Transportation Research Part C 42 (2014) 60–75

Contents lists available at ScienceDirect

Transportation Research Part C

journal homepage: www.elsevier.com/locate/trc

Formalizing a subset of ERTMS/ETCS specifications for verification purposes

Mohamed Ghazel

French Institute of Sciences and Technology for Transport, Development and Networks (IFSTTAR/COSYS-ESTAS), Univ Lille Nord de France, F-59000 Lille, France

ARTICLE INFO

Article history:
Received 18 November 2013
Received in revised form 17 January 2014
Accepted 5 February 2014

Keywords:
ERTMS/ETCS
Railway interoperability
Railway safety
Specification
Formalization
Verification and validation
Model-checking

ABSTRACT

ERTMS is the standard railway control-command and signaling system which aims to ensure railway interoperability throughout Europe while enhancing safety and competitiveness. ERTMS is composed of two main subsystems which include GSM-R, a radio system for enabling communication between the train and the traffic management center and ETCS, an automatic train protection system (ATP) to replace the existing national ATP systems. The ERTMS specifications are defined by means of standard documents which set out the requirements ensuring interoperability. These documents evolve regularly to give rise to successive versions. The ERTMS/ETCS standard defines different levels and operation modes according to various trackside and onboard setups and some operational conditions. Given the complexity and the high criticality of railway operation, verification and validation (V&V) are crucial tasks in railway application development.

In this paper, after setting the background and the motivations, a mechanizable formalization of a subset of ERTMS/ETCS specifications relative to ETCS modes and transitions is developed. The present work aims to offer a readily available model for formal V&V. Using formal techniques to check SRS is highly recommended to tackle the complexity of the defined requirements and prevent specification errors. Model-checking technique, which is targeted here, offers exhaustive analysis of the system behavior based on its model and is highly automated, since it is supported by software tools. Based on the last available version of SRS specifications, a progressive process is undertaken to get a formal model which makes explicit the various modes characterized by their respective active functions, as well as the numerous combinations of conditions for switching between modes. The various steps guiding the translation of the SRS literal specifications into a formal model are explained. As will be shown through different examples, the obtained model is a convenient basis to check safety, interoperability and liveness properties.

1. Background and Motivations

The European railway sector is characterized by high fragmentation due to the coexistence of a variety of railway signaling and control systems issued by different manufacturers. This situation represents a major obstacle for railway interoperability across Europe. Moreover, the deployment of proprietary systems deprives rail companies of economies of scale that would be generated if standard systems were implemented. At the same time, the emergence of new high-speed lines in Europe and the development of several freight corridors create a demand for increasing cross-border traffic. In addition,

E-mail address: mohamed.ghazel@ifsttar.fr

http://dx.doi.org/10.1016/j.trc.2014.02.002
0968-090X/

some railways anticipate a significant increase in the traffic density and are rethinking their infrastructure strategy to accommodate to heavy traffic in which ATC/ATP[1] systems play an important part.

Faced with this situation, the European Commission (EC) took the initiative to order the development of a new standard railway control and signaling system. In particular, Directive 96/48/EC (Council Directive, 1996) enforces "*the ability of the Trans-European high-speed rail system to allow the safe and uninterrupted movement of high-speed trains which accomplish the specified levels of performance*". Consequently, a major European project called ERTMS (European Rail Traffic Management System) has been established to ensure railway interoperability across Europe and enhance railway transport safety and competitiveness and increase track utilization. The ERTMS system is composed of two main components: ETCS (European Train Control System), implemented, in part, onboard trains and partly as a fixed infrastructure, which fulfills train control-command functionalities, and GSM-R, which is the communication system between the ETCS onboard subsystem and the ETCS trackside subsystem, through a reliable and timely data/voice exchange. ERTMS/ETCS is going to become the reference standard for modern railway signaling throughout Europe and even beyond. The first versions of ERTMS specifications include mainly the Functional Requirements Specifications (FRS) (ERTMS/ETCS, 2007) which identify the functions required for technical interoperability and the System Requirements Specifications (SRS[2]) that define the system requirements for ETCS. However, an amendment has been brought through the European commission decision of 6 November 2012 (Commission Decision, 2012) making the FRS no longer mandatory. SRS specifications are mostly written in natural language and are far from being formal. This is a major issue when dealing with such systems since, by nature, literal specifications often hold ambiguities as they can be subject to different interpretations. Requirement specification is a major concern when dealing with complex critical systems in which errors may have serious human and/or material consequences. The part of the bugs due to requirement specification is very significant in complex systems. For instance, a study pertinent to computerized railway signaling systems showed that about 75% of accidents/incidents involving safety issues are due to specification issues (Antoni, 2012). Similarly, another study that has been led by the automobile branch of Bosch revealed that 60% of bugs are due to the requirement specification phase; namely ambiguity, inaccuracy and inconsistency (Ghazel and Mekki, 2010). In large distributed systems like ERTMS, specification errors may have serious effects on the system development cycle as they can affect safety and interoperability. It is worth noticing that, although the primary motivation behind setting up the ERTMS standard is to ensure railway interoperability, there is still some way to go before addressing completely the interoperability challenge. Besides, the integration phase in ERTMS projects is often burdensome especially when onboard and trackside subsystems are issued by different manufacturers. This is mostly due to different interpretations that rail system providers make about specifications. This leads K. Vinck, the European ERTMS coordinator at the European Commission, to declare[3] "*our priority for the coming weeks, months and years is to ensure interoperability between ERTMS systems*". Therefore, rigorous notations as well as formal methods are highly recommended to express and validate specifications. This is explicitly highlighted as a priority for the Shift2Rail initiative,[4] which involves the major railway actors in Europe and aims to increase competitiveness of the railway sector in mid-long term.

