Analysis and Formal Modeling of Systems Behavior Using UML/Event-B

Kenza Kraibi, Rahma Ben Ayed, Simon Collart-Dutilleul, Philippe Bon, Dorian Petit

Analysis and Formal Modeling of Systems Behavior Using UML/Event-B

Kenza Kraibi[1], Rahma Ben Ayed[1], Simon Collart-Dutilleul[1,2], Philippe Bon[1,2], and Dorian Petit[1,3]
[1] Institut de Recherche Technologique Railenium, F-59300, Famars, France.
[2] IFSTTAR-COSYS-ESTAS, F-59666, Villeneuve d'Ascq, France.
[3] Université Polytechnique Hauts-de-France, LAMIH UMR CNRS 8201, F-59313 Valenciennes, France.
Email: kenza.kraibi@railenium.eu; rahma.ben-ayed@railenium.eu; simon.collart-dutilleul@ifsttar.fr; philippe.bon@ifsttar.fr; dorian.petit@uphf.fr

Abstract —The verification of safety properties of critical systems, such as railway signaling systems, is better achieved by formal reasoning. Event-B as a formal method, allows to get safe and reliable systems. Nevertheless, modeling with Event-B method requires some knowledge on mathematical logic and set theory. In opposition, UML (Unified Modeling Language) is a commonly used graphical language, but it does not guarantee the verification of safety properties. This paper presents an approach combining UML and Event-B. In fact, we focus in this work on modeling the systems behavior with the joint use of some UML behavioral diagrams. The UML models are then translated into Event-B models for the systems validation as well as the verification of safety properties using B tools. This methodology is illustrated by an application on a case study of railway signaling system.

Index Terms— Event-B, UML, Behavior, Formal Verification, Safety, Railway Signaling.

I. Introduction

The dynamic behavior of critical systems is often expressed by formalism of automata or state machines as well as by various notations of graphical sequencing procedures. However, the absence of explicit formalization of these notations does not facilitate modeling. Within PRESCOM project, we are interested in modeling and analyzing formally the behavior of railway signaling systems. Railway signaling is, by nature, safe train movement management. It is based either on directives, in the form of procedures to be respected by railway actors, or it specifies dynamic behaviors to be respected for the control-command systems programming.

The goal of this work is to formalize the structuring of dynamic behaviors and define an explicit syntax for their description. UML [1] as a known standard allows better understanding of the system structure but it lacks precise semantic description and does not guarantee the formal verification and validation which are highly recommended for safety-critical systems such as railway signaling systems. As for the formal Event-B method [2], the extension of B method [3], its models have a strictly defined semantics and leave no possibility of divergence in their interpretation. However, these models require more advanced expertise and especially a good level in mathematics. In order to model graphically and reason formally on the specification, we use on the one hand UML behavioral dynamic diagrams: sequence and state machine. On the other hand, we translate the UML models into Event-B models for the systems validation as well as the verification of safety properties. In fact, our proposition is the joint use of sequence and state machine diagrams using a UML profile. This profile allows the restriction of UML modeling by combining both of the behavioral diagrams. Once the UML models are transformed into Event-B models, they are validated using the formal proof techniques.

This paper is organized as follows. Section 2 gives an overview of the approach. Section 3 emphasizes the application of this approach to a railway case study. Then, section 4 illustrates related work and discussion. Finally, we conclude and provide some ideas for future work.

II. Methodology

Fig. 1 illustrates our three-stepped methodology. In a first step, we model in UML the system behavior stemmed from an informal specification. For this purpose, we use several types of UML diagrams: class, sequence and state machine diagrams, in order to obtain a complete model or almost complete in Event-B. In a second step, the models are transformed into Event-B. For the class diagram modeling and transformation, we opt for the use of B4MSecure tool to get sets, variables and invariants. The latter is recently used to structure B specifications of railway operating rules considering safety properties [4], [5]. B4MSecure does not handle the transformation of sequence and state machine diagrams. For this reason, the proposed work can be an extension of this tool. Thereafter, in a third step, we proceed with the validation of the scenarios and the verification of safety properties.

