Checking the European Railways Traffic Management System (ERTMS) operating rules using UML and B method

Rahma Ben Ayed, Philippe Bon, Simon Collart-Dutilleul

To cite this version:

Rahma Ben Ayed, Philippe Bon, Simon Collart-Dutilleul. Checking the European Railways Traffic Management System (ERTMS) operating rules using UML and B method. 14th International conference on Railway Engineering Design and Optimization, Jun 2014, Italy. p139-149, 2014. <hal-01068376>

HAL Id: hal-01068376

https://hal.archives-ouvertes.fr/hal-01068376

Submitted on 25 Sep 2014

Checking the European Railways Traffic Management System (ERTMS) operating rules using UML and B method

Rahma Ben Ayed, Philippe Bon, Simon Collart-Dutilleul
Univ Nord de France,
IFSTTAR/COSYS-ESTAS
20 rue Élisée Reclus
F-59650 Villeneuve d'Ascq, France

Abstract

Interoperability is a critical factor for cost cutting and performance increasing in European railway exchanges. The European Railways Traffic Management System (ERTMS), which is both a specification and a technological framework, aimed at providing an answer to the above interoperability needs. Considering the implementation of ERTMS in a particular national context, operating rules must be compliant with the ERTMS specification, whereas the whole system has to provide some safety properties. Moreover, the management of railway signalling in ERTMS is based on "not on board rules" pertaining to each country and not on global rules. In consequence, it is difficult to evaluate the system in terms of safety. Thus, one of the main propositions of this study is to supply methodological tools for the evaluation of the global consistency between the specification and the operating rules, with regard to safety. This issue is crucial and yet it has scarcely been covered by scientific literature.

Keywords: ERTMS/ETCS, operating rules, functional requirements, safety requirements, *UML* modeling, formal validation, *B* method

1 Introduction

Currently, in European countries, each train has at least one specific railway system. Each system is therefore stand-alone and non-interoperable. This implies engineering efforts and important financial costs dedicated to the cross-border traffic. To cope with these problems, European Railway Traffic Management System (ERTMS) is designed to replace the existing national systems in Europe, to make rail transport safer and more competitive, and to improve cross-border connections. The presented work is a step forward in the French national research agency project named "Performing Enhanced Rail Formal Engineering Constraints Traceability" contributing to the validation and implementation of ERTMS. Its theme issues are based on analysing the European specification in front of national operating rules, as well as the verification with formal models to determine whether a given scenario fulfils the specification regarding the functional and safety requirements. In this context, we select two scenarii as a case study from the document of the description of principles and operating rules of ETCS system [1], applied on the European LGV-Est line. The modelling of the two scenarii does not correspond strictly to the operating rules in [1] which are not yet published. The reason is that the core document is still not the definitive version and we only want to present a scientific case study which does not have to integrate all the details. These scenarii are modelled with the semi-formal *UML* language and formally validated with the *B* method. Our motivation of using formal and graphical notations is their complementarity. The *UML* graphical modelling copes with the difficulties of understanding the system thanks to their synthetic, structural and intuitive aspects, while the *B* method copes with the lack of precision and the rigorous checking of systems with semi-formal models. There are several research works supporting $UML - B$ approaches referenced in [2].

The paper is structured as follows. Section 2 gives an overview of ERTMS/ETCS. Section 3 describes our case study featuring two scenarii: a nominal scenario (MA) and an exceptional/incident one (Override EOA). Section 4 describes *B4MSecure* platform used in *UML* modelling and its transformation into *B* specifications. Finally, section 5 concludes and discusses further work.

2 ERTMS/ETCS

The architecture of ERTMS/ETCS system referenced from the System Requirement Specification (SRS) of ERA [3] is now presented. The system includes two sub-systems: on-board and track-side. Fig. 1 depicts the architecture which encompasses on-board, track-side sub-systems and the different components linking them.