The present study fits in this context and attempts to develop a translation of the SRS subset relative to ERTMS/ETCS modes and transitions (Subset-026/4) into a formal model; namely a conditional transition system. We also show how the obtained model can be operated, using automatic techniques, to check different types of properties. Verifying specifications relative to ETCS modes is so crucial since this allows for checking properties like deadlock-freeness and reactivity within the system behavior. Moreover, one can easily observe that, given the large number of items and conditions involved in mode transitions, checking such properties manually is not workable. Besides, providing means for automatic verification is all the more important as the SRS regularly give rise to new versions, and given that a simple modification may jeopardize several properties validated in the previous versions.

The literature review pertinent to the railway safety topic shows that, historically, classical approaches, namely probabilistic ones, are predominant in comparison to formal methods. For instance in Dobias et al. (2008), the authors use Markov chains to assess safety on a railway interlocking system. Some statistical techniques have been applied in Quaglietta (2013) for design optimization of railway infrastructure. In Jafarian and Rezvani (2012), fuzzy fault trees have been used to assess the root causes for train derailment. Timed stochastic models have been used in Vale and Lurdes (2013) to characterize track degradation over time. The same issue has been addressed in Oukhellou et al. (2008), where Bayesian networks have been used to structure sensor data for track monitoring and diagnosis purposes. In Malavasi and Ricci (2001), self-learning processes have been employed to develop simulated stochastic models for various railway subsystems and in Beugin and Marais (2012), simulation techniques are used to assess the dependability of satellite based localization in an ERTMS operating context.

Nevertheless, with the advent of modern computer-based railway systems, formal methods are gaining growing interest in railway safety studies Fantechi et al. (2012). One of the earliest successful projects involving formal methods in guided transportation systems was the Météor project (Behm et al., 1999) in which the B-Method has been advantageously extended to validate safety and functional properties on a fully-automated metro line. The B-Method (Abrial, 1996) is a

[1] ATC: Automatic Train Control – ATP: Automatic Train Protection.

[2] SRS Subsets. http://www.era.europa.eu/Core-Activities/ERTMS/Pages/Set-of-specifications-2.aspx.

[3] Plenary session of the last ERTMS conference, organized by the European Railway Agency, Lille, November 12th and 13th, 2013, http://www.era.europa.eu/conferences/CCRCC2013/Pages/Agenda.aspx.

[4] http://www.shift2rail.org/.

formal technique for software development which allows abstract modeling of system behavior. After that, by iterative refinement, a concrete model which can be easily translated into operational code is obtained. At each refinement step, a set of predefined properties are checked and validated, thus ensuring that the ultimate obtained model fulfills, by construction, the predefined desirable characteristics. The B-Method has also been used in a similar project for the Roissy VAL shuttle (Badeau and Amelo, 2005). Antoni (2012) is a report which discusses the limits of probabilistic approaches while dealing with the safety of critical railway systems, and calls for a greater involvement of formal methods in this domain. This report gives several examples, namely pertaining to ERTMS applications, which show the incapacity of used classical safety assessment methods to prevent some catastrophic errors, often with low occurrence likelihood, in such railway critical systems. In the same reference, the author also shows cost/benefit efficiency and confidence gain due to using formal methods instead of classical testing techniques in the development of critical railway automations. More recently, the Open-ETCS project has been launched in order to develop an integrated modeling, development, validation and testing framework for leveraging the cost-efficient and reliable implementation of ETCS.[5] This project brings together several railway operators, manufacturers, consultants as well as public and private laboratories and has as a main deliverable a domain-specific language (DSL), which allows for expressing the ERTMS specifications in a concise and formal representation. In Mekki et al. (2012), model-checking was exploited to validate a new functional architecture for automatic level crossings. The same technique was used in Meng et al. (2013) for verifying safety properties in a railway signal safety protocol. Model-checking, which will be introduced later on in this paper, is an automatic formal verification technique that allows for checking behavioral properties formally expressed on a dynamic model, often in the shape of a transition system (Clarke et al., 1999).

The remainder of the paper is organized as follows: Section 2 gives an overview of ERTMS, a prerequisite for the sequel. In Section 3, we first recall the motivation underlying the current research before explaining the basics of the targeted formal method. Then, the general procedure for transforming SRS natural-language specifications pertinent to ERTMS/ETCS modes management into a formal model is explained. The various translation artifacts are also illustrated through some selected typical examples. The effective exploitation of the developed model for assessing various types of properties through model-checking technique is explained in Section 4. Finally, in Section 6 some concluding remarks are given and directions for future work are outlined.

