B Formal Validation of ERTMS/ETCS Railway Operating Rules

Rahma Ben Ayed[1], Simon Collart-Dutilleul[1], Philippe Bon[1], Akram Idani[2], and Yves Ledru[2]

[1] IFSTTAR-Lille, 20 Rue Elisée Reclus BP 70317, 59666 Villeneuve d'Ascq Cedex, France
`{rahma.ben-ayed,simon.collart-dutilleul,philippe.bon}@ifsttar.fr`
`http://www.ifsttar.fr`

[2] UJF-Grenoble 1/Grenoble-INP/UPMF-Grenoble 2/CNRS, LIG UMR 5217, 38041 Grenoble, France
`{akram.idani,yves.ledru}@imag.fr`
`http://www.liglab.fr`

Abstract. The *B* method is a formal specification method and a means of formal verification and validation of safety-critical systems such as railway systems. In this short paper, we use the B4MSecure tool to transform the UML models, fulfilling requirements of European Railway Traffic Management System (ERTMS) operating rules, into *B* specifications in order to formally validate them.

Keywords: Railway operating rules, UML models, Role Based Access Control, *B* method, formal validation.

1 Introduction

ERTMS [6] is the European Railway Traffic Management System which is designed to replace the national on-board railway systems in Europe in order to make rail transport safer and more competitive, and to improve cross-border connections. ERTMS includes the European Train Control System (ETCS) which specifies the on-board equipment and its communication with the trackside.

The aim of our work is to confront the European specifications with the national operating rules, as well as the use of formal models to validate whether a given scenario fulfills the specification regarding the functional and safety requirements. We propose to model a nominal scenario of *Movement Authority (MA)*, extracted from ERTMS operating rules, and to translate it into *B* specifications in order to validate it.

In the following section, an overview of the nominal scenario *MA* is given and its UML models are described. Section 3 highlights the *B* formal validation after an automatic translation of these models into *B* specifications using B4MSecure. Finally, section 4 concludes this paper.

2 *Movement Authority* Overview

Movement Authority (*MA*) is an authorization given to a train to move to a given point as a supervised movement. Some features can be used to define an *MA*, such as sections subdividing it, the time-out value attached to each section, etc. The *MA* function unfolds with interactions between the *OnboardSafetyManagement* (the on-board computer-based machine), the *TracksideSafetyManagement* (the trackside computer-based machine), and the *Driver*, as follows:

MA.1 The *OnboardSafetyManagement* requests an *MA* to the *TracksideSystem.*
MA.2 The *TracksideSafetyManagement* receives the *MA* request from the *TracksideSystem.*
MA.3 The *TracksideSafetyManagement* proposes an *MA* to the *TracksideSystem* after creating it. It can also modify and/or delete the *MA*.
MA.4 The *OnboardSafetyManagement* receives the proposed *MA* from the *TracksideSystem*, authorizes it and processes the *MA* authorization in order to be displayed in the *Driver Machine Interface (DMI).*
MA.5 The *Driver* reads the authorized *MA*.

Each step of this scenario represents a permission to do an action on an entity by a role. On this basis, 3 roles (*OnboardSafetyManagement*, *TracksideSafetyManagement*, *Driver*), 3 system entities (*TracksideSystem*, *MA*, *DMI*) and 10 possible permissions (underlined actions) can be extracted.

Our approach consists in, on the one hand, the modeling of ERTMS operating rules in semi-formal UML notations with their graphical views and dedicated profiles extensions taking into account various aspects (structural, dynamic, behavioural, etc.), and on the other hand, their validation and verification with a formal *B* method with its mathematical notations and automated proof. The combination of these two notations has been studied and several approaches of UML to *B* translation have been proposed, cited in [3]. In order to model the scenario above, we use B4MSecure platform supporting the UML/B modeling process and lying within the scope of Model Driven Engineering (MDE).

For the sake of concision, the B4MSecure platform [7] is briefly presented. As an Eclipse platform, it is dedicated to formally validate a functional UML model enhanced by an access control policy. It uses a Role Based Acces Control (RBAC) profile inspired from SecureUML profile [4]. This profile aims at specifying information related to access control in order to model roles and their permisssions. This platform acts in 3 steps: a functional UML class diagram specifying system entities, security UML models with an access control policy and the translation of both models into *B* specifications.