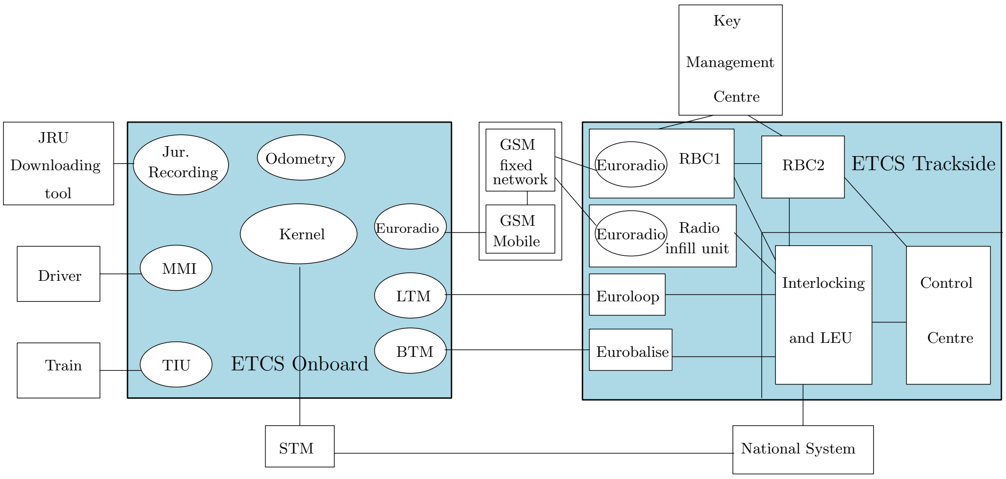

Figure 1: ERTMS/ETCS architecture

The on-board sub-system includes essentially on-board equipment and possibly the on-board part of GSM-R radio system according to the ERTMS/ETCS level. The on-board equipment is a computer-based system that supervises the movement of the train to which it belongs, on the basis of information exchanged with the track-side sub-system. According to the application level, track-side sub-system can be composed of the following elements: a)**Balise** is a transmission device that sends telegrams to the on-board sub-system. b)**Line-side electronic unit** is an electronic device that generates telegrams to be sent by balises, based on information received from external track-side systems. c)**Euroloop and Radio infill** provide signalling information in advance. d)**Radio Block Center (RBC)** is a computer-based system that elaborates messages to be sent to the train, based on information received from external track-side sub-systems and information exchanged with the on-board sub-systems. e)**Track-side radio communication network (GSM-R)** is used for the bi-directional exchange of messages between on-board sub-systems and RBC or radio infill units.

Interactions between on-board and track-side sub-systems depend on the level of ETCS application. There are several levels of using ERTMS/ETCS. The following case study uses the level 2. In level 2, train equipped with ERTMS/ETCS operates on a line controlled by RBC. Train location and train integrity supervision are performed by the track-side.

3 Case study

To cope with the sheer complexity of railway systems and their operating rules, two different scenarii are considered. The first one is a nominal scenario that describes *Movement Authority* function [4]. The second one is an exceptional/incident scenario that describes the *Override EOA* function.

3.1 Nominal scenario (Movement Authority)

Movement Authority (MA) is an authorization given to a train to move to a given point as a supervised movement. Some characteristics can be used to define a Movement Authority as follows [3]: a)**The End Of Authority (EOA)** is the location to which the train is authorized to move. b)**The target Speed at the EOA** is the permitted speed at the EOA. When the target speed is not zero, the EOA is called the limit of Authority (LOA). This target speed can be time limited. c)**The Danger Poin**t is a location beyond the EOA that can be reached by the front end of the train without risk of a hazardous situation. d)**Sections** represent the split of the MA. The last section is called *End Section*. e)**The time-out**: A first time-out value can be attached to each section. This value will be used for the revocation of the associated route when the train has not entered into it yet. It is called the *Section time-out*. A second time-out value can be attached to the End Section of the MA. This second time-out will be used for the revocation of the last section when it is occupied by the train. It is called the *End Section time-out*.

The on-board equipment can start an MA request addressed to the track-side via RBC. This latter proposes an MA that can be confirmed or rejected by the on-board equipment, after some verifications. To simplify our studied system, we assume that the RBC is a part of the track-side system.