2. ERTMS: A general overview

2.1. A brief introduction to ERTMS

Development of rail transport in the last twenty years has led railway stakeholders and policy-makers in Europe to focus on the crucial issue of railway interoperability which ensures cross-border continuity. Historically, European railway signaling and control systems have been developed on a national basis without common technical and operational standards. In some cases, this disparity has even been deliberate as a consequence of strategic protectionist measures. As can be shown in Fig. 1, fifteen different railway signaling systems can be inventoried throughout Europe nowadays.

As a response to this urgent issue, the ERTMS standard has been developed to harmonize railway control and signaling systems and to improve trans-border rail services over long distances in Europe. At the same time, a new generation of technological solutions needs to be implemented so as to support interoperability. Railway operators should then become an integrated part of national and European traffic. Indeed, ERTMS seeks to offer an economic and technical solution for railway interoperability which, according to Council Directive (2008) is defined as "*the ability of a rail system to allow the safe and uninterrupted movement of trains which accomplish the required levels of performance for these lines. This ability depends on all the regulatory, technical and operational conditions which must be met in order to satisfy the essential requirements*". Based on the directives mentioned in these references, several working groups have been set up to develop various specifications to fulfill railway interoperability throughout Europe. ERTMS compromises two basic components:

(1) The ETCS (European Train Control System) is an ATP system of which the main function is to monitor train speed according to information received from the trackside and to slow down the train automatically if the maximum permitted speed is exceeded. The ETCS modules are dispatched between onboard trains and the infrastructure.
(2) The GSM-R is based on the GSM communication standard, but using different frequencies specific for railways, and integrates some advanced functions as integrity control, and some specific operating modes as for shunting and maintenance operations. The GSM-R allows for digital mutation with a competitive cost and replaces all wired communication systems (along the track) and existing analogue railway radio networks in each country, that are incompatible with each other. Today, more than 35 different railway communication systems can be recorded in Europe alone.

Note that a third module relative to traffic management is also planned for a later release. This module called ETML (European Train Management Layer) is still under development. Furthermore, it is worthwhile to mention that ERTMS has also

[5] http://www.openetcs.org.

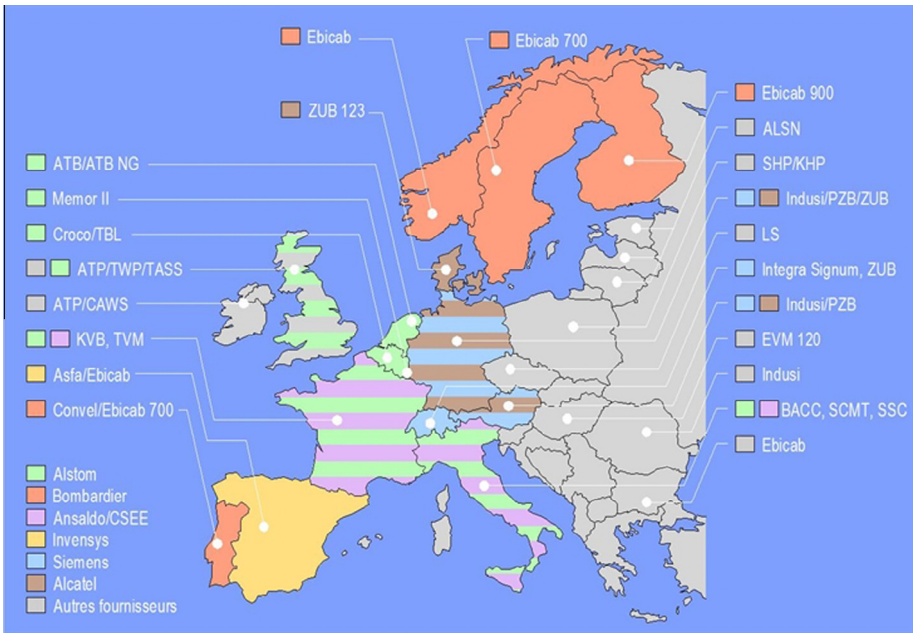

Fig. 1. Railway signaling systems in Europe – Source. http://mediarail.be/UE_ERTMS_ETCS/ERTMS_ETCS_P01.htm.

been adopted in several projects outside Europe, such as in South Korea, Taiwan, Turkey, Algeria, Argentine, and New Zeland. Moreover, an equivalent Chinese signalling and control system called CTCS (for Chinese Train Control System) and highly inspired from ERTMS, is being developed (Ning et al., 2006; Wang, 2011).

The ERTMS initiative has been backed by several related projects addressing various aspects:

- EIRENE (European Integrated railways Radio Enhanced Network), the goal of which is to write the specifications for GSM-R.
- MORANE (Mobile RAdio Networks for Europe) to test and prove GSM-R.
- HEROE (Harmonisation of European rail Rules for the Operation of ERTMS) which addresses the challenging issue of harmonizing operational rules across Europe.
- ESROG (ERTMS Safety Requirement & Objective Group) to set ERTMS tolerable hazard rates.

The ERTMS specifications include, among others, the System Requirements Specifications (SRS[2]) which define the system requirements for ETCS. These specifications define three so-called application levels of ERTMS/ETCS operation. These application levels will be detailed in the next section.

