector. It defines and describes the structure of a railway network together with the physical installations that it manages. These structural business assets are intended to be as complete as possible, however the model does not provide operating rules such as route computations and train movements. Our work addresses these behaviours by focusing on the European signalling and train control system ERTMS/ETCS in order to introduce standardized management rules and their underlying safety properties within RailTopoModel. Our approach is assisted by the B method which allows to define, prove and animate safety-critical behaviours given domain specific models designed in our DSL framework.

Section 2 outlines the main principles of this ongoing work. In Sect. 3 discusses how the formal B method will be integrated within our DSL-based framework. Finally, Sect. 4 draws the conclusions and the perspectives of this work.

2 Proposed Approach

2.1 Overall Architecture

Figure 1 gives an overall view about our approach for merging RailTopoModel and ERTMS/ETCS in a formal DSL-based framework. Our framework is composed of the two layers presented at the top and at the bottom of Fig. 1.

The semantics layer covers both static and dynamic semantics of our DSLs: the static semantics are built on meta-models that we derive from RailTopoModel, and the dynamic semantics are built on ERTMS/ETCS specifications. Regarding the execution layer, it is managed by tool Meeduse[3] that animates behaviours of domain specific models conforming to the semantics layer. Formal B specifications are used in both layers in order to apply formal reasoning techniques to our DSLs: proofs for the semantics definition, and animation/model-checking for models execution. The choice of the B method is motivated by several aspects. First, the B method is widely used in the railway field and there are several success stories that support this fact [9], such as for example Meteor, the automated Paris subway. Recently, a comparative study of several formal methods regarding their industrial suitability [10] has been done and rated B high when it comes to formal constructs like those applied in our work.

[2] The IRS 30100 is the foundation for quick, unambiguous and error-free data storage and data exchange inside and between business processes [8].

[3] http://vasco.imag.fr/tools/meeduse.

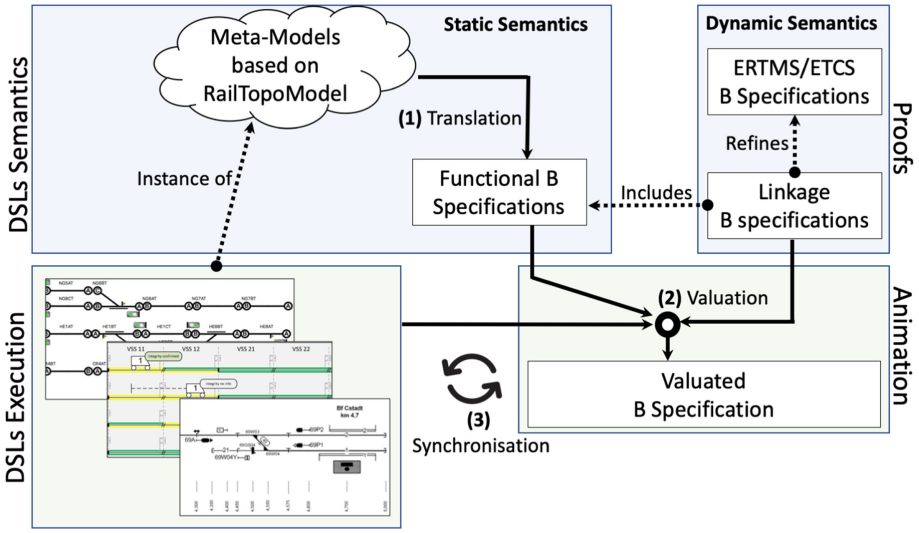

Fig. 1. Architecture of the proposed framework

2.2 Methodology

RailTopoModel is presented in [8] based on a UML class diagram divided into four packages. The Base package of RailTopoModel defines a railway network by an abstraction level (meso, micro, macro) and a composition of railway resources. For example, a network resource can be a NetElement such as line sections, or an InterlockingNetEntity such as signals. Having this reference UML model of RailTopoModel, we introduce two additional meta-models with specific concepts, each of them led to a particular DSL: (1) the Topology DSL allows the domain expert to represent lines, tracks and their connections; and (2) the Infrastructure DSL allows to add objects over a given topology such as physical objects (*e.g.* stations), immaterial objects (*e.g.* speed limits) and logical objects (*e.g.* signals). In order to ensure the conformance of our DSLs with RailTopoModel, our methodology follows the following established rules:

- The Core package contains the exact RailTopoModel: as the semantics of RailTopoModel are defined using a UML class diagram, this step simply introduces the underlying UML concepts within the Eclipse Modeling Framework (EMF), as an EMF meta-model.
- Define the additional meta-models (Topology and Infrastructure) outside the core package and use references. Our aim is to guarantee that the initial RailTopoModel semantics are kept unchanged during the DSL development. The core meta-model remains then low-coupled with the additional meta-models. This rule provides two main advantages: (1) there is no need to modify or extend the core meta-model and therefore it can be considered as

an independent artifact, and (2) the additional meta-models could be easily extended or replaced without any impact on the core meta-model.
- Classes of our meta-models (such as those of ERTMS/ETCS) must inherit from classes issued from the Core meta-model. This inheritance allows one to associate clearly identified semantics from RailTopoModel to any additional class.
- Associations between the additional classes must be computed as much as possible from the elements of RailTopoModel. This rule allows to reduce the number of relations as much as possible and carefully check whether there exists a way to compute these relations from relations of the core meta-model.

2.3 The Core Meta-Model

Figure 2 shows a subset of the Core meta-model. This meta-model applies generic concepts used in railway networks. Class Network for example is composed of network resources (class NetworkResource) that represent its topological and structural properties such as the various net elements and their locations.

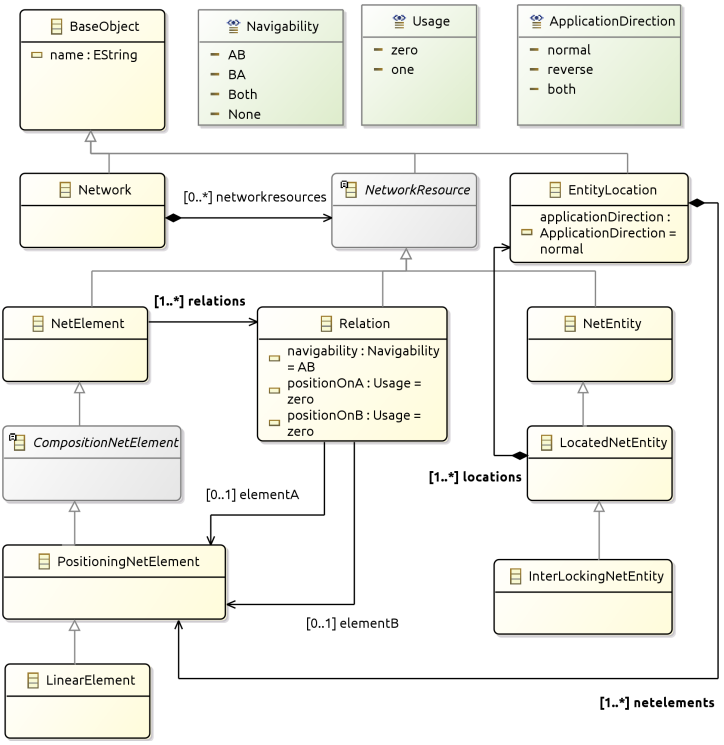

Fig. 2. Subset of the core Meta-model

2.4 Defining the Additional Meta-Models

Figure 3 illustrates the Topology meta-model where the upper part contains the root class of this meta-model called Topology. It consists of LinearElements and InterlockingNetEntities. The bottom part of meta-model shows the class Track (inherited from LinearElement) and the classes Switch and BufferStop (inherited from InterLockingNetEntity). BufferStop can be used as a start or end of any network track while Switch is the intermediate junction among three tracks. Each switch has an attribute called continueCourse which sets the track to be used (right track or left track). Note that classes LinearElement and InterlockingNetEntity are defined in the core package. On the one hand they are referenced by the root class and on the other hand they are specialized by the additional classes Track, Switch and BufferStop. Indeed, these three classes are not initially defined by RailTopoModel but they are required by a railway DSL especially to define the dynamic semantics.