3.2 Exceptional/incident scenario (Override EOA)

Override EOA is a function triggered by the driver in some specific degraded situations. When activated, this function allows a train to pass its EOA or an ETCS standstill. But, the driver must not overpass an EOA before receiving an authorization of Override EOA through a written order from the traffic agent. Before delving into details of this scenario, let us define a written order. A written order is a safety message sent by the traffic agent to the driver in order to provide some instructions. Written orders may be delivered to the driver physically or in a verbal communication by telephone or by radio according to the technical safety regulations relative to the communication. There are several written orders that can be used regarding the operating rules of ERTMS/ETCS level 2 applied to the LGV-Est line.

Therefore, *Override EOA* corresponds to ETCS01 written order. This latter includes as a minimum the type of authorization, number of authorization, post of deliverance, addressed train, applying position, time, date and actions to be taken. Orders and instructions, including those related to MA and Override EOA, are displayed as textual messages or symbols on the Driver Machine Interface (DMI). The DMI allows the communication between the on-board system and the driver. The driver can also inform the system by entering information via DMI.

Railway systems are critical systems, for which failures can lead to disastrous consequences. Therefore, the formal verification and the validation of its functional requirements have a prominent role since many of them are safety-critical. Our case study, based on these two selected functions of ERTMS/ETCS operating rules, shows the existence of interactions and requests for authorization between different agents acting on the system. In the remainder of this paper, the *UML* modelling [5] and the formal validation with *B* method [6] of our case study are presented. The first step presents the *B4MSecure* platform which supports the modelling process.

4 *B4MSecure* platform

This platform is the result of research work done by the VASCO team in the scope of the Selkis project, funded by the French national research agency (ANR) and aimed to defining a development strategy for a secure healthcare network IS from requirements engineering to implementation.

B4MSecure is an Eclipse platform dedicated to formally validate functional *UML* models enhanced by an access control policy which follows the Role Based Access Control (RBAC) model as a *UML* profile. A *UML* profile is defined as a generic extension mechanism for customizing *UML* models in particular domains and platforms. The used RBAC profile is inspired from SecureUML [7] which is a graphical modelling language specifying information related to access control with additional support for specifying authorization constraints in order to model roles and their permissions. Research works done in the Selkis project [8], [9] and [2] show the efficiency of this platform and its different steps leading to the formal validation of scenarii in healthcare domain by seeking for malicious sequences of operations.

B4MSecure acts in three steps:

- Graphical modelling using the Topcased tool of a functional *UML* class diagram.
- Graphical modelling of an access control policy using the RBAC *UML* profile.

- Translation of both models into *B* specifications in order to formally validate them.

4.1 *UML* modelling

In this step, a set of classes, related to requirements identified in a previous section, and relationships between them are described. We split our model into two packages, one package for the functional model and the other one for the security policy.

4.1.1 Functional model

As depicted in Fig. 2, the functional model contains MA and ETCSOrder classes for *Movement Authority* and *Override EOA* functions, respectively. The ETCSSystem class is composed of OnboardSystem class and TracksideSystem class corresponding to on-board sub-system and track-side sub-system, respectively. The onboard system is a part of the ERTMS/ETCS train, hence the relationship of aggregation between TrainETCS class and OnboardSystem class. The DMI allows the display of information about distance, speeds, ERTMS/ETCS level, ERTMS/ETCS mode and instructions as textual messages.

Features of MA and Override EOA appear as attributes of MA and ETCSOrder classes. The Override EOA function is modelled as an ETCSOrder class since it is a particular kind of ERTMS/ETCS written order. For this reason, the *AuthorizationType* attribute of type Enumeration *ETCSOrderNumber* is initialized to ETCS01.

4.1.2 Security Policy

The package of Policy contains all security models which assign permissions to roles acting on the entities of the functional model. Therefore, the package Policy contains sub-packages of roles (package Roles) and access control policies.