2.2. ERTMS/ETCS application levels

The ETCS has been designed with three application levels which determine different manners of interaction between the track and the trains. Depending on the application level, various onboard and trackside equipment is used. The features pertaining to the three application levels are discussed below:

- **ETCS Level 1.** is designed as an add-on or an overlaid to a conventional line already equipped with lineside signals and train detectors. In general, lineside signaling is kept. Communication between train onboard and trackside is ensured by Eurobalises located along the track adjacent to the lineside signals at required intervals and connected to the train control center. Various static or variable data can be forwarded to train onboard through eurobalises, especially movement authority according to which the Eurocab automatically determines the maximum speed of the train and the next braking point if needed, while considering the train braking characteristics and the track description data are also delivered by eurobalises. This information is displayed to the driver through the DMI (Driver Machine Interface). The EVC (European Vital Computer), which is the core module of onboard ETCS, continuously supervises the train speed. Optionally, Euroloops or radio infill units may also be used in ETCS level 1 (cf. Fig. 2).
- **ETCS Level 2.** does not require lineside signals which remain optional thus allowing for substantial savings in investment and maintenance. The movement authority is communicated directly from a Radio Block Center (RBC) to the onboard ETCS unit using the GSM-R link (cf. Fig. 3). Also, train detection by the trackside is required in level 2 and the balises are only used to transmit static messages such as location, line profile, and speed limit. Location information

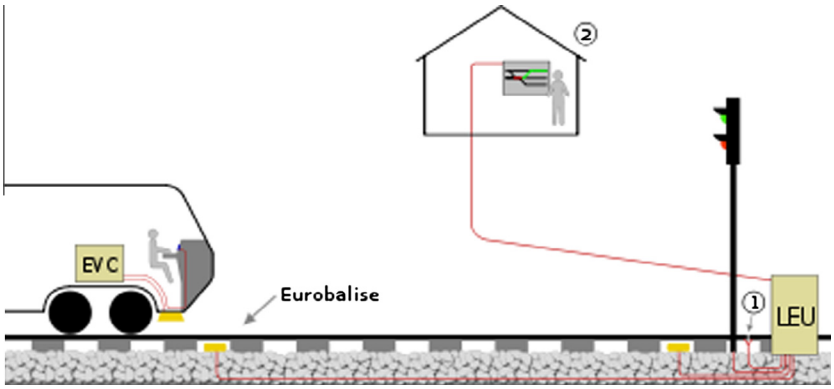

Fig. 2. ERTMS/ETCS Level 1.

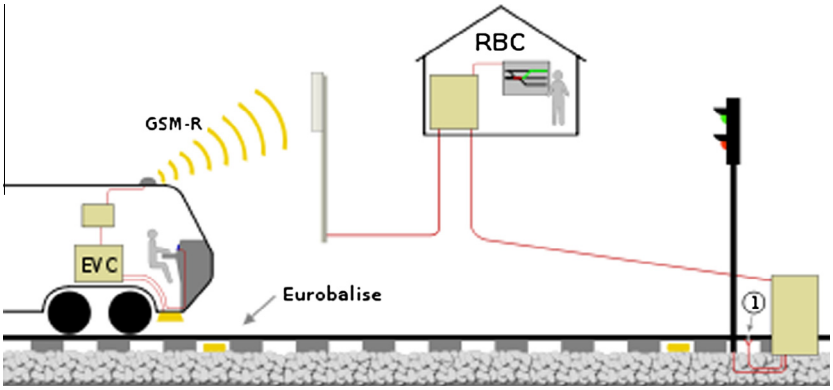

Fig. 3. ERTMS/ETCS Level 2.

allows the ETCS onboard unit to readjust the value calculated by the odometer. A continuous stream of data informs the driver of line-specific data and signals status on the route ahead, allowing an optimal determination of train speed. ERTMS Level 2 thus offers the possibility for substantial line capacity increase by enabling higher operational speed while reducing headways.

- **ETCS Level 3.** introduces the moving bloc paradigm, as opposed to levels 1 and 2 which set fixed blocs, thus furthering line capacity optimization. Moreover for ETCS level 3, train detection by the trackside is no longer required, but the train rear end location as well as the integrity status are sent to the RBC by the train itself in a continuous way. Hence, the RBC sends movement authority to the train continuously and allows the movement authority value to be computed according to the rear end position of the preceding train (cf. Fig. 4). A minimum of trackside equipment is used in level 3; Eurobalises are optional, but if used they only fulfill location readjustment. Note that this application level is still in its conceptual phase and that currently there is no ERTMS line operating under ERTMS level 3.

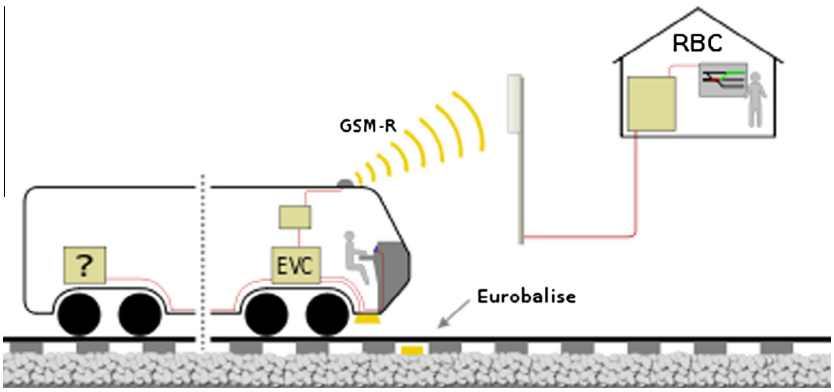

Fig. 4. ERTMS/ETCS Level 3.