Following the three-stepped approach of B4MSecure, a functional UML class diagram containing all system entities as classes and the relationships between them is built. Then, security UML class diagrams enhance the functional model by expressing which role has the permission to perform a given action in the railway system: a class diagram dedicated to the roles and others dedicated to the access control policies which are based on permissions linking the roles to the entities, such as the access control of the *MA* in Fig. 1. A permission is modeled as a UML association class, between a role and a class of the functional model, with a stereotype *Permission*. For instance, **MA.4** is modeled in Fig. 1 by the permission of the *OnboardSafetyManagement* to authorize the *MA*. All these diagrams are translated to *B* specifications.

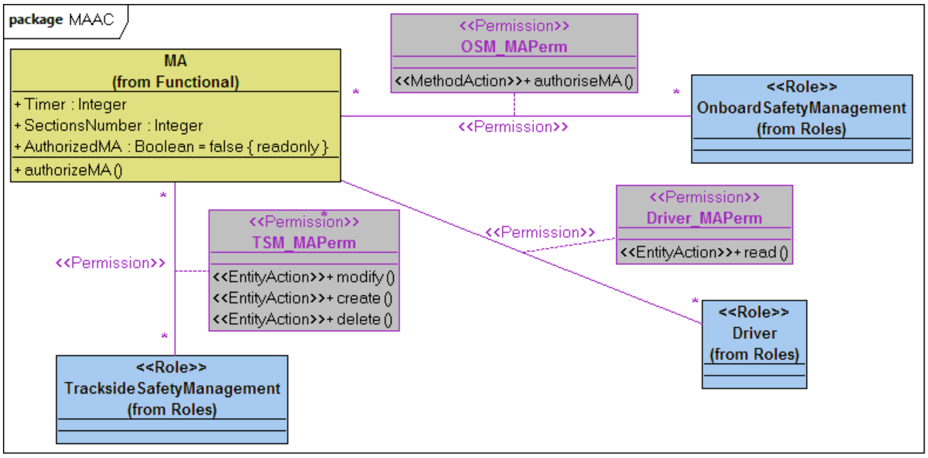

Fig. 1. Roles and permissions associated with MA

3 *B* Formal Validation of *Movement Authority*

The functional model is translated into a unique *B* machine, named *Functional*, and permissions are translated into a *B* machine, named *RBAC_Model*. As shown in Fig. 2, the functional formal model follows a classical translation scheme similar to [5]. The *RBAC_Model* adds variables about permissions and roles. For example, *PermissionAssignement* is a total function from *PERMISSIONS* to the cartesian product *(ROLES * ENTITIES)*, and *isPermitted* is a relation between *ROLES* and *Operations* sets. *PERMISSIONS*, *ENTITIES* and *Operations* are the sets defined in *RBAC_Model*, while *ROLES* is a set defined in the included *UserAssignments* machine. Initialization of these variables is conformant to the SecureUML model. Then, initialization proof obligation, produced by the AtelierB prover for these variables, allows to verify whether the SecureUML model respects RBAC well-formedness rules such as no cycles in role hierarchy, etc. The operations of the security formal model encapsulate the operational part of the functional formal model. Each functional operation is associated with an operation in the security model verifying that a user has permission to call the functional operation. For instance, *secure_MA__authorizeMA* operation of *RBAC_Model* checks the permissions associated with the functional operation *MA__authorizeMA*. *Secured* operations add a statement in the postcondition

```
Machine
 Functional

SETS
 MA_AS; ...

ABSTRACT_VARIABLES
 MA, ...

INVARIANT
 MA <: MA_AS & ...

INITIALISATION
 MA := {}  || ...

OPERATIONS

MA__authorizeMA(Instance)=
  PRE
  Instance : MA &
  MA__AuthorizedMA(Instance) = FALSE
  THEN
  MA__AuthorizedMA(Instance) := TRUE
END; ...

END
```

```
Machine
 RBAC_Model
INCLUDES
 Functional, UserAssignments
SEES
 ContextMachine
SETS
 ENTITIES = {MA_Label, ...};
 Attributes = {MA_AuthorizedMA_Label, ...};
 Operations ={MA_authorizeMA_Label, ...} ...
VARIABLES
 PermissionAssignement, isPermitted, ...
INVARIANT
 PermissionAssignement: PERMISSIONS --> (ROLES * ENTITIES)
 & isPermitted: ROLES <-> Operations ...
INITIAISATION
 PermissionAssignement :=
 {(OSM_MAPerm|->(OnboardSafetyManagement|->MA_Label)),...}
OPERATIONS
secure_MA__authorizeMA(Instance)=  PRE
 Instance: MA & MA__AuthorizedMA(Instance) = FALSE THEN
 SELECT MA__authorizeMA_Label : isPermitted[currentRole]
   THEN MA__authorizeMA(Instance)
   END
END; ...
END
```

Fig. 2. *Functional* and *RBAC_Model* machines

e.g *SELECT MA__authorizeMA_Label: isPermitted[currentRole]* in order to verify whether *MA__authorizeMA_Label* is allowed to the connected user using a particular role. Indeed, *isPermitted* computes, from the initial state, the set of authorized functional operations for each role.