In the case study, five roles (Fig. 3) are extracted. The driver and the traffic agent roles represent the control/command staff. The OnboardSafetyManagement role and the TracksideSafetyManagement role correspond respectively to the onboard computer-based machine and the track-side computer-based machine. The ETCSCreator is a virtual role allowing the entities to be created. A role is modelled as a class with stereotype *Role*. Stereotypes are defined as extensions of *UML* in order to create new model elements derived from existing one, but that have specific properties that are suitable for a particular problem or a specialized usage.

Before delving into details about security models regarding the different entities modelled by the functional model, we outline the different interactions and the granted permissions according to our selected scenarii.

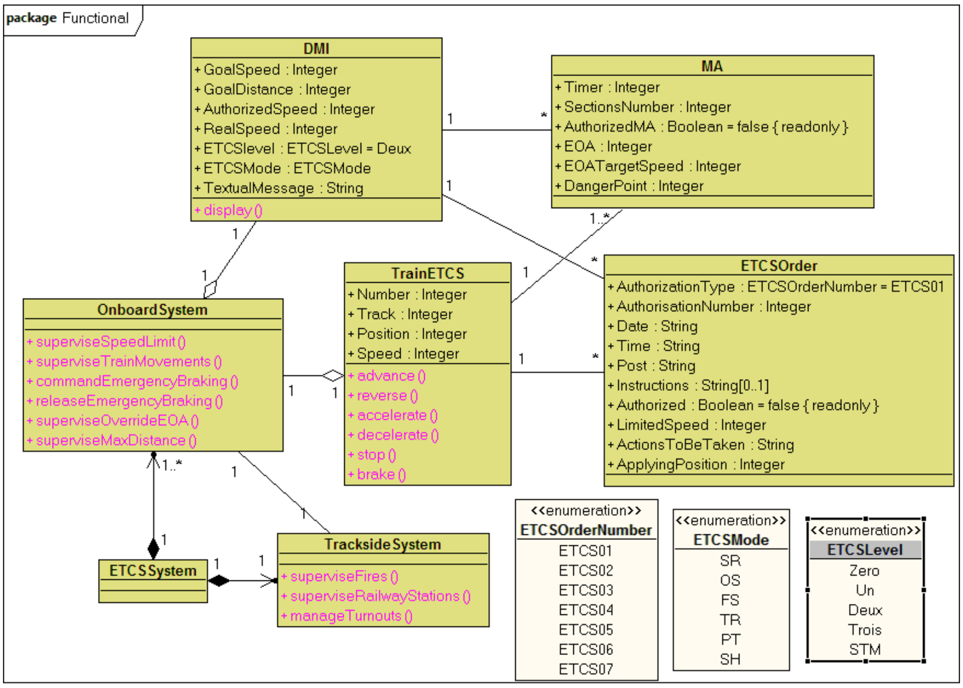

Figure 2: UML class diagram of functional model

The nominal scenario *Movement Authority* is composed of the following steps: **MA.1)** The OnboardSafetyManagement requests an MA to the TracksideSystem. **MA.2)** The TracksideSafetyManagement receives the request from TracksideSystem. **MA.3)** The TracksideSafetyManagement proposes an MA after some verifications. It can also create, 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 DMI. **MA.5)** The Driver reads the authorized MA.

The exceptional scenario *Override EOA* is composed of the following steps: **OverrideEOA.1)** The Driver requests an Override EOA through the DMI. He can also advance, stop, brake the train through the DMI. **OverrideEOA.2)** The OnboardSafetyManagement processes the request of Override EOA. **OverrideEOA.3)** The OnboardSafetyManagement transmits the request to the TracksideSystem. **OverrideEOA.4)** The TrafficAgent receives the request from the TracksideSystem. **OverrideEOA.5)** The TrafficAgent creates the ETCSOrder. He can also au-