The choice between these three application levels can be dictated mainly by the railways' needs, but also depends on whether to equip a new line or a line already holding a specific signaling system. Some other factors may also come into play such as the possibility to equip the line with GSM-R technology, the maximum speed allowed or the capacity upgrades. However, acquired experience through numerous ERTMS projects demonstrates that ERTMS Level 2 brings the full benefit of the system to reality, as it allows for increased capacity and significant cost savings for maintenance through the removal of lineside signals. Note that the advantage brought by ERTMS level 1 is already significant since it allows for high speed travel.

It is worth mentioning that upgrading from one ERTMS/ETCS level to another is possible, and that ERTMS is assumed to ensure a smooth migration. For instance, migrating from level 1 to level 2 mainly requires the installation of a GSM-R network, Radio Block Center(s) and some additional balises for the positioning functionality. Upgrading to level 3 allows infrastructure managers to dispense with some trackside equipment such as axle counters and track circuits.

As a concluding remark for this section, let us recall that the ERTMS/ETCS application level is not the only criterion to define the operation context under ERTMS. Actually, as will be explained later on in this paper, several ETCS operation modes have been defined according to various circumstances.

3. Modeling of the ERTMS/ETCS mode transitions

3.1. Preliminaries and objectives

The operation context of ERTMS/ETCS integrating several considerations defines what is called ERTMS/ETCS operation `mode`. Combined with the ERTMS/ETCS application level, the ERTMS/ETCS mode determines the way the system behaves in interaction with its environment. An operation mode is defined by:

(1) the ETCS levels in which it applies,
(2) a set of onboard active functions under the mode,
(3) the responsibility of the driver and of the ERTMS/ETCS onboard equipment, once the equipment is in this mode and,
(4) by a set of transitions to and from other modes.

Let us recall the motivation behind our work. Actually, the process of verifying and validating systems on the basis of natural language specifications is notoriously error-prone due to the inherent ambiguity of such a language This is a major challenge when developing critical complex systems. Generally, developers of such systems largely rely on the decomposition of problems, capitalization and return of experience accumulated through successive projects to ensure confidence in their realizations. However, these mechanisms show significant limitations when one deals with novel systems, which is quite the case with ERTMS. Another important element which makes classical approaches all the more limited in the case of ERTMS is that ERTMS specifications are regularly updated. Generally classical approaches do not indeed integrate means of non-regression guarantee and largely rely on technical expertise. Moreover, in the case of ERTMS it is essential to check specifications against errors and inconsistencies since this prevents implementation problems (in the case of inconsistency), interoperability problems (in case of different interpretations) and safety errors (when the implementation of the specifications may conduce into dangerous states). Obviously, given their complexity and the number of parameters they integrate, checking ERTMS specifications in a trustworthy way cannot be ensured based only on manual verification.

Given the numerous advantages they offer, formal methods would be an interesting remedy for this situation. Formal methods typically address model correctness and operate on a purely mathematical formalization of the model. In particular, model-checking (Clarke et al., 1999) is an automatic formal technique which allows verification of requirements expressed in a formal way, namely as temporal logic properties (Pnueli, 1977), on a given transition system model, which is assumed to depict the system behavior. This makes it possible to prevent errors at a low cost during the early design stages, by building abstract system level models. Moreover, model-checking can be processed iteratively as system description is refined progressively. Model-checking has been garnering more and more interest during the last decade both in academic and industrial fields. This gave rise to several improvements to this technique such as symbolic representation and abstraction of the state space (Burch et al., 1994), making it possible to tackle verification problems in systems of big size (10^{20} states and more). Further extensions have been developed as, for instance, the integration of likelihood and time quantification. Besides, several tools implementing model-checking techniques have been developed such as Prism for probabilistic model-checking (Kwiatkowska et al., 2011), or Kronos (Bozga et al., 1998) and Uppaal (Bengtsson et al., 1995) for timed model-checking.

One of the major issues when invoking model-checking is to build a system model on which model-checking algorithms can be processed. Indeed, the model as well as the temporal logic property to be verified are assumed to be given for model-checking. In the case of the SRS subset addressed here, first we develop a static model that gathers the whole pertinent information relative to modes and transitions that can be captured from the specifications (cf. Section 3.2). Then a formal model tractable with model-checking will be derived (cf. Section 3.3).