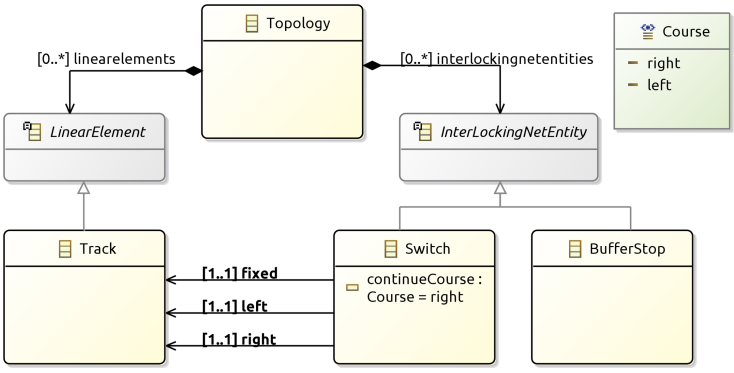

Fig. 3. Topology Meta-Model

Regarding the Infrastructure meta-model, we apply the same principles. This meta-model contains infrastructure elements to make railway network operational. For this purpose, we introduce concepts from ERTMS/ETCS such as: movement authority, train, virtual block and track-side.

2.5 Modeling

Our DSL tool allows to instantiate the aforementioned meta-models using domain specific notations. Figure 4 is an example of a topology designed based on the Topology meta-model. It represents buffer stops (bus01, bus02, bus03), switches (sw01, sw02, sw03 etc) and tracks (trc01, trc02, trc03 etc).

As presented by the topology meta-model, each switch has three branches: the fixed branch, the left branch and the right branch. The fixed branch is a fixed course for the switch which is not change-able while the continue course is

change-able and can be set to left or right which directs the train either to the left branch or the right branch. In the left hand-side of Fig. 5, the green arrow shows the course assigned to the branches. The arrow to right branch is green as continue course of switch is set to right. The continue course of the same switch shown in the right side of Fig. 5 is set to left which turns the color of arrow to left branch into green and arrow to right branch into red.

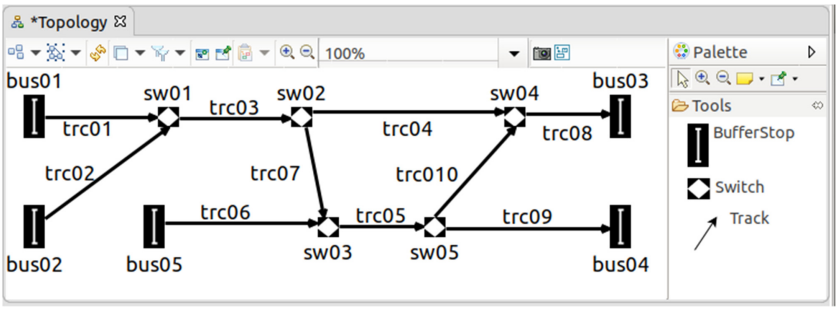

Fig. 4. Designed topology

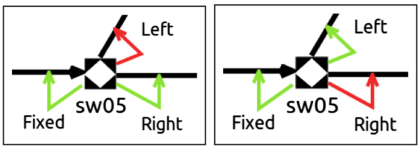

Fig. 5. Switch branches (Color figure online)