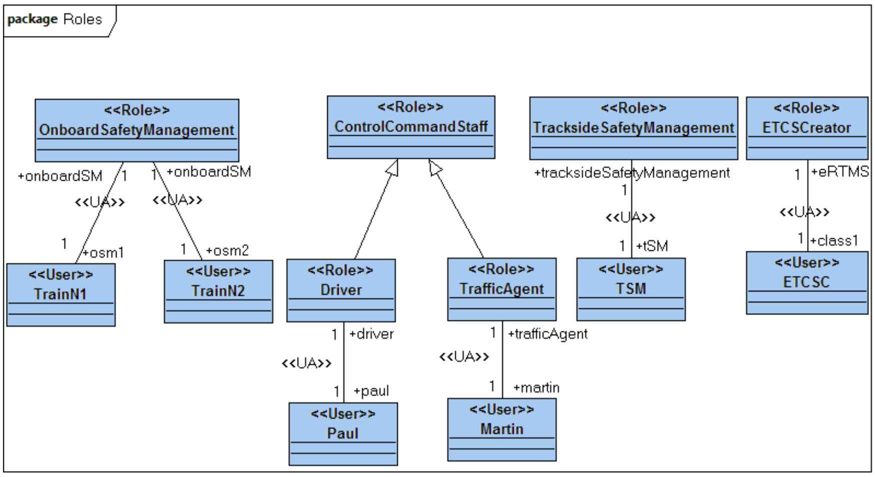

Figure 3: Roles and associated users

thorize, modify or delete it. **OverrideEOA.6)** The TrafficAgent transmits the authorization to the OnboardSystem. **OverrideEOA.7)** The OnboardSafetyManagement processes the authorization of the ETCSOrder in order to be displayed in the DMI. **OverrideEOA.8)** The Driver reads the authorized ETCSOrder.

Each step of these two scenarii represents a permission to do an action on an entity by a role. In *B4MSecure*, a permission is modelled in security models as a *UML* association class, between a role and a class of the functional model, with a stereotype *Permission* defined in the RBAC profile. For example, we note in Fig. 4 of *MAAC* package (Acces Control policy package of MA) the permission of TracksideSafetyManagment to create, modify and/or delete the MA (MA.3), the permission of the OnboardSafetyManagement to authorize the MA (MA.4) and the permission of Driver to read the authorized MA (MA.5).

Similarly, in *ETCSOrderAC* (Access Control policy package of ETCSOrder), we find the permission of the TrafficAgent to create, modify, delete and/or authorize ETCSOrder (OverrideEOA.5), etc.

4.2 Formal validation with *B* method

The use of formal methods is often advocated as a way of increasing confidence in critical systems like aeronautic and railway systems. In our research work, we have chosen the *B* method to formally validate models. The *B4MSecure* tool automatically translates both *UML* models into *B* specifications.