3.2. Modes & transitions – A static representation

One of the main concerns to deal with when translating literal specifications into a formal model is to bridge the gap between both notations. One way to do this is to take advantage of semi-formal notations, as being half-way between literal

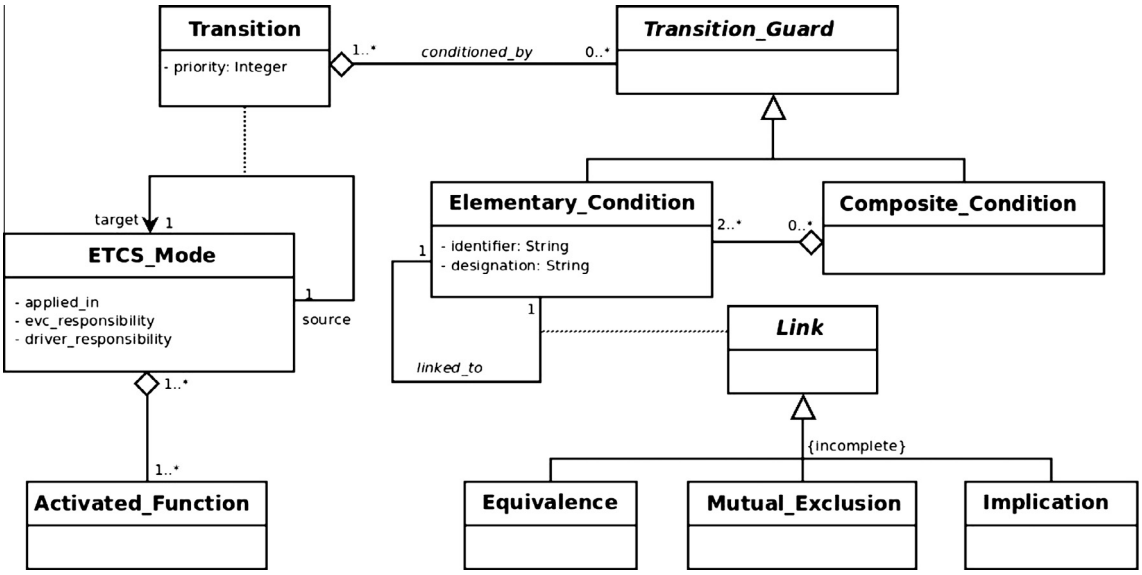

Fig. 5. Class diagram illustrating ETCS modes and transitions.

and formal notations, in order to organize the required knowledge pertinent to the concepts involved in our translation. Namely, as a first step and in order to gather and structure the various data featuring modes and transitions between modes as specified in Subset 026 – chapter 4, a UML class diagram[6] has been established as shown in Fig. 5. However, it is worth mentioning that the established class diagram is not a part of the translation chain, strictly speaking. Rather, the goal behind setting up such a diagram is twofold: (1) first to determine all the items that need to be taken into account, and thus to define the scope of the translation process; and (2) secondly and in addition to that, thanks to the UML class diagram one make it explicit all the relations between the involved concepts.

In this diagram, UML class **`ETCS_Mode`** will be instantiated into the various ERTMS/ETCS modes as defined in subset 026-chapter 4.4. This class includes the following attributes:

- `applied_in`: is an *enumeration* of the ETCS levels in which the mode applies,
- `evc_responsibility` is a *string* that gives a description of the responsibility of the ETCS onboard under the mode, and `driver_responsibility` is a *string* giving a description of the responsibility of the train driver under the mode.

One or several onboard functions must be activated according to the current ETCS mode. Moreover, a given ETCS onboard-function can be activated in one or several ETCS modes. This is depicted using an aggregation association linking the **`ETCS_Mode`** to the **`Activated_Function`** class. In addition, an ETCS mode m_1 can be linked to another ETCS mode m_2 by a relation `switch_to` which depicts the possibility to switch from the `source` mode m_1 towards the `target` mode m_2. Since several features characterize mode transitions, instead of using a simple UML association, an association-class **`Transition`** is rather set to depict mode transitions. Some conditions may be prerequisite for switching from a given `source` mode into a `target` mode. The conditions assigned to a given transition are linked by a disjunction operator (logical *OR*). In other terms, whenever at least one of these conditions is fulfilled, the transition becomes achievable, provided that there is no conflict with some other transitions (cf. priority attribute). **`Transition`** class holds the following class attribute:

- priority: an *integer* which determines the priority level of the transition. Priority applies when there is indeterminism between transitions, i.e. when more than one outgoing transition from the mode has its conditions satisfied. The less the priority value is the more priority the transition has.

Conditions are not defined as attributes for the **`Transition`** class since they may hold various data. Instead, a **`Transition_Guard`** class is defined and an aggregation relation links **`Transition`** class to **`Transition Guard`** class. Moreover, as a transition condition can be either an elementary condition or a conjunction (logical *AND*) of several elementary conditions (cf. chapter 4.6.3 of the SRS subset-026), two subclasses **`Elementary_Condition`** and **`Composite_Condition`** inheriting from the **`Transition_Guard`** class are defined, and the later is converted into an abstract class to prevent its instantiation. A *composite_condition* is the conjunction of at least two *elementary_conditions*, thus an aggregation relation between these two classes is defined. A given *elementary_condition* can be shared by several *composite_conditions* that is why an aggregation relation has been set between the corresponding classes, rather than a composition relation. In addition, by analyzing the

[6] http://www.omg.org.

involved elementary conditions, one can note that several interdependencies can be found between two elementary conditions, let us say ec_1 and ec_2, such as: mutual exclusion ($ec_1 \iff NOT\ ec_2$), logical implication ($ec_1 \Leftarrow ec_2$), logical equivalence ($ec_1 \Rightarrow ec_2$). To illustrate these relations, a reflexive association on class **`Elementary_Condition`** needs to be set. This association is implemented as an association_class **`Link`**, *specialized* into various subclasses **`Equivalence`**, **`Mutual_Exclusion`**, **`Implication`**, etc. Similarly to **`Transition_Guard`** class, **`Link`** is defined as an abstract class to prevent its instantiation. Further analysis of the condition interdependencies will be undertaken in the sequel.