3 Formal Semantics

The advantage of a MDE architecture is that it allows to easily develop DSL tools with graphical or textual concrete syntax. This approach puts into practice a clear separation of concerns ranging from requirements to target platforms, and going through several design stages. This is useful especially for railway model editors, because the interoperability between these tools is favored by the use of standardized meta-modeling formalisms. A DSL allows to reduce the risk that human errors such as misinterpretation of the requirements and specification documents lead to erroneously validate the specifications, and produce a wrong real system. Still, while MDE provides solutions to the validation problem, the verification problem remains a major challenge. In this ongoing work we formally define the semantics of our DSL tool using the B method and apply the underlying reasoning tools such as the AtlierB prover and the ProB model-checker. Note that most of the static semantics of our railway DSL tool and the associated graphical concrete syntax are available, and currently we are actively working on the definition of the formal semantics.

3.1 Static Semantics

The formal definition of static semantics is ensured by our tool Meeduse [6]. It applies a classical UML-to-B translation [5] (step (Translation) in Fig. 1) to meta-models and produces a functional B specification covering data structures as well as basic operations (getters, setters, etc). The structure of the resulting B specifications is presented in Fig. 6 where the B machine of the Infrastructure DSL requires data (*sees* dependency) defined in the Topology DSL. The Infrastructure machine is further refined in order to redefine abstract infrastructure objects by means of ERTMS data objects (such as eurobalises, virtual blocks) that are not initially provided by RailTopoModel but which are part of the static semantics. The refinement is then dedicated to guarantee by proofs that this redefinition preserves the infrastructure DSL invariants.

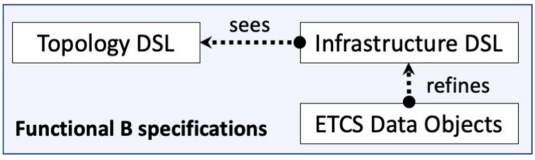

Fig. 6. Formal static semantics

3.2 Dynamic Semantics

The dynamic semantics of a DSL deal with behavioural descriptions that make the DSL executable. In our work, we apply ERTMS/ETCS as a way to introduce execution within RailTopoModel. Indeed, ERTMS/ETCS defines safe train behaviours thanks to the mechanism of movement authority. It describes how and when permissions to enter block sections are assigned to trains. Note that in the last decade, several works have been devoted in order to provide formal models of ERTMS/ETCS. Recently the ABZ'2018 conference [4] has published several B models, which provides us a rich catalog of proved B operations and invariants.

Our objective is to reuse these existing B specifications for the dynamic semantics definition. For this purpose, we create linkage B specifications (Fig. 1) in which we apply two mechanisms from the B method: refinement and inclusion. In the B method, refinements have two main principles: add requirements by going from abstract models to more concrete ones and prove the preservation of the abstract model invariants. The composition, such as inclusion, allows to beak down the system by applying the separation of concerns principle.

In our approach, refinements would guarantee the preservation of the safety invariants of ERTMS/ETCS defined in the re-used B specifications and inclusion provides an access to the B variables that represent the static semantics and use them in place of those of the refined machines. Proved B operations are

then refined by DSL-centric operations and hence behaviours that comply with ERTMS/ETCS are applied to our DSLs.

3.3 Execution

DSL execution is intended to perform early validation since the DSL is expected to behave as the target system should run. In our framework, this execution is done by Meeduse given domain-specific models that represent railway topologies and infrastructures conforming to RailTopoModel. First, Meeduse injects these models into the functional B specifications issued from our meta-models. This step, called (Valuation) in Fig. 1, creates enumerations and generates substitutions that assign concrete initial values to the B variables. Then, the tool asks ProB to compute the initial state of the B specifications and the list of operations that can be animated from this state. At this stage, railway experts can start playing with the B operations of the linkage machines in order to simulate ERTMS/ETCS train behaviours. All along the interactive animation, Meeduse synchronises the current state of the B specifications with the input models (step (Synchronisation) in Fig. 1), which results in a domain-centric visual animation. The interest of this approach in comparison with classical visual animation is that our framework allows railway experts to design by themselves the input models and validate their behaviours without being trained in formal methods.