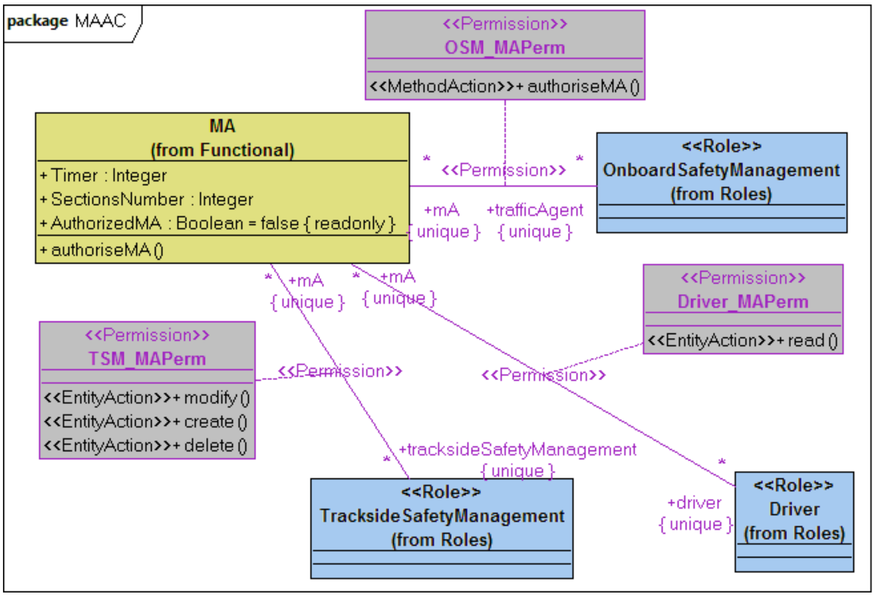

Figure 4: Roles and permissions associated with MA

4.2.1 *B* method

The *B* method is a formal specification method and a means of specifying, designing and coding software systems. It is based on a specification language and mathematical notations founded on first-order logic, integer arithmetic and set theory where the system state is modelled by abstract types of predefined data. It allows modelling of static and dynamic aspects of software structured in abstract machines. The static aspect is characterized by a set of variables and constants, whilst dynamic aspect is fulfilled by a set of operations. It covers all phases of software development life cycle: specification, refinement and implementation. *B* specification can be analysed using either animation tools such as ProB or proof tools such as Atelier-B. It checks correctness and consistency of the whole system.

4.2.2 Generated *B* specifications

In order to formally validate the *UML* functional and security models, we translate them through *B4MSecure* tool into *B* specifications. The functional model is translated into a unique *B* machine, named *Functional* (Fig. 5), and all security models are translated into a unique *B* machine, named *RBAC_Model* (Fig. 6). As shown below, the functional formal model follows a classical translation scheme, based on [10] and described in [2].

```
Machine
   Functional
SETS
   ETCSORDER
   ; MA_AS ; ...
ABSTRACT_VARIABLES
   ETCSOrder
   , MA, ...
INVARIANT
  ETCSOrder <: ETCSORDER
   & MA <: MA_AS & ...
INITIALISATION
   ETCSOrder := {}
   || MA := {}  || ...
```

```
OPERATIONS
ETCSOrder__authoriseETCSOrder(Instance)=
   PRE
   Instance : ETCSOrder &
   ETCSOrder__Authorised(Instance) = FALSE
   THEN
   ETCSOrder__Authorised(Instance) := TRUE
END;
MA__authoriseMA(Instance)=
   PRE
   Instance : MA &
   MA__AuthorisedMA(Instance) = FALSE
   THEN
   MA__AuthorisedMA(Instance) := TRUE
END; . . .
END
```

Figure 5: *Functional* machine

Security formal model *RBAC_Model* adds variables about permissions. For example, *PermissionAssignement* is total function from *PERMISSIONS* to the Cartesian product *(ROLES * ENTITIES)*, *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.

Operations of the security formal model add security aspects to the operational part of the functional formal model. Each functional operation is associated with a secured operation in the security model verifying that a user has permission to call functional operation. Operation *secure_ETCSOrder__authoriseETCSOrder* of *RBAC_Model*, as an example, adds security aspects to the functional operation *ETCSOrder__authoriseETCSOrder* of functional model.

Secured operations add a statement in the postcondition in order to verify whether operation *ETCSOrder__authoriseETCSOrder_Label* is permitted 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 our ERTMS/ETCS system 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. These formal models are animated successfully using ProB model animator.

5 Conclusion and future works

Our case study is based on the functional and safety requirements of ERTMS/ETCS distributed railway systems applied on the LGV-Est line. The aim of this

```
Machine
 RBAC_Model
INCLUDES
 Functional, UserAssignments
SEES
 ContextMachine
SETS
 ENTITIES =
  {MA_Label,
   ETCSOrder_Label, }
 Attributes =
  {MA_AuthorisedMA_Label,
   ETCSOrder_Authorised_Label}
 Operations =
  {ETCSOrder_authoriseETCSOrder_Label,
   MA_authoriseMA_Label,}
VARIABLES
 PermissionAssignement, isPermitted,
 ...
INVARIANT
 PermissionAssignement :
  PERMISSIONS --> (ROLES * ENTITIES)
 & isPermitted : ROLES <-> Operations
```

```
INITIAISATION
 PermissionAssignement :=
  {(OSM_MAPerm|->
     (OnboardSafetyManagement|->MA_Label)),
  (TA_ETCSOrderPerm|->
     (TrafficAgent|->ETCSOrder_Label)),...}
...
OPERATIONS
secure_ETCSOrder__authoriseETCSOrder(Instance)=
   PRE Instance :
    ETCSOrder & ETCSOrder__Authorised(Instance)
        = FALSE
   THEN
    SELECT ETCSOrder__authoriseETCSOrder_Label :
     isPermitted[currentRole]
    THEN ETCSOrder__authoriseETCSOrder(Instance)
   END
END;
secure_MA__authoriseMA(Instance)=
PRE Instance :
 MA & MA__AuthorisedMA(Instance) = FALSE
THEN
 SELECT MA__authoriseMA_Label :
  isPermitted[currentRole]
 THEN MA__authoriseMA(Instance)
END
END; . . .
END
```