Manuscript received September 15, 2018; revised October 25, 2018.
This work was supported by IRT Railenium within PRESCOM project.
Corresponding author email: kenza.kraibi@railenium.eu
doi:10.12720/jcm.v.n.p-p

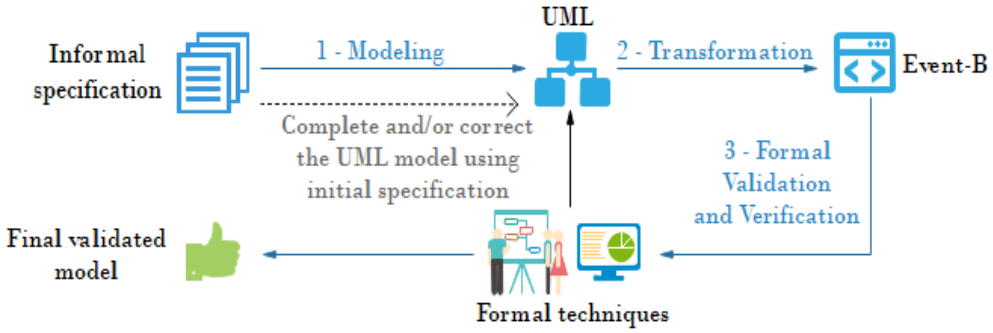


Fig. 1. Steps of the methodology

Step 1: Modeling of Behavior. In UML sequence diagrams, an exchanged message between the actors of the system can be sending of a signal, invocation of an operation and creating or destructing an object. When the message is an invocation of a class operation, the state's changes of the class attributes cannot be specified directly in a sequence diagram. It is necessary to provide a specification for these states changes to complete the modeling of system behavior. Otherwise, the transformation of the invocations of class operations will provide only the signatures of Event-B events [6]. This can be performed by conditions in class operations in the UML class diagram. But, UML class diagram is structural in nature.

We suggest here to model the state machine diagrams, which corresponds to an instance of a class, and to specify the state machine guards and transitions. Then, we define a UML profile which combines sequence and state machine diagrams. This profile regroups the different possible state changes in the sequence diagram for each message. Fig. 2 describes this UML profile:

- *"Trans":* extension of the meta-class *"Transition"* with tagged values *"guard"* (constraint) and *"action"* (behavior).
- *"msg":* extension of the meta-class *"Message"* takes *"Trans"* as the type of its first tagged value *"trans"* and defines *"refine"* to present the refined message if any.

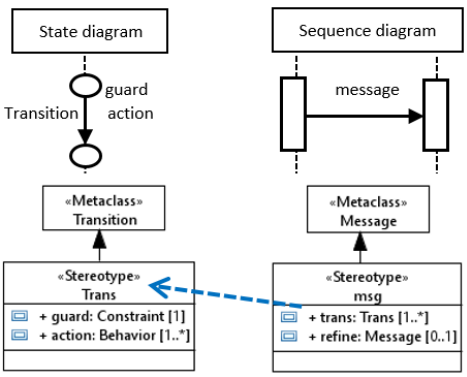


Fig. 2. UML profile combining sequence and state machine diagrams

Step 2: Transformation from UML to Event-B. A message is defined by a guard, one or more actions and the refined message (in case this message refines another one). Therefore, each **message** sending and receiving gives rise to an event. The event *"message$_0$ ≙ ..."* is the translation of a message where the tagged value *"refine"* is empty, whereas the event *{message$_1$ ref message$_0$ ≙ ...}* is the translation of a message where *"refine"* takes as value *"message$_0$"*.

In order to represent the **messages sequencing** of UML sequence diagrams, we use a variable of sequence *"seq"* whose value is incremented at each event execution in order to guarantee this sequencing (*"msg$_{n+1}$"* starts only after *"msg$_n$"*).

EVENTS

msg$_n$ ≙
guard: guard$_{msgn}$ & seq = n
action: act$_{msgn}$ || seq := seq + 1

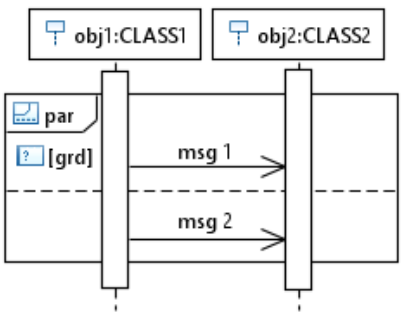
msg$_{n+1}$ ≙
guard: guard$_{msgn+1}$ & seq= n+1
action: act$_{msgn+1}$ || seq := seq + 1

In sequence diagrams, combined fragments group conditional structures such as parallelism, loop and alternative.