The whole class diagram developed to illustrate the above features is given in Fig. 5.

Strictly speaking, several OCL (Object Constraint Language) constraints should be added to our class diagram in order to express some restrictions that have to be guaranteed. Nevertheless, since the class diagram serves here as a structuring model, the constraints that need to be fulfilled shall be taken into account within the ultimate formal model (cf. Section 3.3).

3.3. Modes & transitions – A formal dynamic model

This section is dedicated to the development of the necessary mechanisms for translating the specifications relative to ETCS mode transitions into a formal model suitable for model-checking. The last available version for the specifications relative to modes and transitions is considered here (UNISIG and ERTMS Users Group, 2012).

Let us recall here that since our ultimate objective is to "model-check" specifications, the target representation must be a transition-system-like model. Here, a labeled transition system (LTS) which is a variant of transition systems, will be developed. An LTS is a tuple $(S, Act, \rightarrow, S_0, AP, L)$ such that:

- S is a set of states (the state space),
- Act is a set of actions,
- $\rightarrow \subset S \times S$ is a transition relation. $(s, \alpha, s') \in \rightarrow$ is denoted by $s \xrightarrow{\alpha} s'$,
- S_0 is a set of initial states or initial conditions,
- AP is a set of atomic propositions,
- $L : S \rightarrow 2^{AP}$ is a labeling function.

Compared to test, model-checking offers the advantage to make an exhaustive verification. Moreover, in the case when the checked property is violated, model-checking generates a counter-example, which is actually a run that does not satisfy the property. This feature is of major importance since this offers valuable guidance for both debugging and refinement operations.

The model-checker targeted for verifying properties is *Cadence SMV* (Burch et al., 1994), a symbolic model-checker which has proven to be efficient through several academic and industrial case studies. In particular, it allows for representing systems in a modular shape, which simplifies the modeling phase. Moreover, it has a specific language for system description which remains quite intuitive while offering interesting expressing features. This language allows the use of numerous types of finite-domain variables (logical, enumerations, range of integers and n-arrays of these types) that can be defined as inputs/outputs signals or as simple variables. *SMV* language offers numerous constructs which make it possible to express behavior invariants, assignment precedence, etc. SMV also allows consideration of fairness constraints. The properties to be checked are declared with an `assert` statement at the end of the specification file. Numerous notations can be used to express theses properties, including LTL and CTL temporal logics, finite automata, embedded assertions and refinement specifications. Cadence *SMV* supports a variety of techniques for compositional verification, allowing it to be applied to large designs, with user guidance. It also includes an intuitive graphical user interface with source level debugging facilities. For more details on *SMV* syntax, the reader can refer to McMilan (1999).

In the sequel, the mechanisms adopted for translating the specifications of ERTMS/ETCS modes transition into *SMV* specifications will be detailed and illustrated.

3.3.1. Modes declaration

The ERTMS/ETCS standard defines several modes, more exactly there are 17 modes according to Subset-026/4. These modes are given in Table 1.

Under ERTMS/ETCS, the system can be in one and only one mode at any given time. Hence, an enumeration-typed variable will be defined in order to declare the ETCS modes in which the system can operate. The items of this enumeration are the various modes defined:

```
output mode:{NP,...,SL}.
```

3.3.2. Transition guards

As made explicit by the class diagram in Fig. 5, zero or several guards can be associated to mode transitions. In case of several guards being associated with a transition, at least one of these guards needs to be fulfilled in order for this transition to be enabled. A guard can be either an elementary condition or a composite condition that is defined as a conjunction of several elementary conditions. Moreover, various types of interdependency may link conditions. It is crucial to determine these interdependencies carefully and to express them in the model for verification purposes. This is not an easy task since

Table 1
ETCS operation modes.

Mode	Designation	Mode	Designation
NP	No Power	**NL**	Non-Leading
SB	Stand By	**UN**	Unfitted
PS	Passive Shunting	**TR**	Trip
SH	Shunting	**PT**	Post Trip
FS	Full Supervision	**SF**	System Failure
LS	Limited Supervision	**IS**	Isolation
SR	Staff Responsible	**SN**	National System
OS	On Sight	**RV**	Reversing
SL	Sleeping		

it needs high expertise in ERTMS/ETCS specifications. Determining the interdependencies between composite conditions is quite burdensome; therefore guards will rather be handled at the elementary condition level.