Figure 6: *RBAC_Model* machine

case study is to outline the combination of *UML* graphical notations to ease the comprehension of the system and *B* formal notations to formally validate system requirements. The *UML* modelling of functional and security models and their translation into *B* specifications, using *B4MSecure* platform, depicts a clear separation of concerns stemming from ERTMS/ETCS requirements. Generated *B* models of our selected scenarii are checked successfully using ProB model animator.

UML supports modelling with different views and describes different aspects of system. In the case study there are a lot of actions and interactions between roles and entities of the system. Nevertheless, *B4MSecure* embedded no dynamic aspects of modelling. In a further work, *UML* dynamic diagrams will be explored. [11] shows how the requirements on the system as stated in sequence diagrams can be validated for *UML* models and proposes a translation of *UML* diagrams into a formal specification language CSP.

The idea is to derive *B* specifications from class diagrams and sequence diagrams, which state scenarii modelling the system's behaviour, in order to check the consistency of the sequential execution of scenarii operations taking into account access control policies.

Acknowledgements This research is funded by the Perfect project (ANR-12-VPTT-0010).

References

[1] Principes et règles d'exploitation du système ETCS - particularités en cas de superposition à un autre système de signalisation. Technical report, Réseau ferré de France, to be published.

[2] 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*, **15(3)**, pp. 87–112, 2010.

[3] ALCATEL, ALSTOM, ANSALDO SIGNAL, BOMBARDIER, INVENSYS RAIL & SIEMENS, System requirements specification, UNISIG subset-026, chapter 7. Technical report, European Railway Agency, 2006. Version 2.3.0.

[4] Ben Ayed, R., Collart-Dutilleul, S., Bon, P., Idani, A. & Ledru, Y., B formal validation of ERTMS/ETCS railway operating rules. *4th International ABZ 2014 Conference*, 2014. To be published.

[5] Unified modelling language version 2.1.1. technical report, OMG, 2007.

[6] Abrial, J.R., *The B Book: Assigning Programs to Meanings*. Cambridge University Press, 1996.

[7] Lodderstedt, T., Basin, D.A. & Doser, J., SecureUML: A UML-based modeling language for model-driven security. *UML*, eds. J.M. Jézéquel, H. Hußmann & S. Cook, Springer, volume 2460 of *Lecture Notes in Computer Science*, pp. 426–441, 2002.

[8] 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. *CAiSE Workshops*, eds. C. Salinesi & O. Pastor, Springer, volume 83 of *Lecture Notes in Business Information Processing*, pp. 592–606, 2011.

[9] 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. *ISSE*, **7(4)**, pp. 303–313, 2011.

[10] Laleau, R. & Mammar, A., An overview of a method and its support tool for generating B specifications from UML notations. *ASE*, pp. 269–272, 2000.

[11] Rasch, H. & Wehrheim, H., Checking the validity of scenarios in UML models. *Formal Methods for Open Object-Based Distributed Systems (7th FMOODS'05)*, eds. M. Stefen & G. Zavattaro, Springer-Verlag (New York): Athens, Greece, volume 3535 of *Lecture Notes in Computer Science (LNCS)*, pp. 67–82, 2005.