The **parallel fragment** contains several asynchronous messages. They are transformed by regrouping their actions together in one single event in the form of parallel actions:

"parallel_event ≙ guard: grd action: act$_{msg1}$ || act$_{msg2}$".

Despite the enforcing of synchrony in this Event-B translation, we avoid the indeterministic sequencing of events when messages are translated separately into events [7].



The **reference fragment** serves for including an interaction into another one in the same level of hierarchy. However, we transform this fragment into an event, that contains only the guard and the action of sequencing, in the abstract machine. Then, the messages of the reference interaction will be translated in the refinement machine in the form of new events refining the reference event.

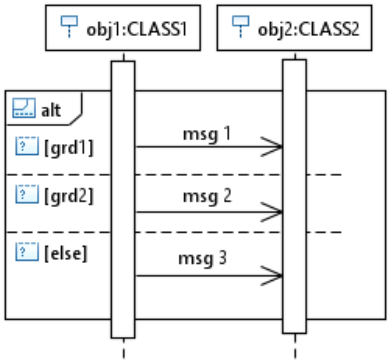
EVENTS

interaction ≙ guard: seq = n
action : seq := seq + 1

EVENTS

Message ref interaction ≙ guard: $grd_{Message}$
action: $act_{Message}$

In the **alternative**, each message sending and reception is transformed into an event. The *'else'* message is presented by an event triggered when all the other guards are not satisfied.



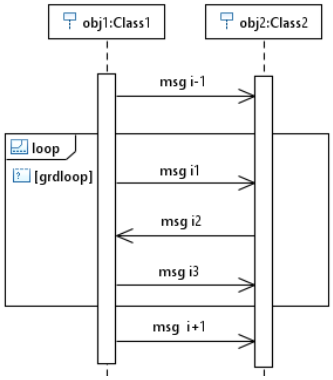
EVENTS

msg_1 ≙ guard: grd_1 & not(grd_{msg2})
action: act_{msg1}

msg_2 ≙ guard: grd_{msg2} & not(grd_{msg1})
action: act_{msg2}

msg_3 ≙ guard: not(grd_{msg1}) & not(grd_{msg2})
action: act_{msg3}

A **loop combined fragment** is repeated at least *"minint"* times. As long as a guard *"grdloop"* is true, the loop continues, at most *"maxint"* times. The variable *"itloop"* specifies here the iteration number. Each message in the loop is translated into an event "msg_{ij}".



The variable *"seqloop"* ensures the internal sequencing of the loop events. When *"grdloop"* is no longer satisfied or the *"maxint"* iterations are reached, the loop stops: the sequencing variable of the entire interaction *"seq"* is incremented in an additional event *"$msg_{i'}$"* in order to leave the loop. Then, the next event *"msg_{i+1}"* is triggered.

EVENTS

msg_{i-1} ≙ guard: grd_{msgi-1} & seq = i-1
action: act_{msgi-1} || seq := seq+1

msg_{i1} ≙ guard: grdloop & grd_{msgi1} & seq = i & seqloop = 1 & itloop < maxint
action: act_{msgi1} || seqloop := seqloop+1

msg_{i2} ≙ guard: grd_{msgi2} & seqloop=2 & seq=i
action: act_{msgi2}||seqloop:= seqloop+1

msg_{i3}≙ guard: grd_{msgi3} & seqloop=3 & seq=i
action : act_{msgi3} || itloop:=itloop+1 || seqloop:=1

$msg_{i'}$ ≙ guard: (not(grdloop) or itloop = maxint) & (itloop>minint)
action: seq:=seq+1

msg_{i+1} ≙ guard: grd_{msgi+1} & seq = i+1
action: act_{msgi+1} || seq := seq+1

Step 3: Formal Verification and Validation. The obtained models must be proved in order to ensure the verification of safety properties and to validate the system behavior. There are several techniques for the formal verification and validation, we use in this approach "Atelier B" tool which makes possible discharging the proof obligations. In addition, we use the animation tool "*ProB*" [8], that identify the errors which are not easily discovered by "Atelier B". These tools are advocated as a way of increasing confidence in the system specifications. They are used in a complementary manner. In fact, "Atelier B" generates the proof obligations and proves automatically some of them. "*ProB*" animates the Event-B specifications, detects the invariants violations, and thus eases the continuation of the second stage of interactive proof.