UML models of extracted ERTMS/ETCS operating rules containing 7 functional classes, 5 roles and 17 permissions are transformed into 830 lines of functional formal model and 1545 lines of security formal model. We use the ProB animator in order to validate these specifications. A first animation checks the nominal behaviour of *Movement Authority.* Then variants of this animation check that the given permissions forbid the execution of secure actions by unauthorized roles, since a secure action can be performed only with a permission given to a role. The ability of the system specified by the class diagram to play the ERTMS scenarios is checked through animations of the corresponding transformed B model. These animations validate the permissions assigned to each role. But they don't check that the sequence of actions models the MA protocol. Actually, the sequence of actions is defined in the animation by the user, but it is not embedded in the UML/B model. This can be resolved by adding some contraints as preconditions in secured operations. Nevertheless, adding these conditions breaks the consistency between the UML model and the B machine. Owing to the lack of dynamic aspects in UML class diagrams, we intend to explore more UML diagrams as future work. Then we will focus on enriching UML class diagrams with, for instance, sequence diagrams which model the ordered interactions in scenarios and deriving *B* specifications from them in order to validate system's behavior.

At this stage, safety requirements have not yet been integrated to *B* specifications. As further work, we will consider enriching the *B* specifications with safety properties stemming from safety requirements of the ERTMS operating rules in order to formally verify them using the *B* prover. Moreover, SysML requirement diagrams combined with our UML diagrams may guarantee the traceability aspects of system requirements when they will be translated to *B* specifications.

4 Conclusion

In this short paper, we have presented a *Movement Authority* function extracted from the ERTMS/ETCS operating rules. This function was modeled using UML graphical notations and then translated automatically, via the B4MSecure platform, into *B* specifications which were checked successfully using the ProB animator. The combination of UML/B aims to ease the understanding of the system with the graphical notations of UML and formally validate system requirements with *B* formal notations. Research works done in the Selkis project [1], [2] and [3] show the efficiency of this platform and its different steps leading to the formal validation of scenarios in the healthcare Information Systems by seeking for malicious sequences of operations. However, in this paper, we show the use of this existant platform in another context related to distributed railway systems and their operating rules.

Acknowledgements This research is funded by the PERFECT project (ANR-12-VPTT-0010) and is partly supported by the SELKIS project (ANR-08-SEGI-018).

References

1. Ledru, Y., Idani, A., Milhau, J., Qamar, N., Laleau, R., Richier, J.-L., Labiadh, M.-A.: Taking into Account Functional Models in the Validation of IS Security Policies. In: Salinesi, C., Pastor, O. (eds.) CAiSE 2011 Workshops. LNBIP, vol. 83, pp. 592–606. Springer, Heidelberg (2011)
2. Milhau, J., Idani, A., Laleau, R., Labiadh, M.A., Ledru, Y., Frappier, M.: Combining UML, ASTD and B for the formal specification of an access control filter. In: Innovations in Systems and Software Engineering, vol. 7, pp. 303–313. Springer (2011)
3. Idani, A., Labiadh, M.A., Ledru, Y.: Infrastructure dirigée par les modèles pour une intégration adaptable et évolutive de UML et B. Ingénierie des Systèmes d'Information Journal 15, 87–112 (2010)
4. Lodderstedt, T., Basin, D., Doser, J.: SecureUML: A UML-Based Modeling Language for Model-Driven Security. In: Jézéquel, J.-M., Hussmann, H., Cook, S. (eds.) UML 2002. LNCS, vol. 2460, pp. 426–441. Springer, Heidelberg (2002)

5. Laleau, R., Mammar, A.: An overview of a method and its support tool for generating B specifications from UML notations. In: Proceedings of the 15th IEEE Interational Conference on Automated Software Engieering, ASE 2000, pp. 269–272. IEEE Computer Society, Washington (2000)
6. ERTMS, `http://www.ertms.net`
7. B4MSecure, `http://b4msecure.forge.imag.fr`