4 Conclusion

The use of Domain-specific modelling languages becomes important in the railway domain as they provide support for railway mechanisms from their semantics definition to their concrete syntax. In the last decade, several tools and platforms [1,2,7,11,12] were proposed in order to allow railway experts to design railway infrastructures and associated signalling systems. However, the limitation with these tools is that either the semantics of their DSLs don't fully comply with international railway standards.

In the railway domain, several specifications are defined by European and national authorities like the ISO[4] specifications from AFNOR[5] or TSIs[6] from EUAR (European Union Agency for Railway). These specifications provide standardised engineering rules and infrastructure guidelines and allow the establishment of common interfaces for railway systems in order to maintain the compatibility among cross-border infrastructure objects. They also provide cost-effectiveness processes to ensure safety by using the best practices.

This paper presents the main principles of our ongoing work for the development and the execution of railway DSLs that comply with current standards: RailTopoModel and ERTMS/ETCS. We apply the B method to formally define the static and dynamic semantics and prove that functional specifications can be

[4] https://www.iso.org/.
[5] Association Française de Normalisation.
[6] Technical Specifications for Interoperability.

executed on a given ETCS-based infrastructure, without braking global safety invariants. Our approach is domain-centric, which allows domain experts to design topological views of a railway system and then play with scenarios that comply with ERTMS/ETCS. This work provides two main contributions in comparison with existing railway editors: our DSL is derived from approved railway standard documents (RailTopoModel and ERTMS/ETCS) and the underlying semantics follow a formal method with available reasoning tools.

References

1. Industrial Railway CAD software. https://hwww.railcomplete.com/
2. Railway Infrastructure and Layout Aided Designer. https://www.rail-aid.com/
3. The ERTMS/ETCS signalling system. http://www.railwaysignalling.eu/wp-content/uploads/2016/09/ERTMS_ETCS_signalling_system_revF.pdf
4. Butler, M., Raschke, A., Hoang, T.S., Reichl, K. (eds.): ABZ 2018. LNCS, vol. 10817. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-91271-4
5. Idani, A., Ledru, Y.: B for modeling secure information systems. In: Butler, M., Conchon, S., Zaïdi, F. (eds.) ICFEM 2015. LNCS, vol. 9407, pp. 312–318. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-25423-4_20
6. Idani, A., Ledru, Y., Vega, G.: Alliance of model-driven engineering with a proof-based formal approach. Innovations Syst. Softw. Eng. 1–19 (2020). https://doi.org/10.1007/s11334-020-00366-3
7. Iliasov, A., Lopatkin, I., Romanovsky, A.: The SafeCap platform for modelling railway safety and capacity. In: Bitsch, F., Guiochet, J., Kaâniche, M. (eds.) SAFECOMP 2013. LNCS, vol. 8153, pp. 130–137. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-40793-2_12
8. International Union of Railways (UIC): RailTopoModel - Railway infrastructure topological model (2016). ISBN 978-2-7461-2513-1
9. Lecomte, T.: Applying a formal method in industry: a 15-year trajectory. In: Alpuente, M., Cook, B., Joubert, C. (eds.) FMICS 2009. LNCS, vol. 5825, pp. 26–34. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-04570-7_3
10. Mashkoor, A., Kossak, F., Egyed, A.: Evaluating the suitability of state-based formal methods for industrial deployment. Softw. Pract. Experience **48**(12), 2350–2379 (2018)
11. Vu, L., Haxthausen, A., Peleska, J.: A domain-specific language for railway interlocking systems. In: Proceedings of the 10th Symposium on Formal Methods for Automation and Safety in Railway and Automotive Systems, pp. 200–209. Technische Universität Braunschweig (2014)
12. Vu, L.H., Haxthausen, A.E., Peleska, J.: A domain-specific language for generic interlocking models and their properties. In: Fantechi, A., Lecomte, T., Romanovsky, A. (eds.) RSSRail 2017. LNCS, vol. 10598. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-68499-4_7