III. Application to Railway Signaling Case Study

The application of formal methods to the railway domain have been investigated by previous research projects: *PERFECT*[1] for modeling railway operating rules [4], [5] and *NExTRegio* [2] for modeling the railway signaling system. The ultimate goal is to produce methods for modeling railway systems efficiently while ensuring safety. As a continuity of these railway projects, *PRESCOM* reveals in particular the modeling of systems behavior. To illustrate the approach, we apply the methodology to a railway case study, also studied by

[1] *PERFECT*: http://www.agence-nationale-recherche.fr/Projet-ANR-12-VPTT-0010
[2] *NExTRegio*: an IRT Railenium project in partnership with SNCF Réseau and Clearsy Systems Engineering.

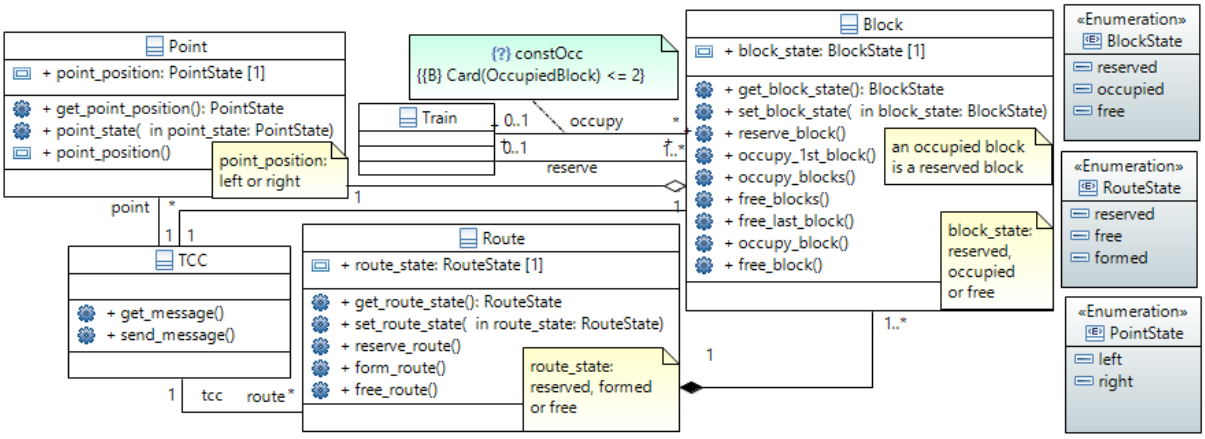


Fig. 3. Class diagram of the system

Abrial in [2]: Train Control System that helps the traffic regulator to control the train movements. We rely on the same requirements specification as Abrial's case study.

Step 1: Modeling. The system is in charge of controlling the reservation process and the trains movements. The reservation process performs the reservation of routes, the positioning of points and the signal setting. The control of train movement manages the process of occupying and freeing the routes. Fig. 3 shows a part of the system specification as a UML class diagram. The model contains several state machine and sequence diagrams. We present only the *Route* state machine and the *route reservation* interaction. Fig. 4 describes the behavior of a *Route* instance. Fig. 5 shows two interactions: *route reservation* and *route formation.*

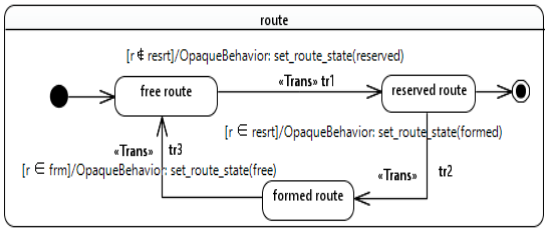


Fig. 4. State machine diagram of Route

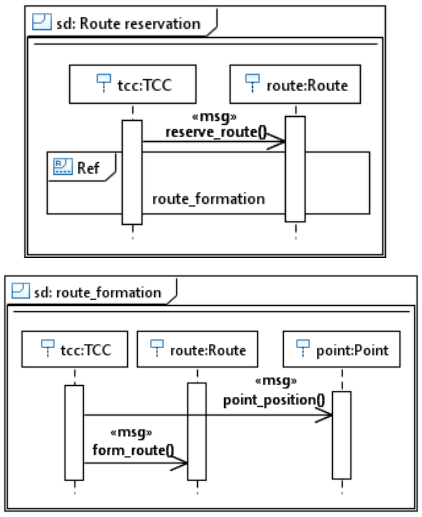


Fig. 5. Sequence diagram of route reservation and route formation

Step 2: Model Transformation. Fig. 6 illustrates an excerpt of the Event-B models resulting from the UML sequence diagrams of Fig. 5. In the abstract machine, two events are generated from the *route reservation* interaction. The events signatures are obtained from the messages. Guards and actions are obtained from the transitions of state machine. "route_formation" event contains only the sequencing variable. As explained in section 2, the messages of the reference interaction are translated in the refinement. Fig. 6 shows the transformation of reference interaction. Both of the *"point_position"* and the *"form_route"* events refine *"route_formation"*.

SYSTEM
M
SETS
ROUTE; BLOCK
CONSTANTS
rtbl // routes containing blocks
VARIABLES
seq, // sequencing variable
resrt, resbl //reserved routes and blocks
INVARIANT
Resrt ⊆ ROUTE ∧ resbl ⊆ BLOCK
∧ seq ∈ ℕ ∧ rtbl ∈ BLOCK↔ROUTE
∧dom (rtbl) = BLOCK ∧ ran (rtbl) = ROUTE
INITIALISATION resrt :=∅ || resbl :=∅|| seq := 0
EVENTS
reserve_route ≙
ANY rr
WHERE rr ∈ROUTE ∧ rr ∉resrt ∧ seq = 0
∧ $rtbl^{-1}$[{ rr }] ∩ resbl = ∅
THEN resrt := resrt ∪ {rr} || seq := seq + 1
|| resbl := resbl ∪ $rtbl^{-1}$ [{ rr}]
END;
route_formation ≙
SELECT seq = 1
THEN seq := seq + 1
END
END

Fig. 6. Abstract Machine

```
REFINEMENT
M_ref

REFINES
M

SETS
POINT //Points set

VARIABLES
frm, //Set of formed routes
pst //Set of positioned points

INVARIANT
frm ⊆ resrt
∧ pst ⊆ POINT

INITIALISATION
frm := ∅

EVENTS
//reserve_route is also refined here
        point_position ref route_formation ≙
        ANY
                pp, rr
        WHERE
                pp ∈POINT - pst
                ∧ rr ∈ resrt - frm
                ∧ seq = 1
        THEN
                seq := seq + 1
                ∧ pst := pst ∪ {pp }
        END;
        form_route ref route_formation ≙
        ANY
                rr, pp
        WHERE
                rr ∈ resrt - frm
                ∧ pp ∈ pst
                ∧ seq = 2
        THEN    frm :=  frm ∪ {rr}
                ∧ seq := seq + 1 END
END
```

Fig. 6. Refinement

Step 3: Formal Verification and Validation. In order to check safety properties and validate the Event-B models, we use B tools, *"Atelier B"* for the proof and *"ProB"* for the animation. Actually, safety requirements of railway signaling are expressed in B language in the form of constraints while modeling in UML. Then, they are imported during the transformation into Event-B models. We notice that the obtained Event-B models verify the identified railway safety properties. Among them, we quote:

- To avoid overtaking, a reserved block cannot be occupied by any other train than the one which reserved it. In other words, for each block belonging to the set of occupied blocks *"occbl"*, an occupied block by a train is a reserved block by this train. This is expressed by this invariant:

! block.(block : occbl => reserve(block) = occupy(block))

with *reserve : BLOCK +-> TRAIN* and *occupy : BLOCK +-> TRAIN* are two partial functions from the set of "BLOCK" to the set of "TRAIN".

- The occupied blocks cannot change their state to free without ensuring if the train has left these blocks. Specifically, each block neither reserved nor occupied is not associated to any train:

! block.(block : BLOCK-{occbl \/ resbl} => block /: dom(reserve) & block /: dom(occupy))

IV. Related Work and Discussion

Our approach combines a diagrammatic modeling notation (UML) with a formal modeling notation (Event-B). Essentially, UML diagrams model graphically the specification, in particular the behavioral aspect of the system, whereas Event-B serves for the verification of safety properties.

In this scope, several UML/B and UML/Event-B coupling approaches have been studied. In [9], [10], a methodology suggests combining UML and B for modeling a railway case study (level crossing) using UML use case, class and state machine diagrams. However, this methodology lacks modeling the dynamic proceeding. Similarly, for [11], the proposed verification approach, using UML class, collaboration and state machine diagrams beside B method, does not address this issue. This same issue is nevertheless raised in [4], [5], where a UML/B modeling approach is proposed for the validation of railway safety operating rules using *B4MSecure* tool. Authors of this work show the limits of scenarios modeling in *B4MSecure*.

In [12], a UML modeler is proposed: UML-B plugin which is a UML-like and graphical front-end for Event-B. This graphical approach restrains modeling of UML class diagrams and state machine diagrams [13], [14]. It lacks some characteristics comparing with standard UML tools with respect to the Object Management Group (OMG), e.g. the enumeration for class diagram and the terminate pseudo-state for state machine diagram. In [15], the authors use sequence diagrams and transform them to B by expressing the sequence in the refinement as operations call, which is not possible in Event-B.

In [6], the authors present the translation of use case and sequence diagrams into Event-B. Despite its modeling of system behavior, this method provides only the signatures of events and does not express explicitly the state change of variables, i.e. the obtained events of Event-B model do not contain the body specification.

Whereas [7] focuses on the translation of activity diagrams into Event-B for distributed and parallel applications. This approach does not guarantee a parallel execution of the events, and it is presented as an interlaced execution of indeterministic events. Most of these works, do not consider refinement.

Our contribution in this paper deals with this shortcoming in addition to the dynamic proceeding modeling issue.

In a nutshell, the proposed methodology relies on: Specifying the body of Event-B events besides their signatures, considering the expression of complex behaviors such as parallelism, loop and alternative ones and taking in consideration the refinement.

V. Conclusion and Future Work

In this paper, a UML/Event-B approach for analysis and formal modeling of systems behavior is presented. As a first step, we model the system from an informal specification using several UML diagrams. Furthermore, we combine the UML sequence and state machine diagrams by means of a UML profile. Then, the dynamic behavioral models of UML are transformed to Event-B models. Finally, we proceed with the formal verification and validation using B tools. In order to illustrate the contribution, this approach is applied on a case study of a railway signaling system.

The use of UML and Event-B for modeling the behavior of safety-critical systems aims at two-fold goal. It makes easier to communicate and more understandable through the UML graphical modeling. In addition, it deals with different issues related to the formal validation of dynamic behavioral systems using Event-B formal method. The exploration of these transformation results allows identifying some limits of the syntax and the semantics of Event-B language with respect to the behavioral modeling. In fact, the underlying Event-B syntax and semantics do not allow modeling the behavior of the system without the use of ad-hoc variables such as those used for the combined fragments transformation (loop fragment for example).

Our goal is to seek for a better standardization of structuring dynamic behaviors and in particular for an explicit syntax for their specification. Also, we are working on the development of the transformation plugin from UML into Event-B as well as the expression of the constraints in a language which can be validated by UML modeler and transformed automatically to Event-B constraints, for instance OCL language [16].

Acknowledgment

This work is supported by **PRESCOM** (Global safety proofs for modular design/**PRE**uves de **S**écurité globale pour la **CO**nception **M**odulaire) as a part of IRT Railenium projects in collaboration with ClearSy Systems Engineering.

References

[1] OMG: Unified Modeling Language (OMG UML), superstructure. Technical report, version 2.4. 1. Tech. rep., Object Management Group (2011)

[2] Abrial, J.R.: Modeling in Event-B: System and software engineering. Cambridge University Press, New York, NY, USA (2010)

[3] Abrial, J.R.: The B-Book: Assigning programs to meanings. Cambridge University Press, New York, NY, USA (1996)

[4] Ben Ayed, R., Collart-Dutilleul, S., Bon, P., Idani, A., Ledru, Y.: B formal validation of ERTMS/ETCS railway operating rules. In: International Conference on Abstract State Machines, Alloy, B, TLA, VDM, and Z, Springer (2014) 124{129}

[5] Ben Ayed, R., Collart-Dutilleul, S., Bon, P., Ledru, Y., Idani, A.: Formalismes basés sur les rôles pour la modélisation et la validation des régles d'exploitation ferroviaires. Technique et Science Informatiques (TSI) 34(5) (2015) p495{521}

[6] Weixuan, S., Hong, Z., Yangzhen, F., Chao, F.: A method for the translation from UML into Event-B. In: Software Engineering and Service Science (ICSESS), 2016 7th IEEE International Conference on, IEEE (2016) 349{352}

[7] Ben Younes, A., Jemni Ben Ayed, L.: Using UML activity diagrams and Event B for distributed and parallel applications. In: Computer Software and Applications Conference, 2007. COMPSAC 2007. 31st Annual International. Volume 1., IEEE (2007) 163{170}

[8] Leuschel, M., Butler, M.: ProB: A model checker for B. In: FME. Volume 2805., Springer (2003) 855{874

[9] Boulanger, J.L., Bon, P., Mariano, G.: From UML to B {a level crossing case study. WIT Transactions on The Built Environment 88 (2006)

[10] Boulanger, J.L.: Formal methods applied to industrial complex systems: Implementation of the B method. John Wiley & Sons (2014)

[11] Truong, N.T., Souquières, J.: Verification of UML model elements using B. Journal of Information Science and Engineering 22 (2006) 357{373}

[12] Snook, C., Butler, M.: UML-B: Formal modeling and design aided by UML. ACM Transactions on Software Engineering and Methodology (TOSEM) 15(1) (2006) 92{122}

[13] Snook, C., Butler, M.: UML-B and Event-B: An integration of languages and tools. (2008)

[14] Said, M.Y., Butler, M., Snook, C.: A method of refinement in UML-B. Software & Systems Modeling 14(4) (2015) 1557{1580}

[15] Truong, N.T., Souquières, J.: Test of object-based specifications using B notations. (2005)

[16] Ledang, H., Souquières, J.: Integration of UML and B specification techniques: Systematic transformation from OCL expressions into B. In: Software Engineering Conference, 2002. Ninth Asia-Pacific, IEEE (2002) 495{504}

[17] S. Chen, B. Mulgrew, and P. M. Grant, "A clustering technique for digital communications channel equalization using radial basis function networks," *IEEE Trans. on Neural Networks*, vol. 4, pp. 570-578, July 1993.

[18] J. U. Duncombe, “Infrared navigation—Part I: An assessment of feasibility,” *IEEE Trans. Electron Devices*, vol. ED-11, pp. 34-39, Jan. 1959.
[19] C. Y. Lin, M. Wu, J. A. Bloom, I. J. Cox, and M. Miller, “Rotation, scale, and translation resilient public watermarking for images,” *IEEE Trans. Image Process.*, vol. 10, no. 5, pp. 767-782, May 2001.

Kenza Kraibi received her Master’s degree from the Department of Computer Science, Faculty of Science, Mohammed V University, Rabat, in 2017. Currently, she is a Ph.D. student in the Computer Science field in IRT Railenium, France. Her research fields of interest are mainly in global safety proofs for modular design, railway signaling, formal methods and safety of critical systems.

Rahma Ben Ayed is a research engineer in IRT Railenium since 2016. She received her Ph.D. in ESTAS laboratory of COSYS department at the French Institute of Sciences and Technologies of Transport, Planning and Networks (IFSTTAR) in 2016. Her thesis deals with the UML / B modeling for the validation of the safety requirements of railway operating rules. In 2011, she obtained her master's degree in software engineering and decision support and, in 2010, her engineering degree from the National School of Computer Science (ENSI) in Tunisia.

Simon Collart-Dutilleul is a doctor of the University of Savoy in 1997. He then occupied a post of lecturer at Centrale Lille from 1999 until 2012. He supports an Habilitation to lead research in 2008. Since 2012 he is a director of research at ESTAS laboratory of IFSTTAR where he conducts works on the validation of railway safety constraints using formal methods. Co-author of more than one hundred scientific publications, his research areas are rail transport systems, transport safety and formal methods and discrete event systems. He currently heads the ERTMS workgroup at IFSTTAR.

Philippe Bon is a researcher in ESTAS laboratory of COSYS department at the French Institute of Sciences and Technologies of Transport, Planning and Networks. He holds a Ph.D. from the University of Lille since 2000. His work focuses on the implementation of requirements traceability throughout the design cycle of railway systems. He has participated in several research projects related to the use of formal methods for traceability and validation (SafeCode, TUCS, Perfect).

Dorian Petit is a researcher in LAMIH and an assistant professor in Polytechnic Hauts-de-France University with a Ph.D. in Computer Science from Valenciennes University in 2003. He has been a head of IT department of IUT from 2008 to 2011 and researcher during 2012 in ESTAS laboratory of COSYS department at the French Institute of Sciences and Technologies of Transport, Planning and Networks. His work focuses on the quality of software development in particular using formal methods and on the assessment of software for safety-critical systems.