As for the interdependencies between elementary conditions, first all these conditions involved in the transition guards, as stated in Subset026-4.6.3, have been identified. One can note that the same elementary condition can be involved in several composite conditions, as depicted through the cardinalities relative to the aggregation linking `Composite_Condition` and `Elementary_Condition` classes. In order to deal with the large number of elementary conditions identified, a classification of these conditions has been undertaken according to their subject. Indeed, spreading the elementary conditions over thematic classes makes it easier to elucidate their possible interdependencies. Several classes have been identified:

- ERTMS level;
- ERTMS level switch;
- Driver action;
- Information available onboard;
- ETCS onboard status: isolation, powering, failure or interfacing with STM;
- Train speed;
- Train location;
- Desk closure status;
- DMI request;
- Procedure activation status;
- Mode profile onboard;
- Balise ID consistency.

Analysis of the elementary conditions dispatched according to the previous classes shows several features:

- Each elementary condition can be either satisfied or not: for each elementary condition, a boolean variable may be assigned to depict its status, i.e. satisfied or violated. However, defining variables must not be done before analyzing the interdependencies between the elementary conditions.
- Some elementary conditions can be further decomposed for the sake of simplification. For example conditions of the form "*ERTMS/ETCS level switches to X or Y*" can be expressed as the disjunction of two conditions "*ERTMS/ETCS level switches to X*" and "*ERTMS/ETCS level switches to Y*".
- Some elementary conditions show a relation of mutual exclusion. Moreover, two cases can be distinguished:
  - only two conditions are involved in the mutual exclusion, which is the case for example in "*a desk is open*" and "*a desk is closed*": in this case there is no reason to assign two variables for such a condition pair. Only one boolean variable suffices to depict both conditions.
  - more than two conditions are involved in a mutual exclusion relation, like for instance for conditions "*the ERTMS/ETCS level is STM*", *the ERTMS/ETCS level is 0*, "*the ERTMS/ETCS level is 1*", "*the ERTMS/ETCS level is 2*" and "*the ERTMS/ETCS level is 3*". In this case, only one enumeration-typed variable is defined to take into account all these variables. The elements of the defined enumeration shall correspond to the different values involved.

    $\texttt{ertmsLevel} : \{\texttt{STM}, 0, 1, 2, 3\};$

- Some of the elementary conditions are controlled by the environment as, for example, the conditions relative to a driver action, or those relative to the desk opening. The variables associated to these conditions will then be defined as input signals so as their respective values is not controllable. Indeed, as we deal with ERTMS specifications which define how the system must react according to certain circumstances, the stimuli issued by external actors (and transmitted to the system by sensors, through communication, etc.) must not be constrained.

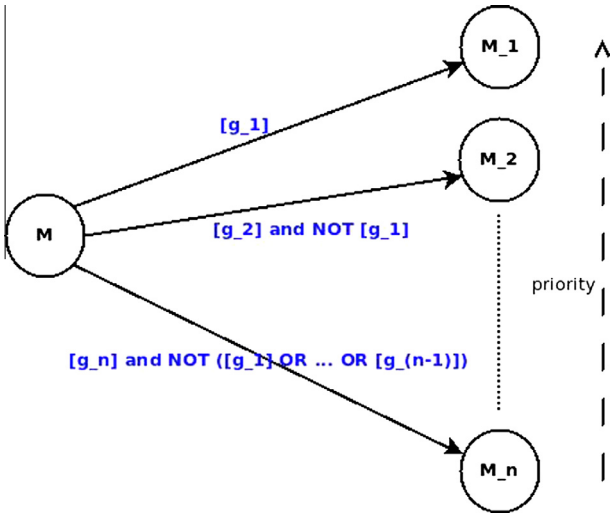

Fig. 6. Transitions with different priority levels.

- Finally, some other relations linking input variables are more subtle and can be introduced as *hypotheses*. An interesting feature of *SMV* will then be brought into play to cope with this issue. Namely, these relations can be incorporated as *fairness* constraints. A fairness requirement is a condition that restrains the checking of some given properties to the part of behavior that fulfills this requirement. In other words, only the traces fulfilling the fairness properties will be considered while checking these properties.

3.3.3. Transition priorities

One of the issues to be tackled when developing our *SMV* model depicting ERTMS/ETCS mode transitions is relative to the management of transition priorities when several transitions originate from a same mode. Indeed, from a given mode M the system can switch towards several modes M_1 to M_n, and a conflict may arise in the case when more than one transition has its guard satisfied. To resolve these potential conflicts, a priority value is assigned to each transition as depicted by the integer-typed attribute `priority` in the **`Transition`** class of the UML class diagram Fig. 5. The value of this attribute indicates the priority level of the transition. Namely, transitions with lower priority value have priority over transitions with bigger priority value. This can be illustrated by means of a finite state machine (FSM) as shown in Fig. 6. In this model, $[g_i]$ denotes the guard associated with the transition ending at mode M_i and $\forall 1 \leqslant i < j \leqslant n$, transition from M to M_i has a higher priority than the one from M to M_j.

By analyzing the *SMV* language, we pointed out an interesting construct that can be quite convenient to manage priorities over transitions. It is about the `default` construct which has the following form:

```
default {block1}
in {block2}
```

This indicates that assignments in `block1` are to be executed by default when the given signals are not assigned in the code of (`block2`). In other words, `block2` takes precedence over `block1`. Note that this construct can be used in a nested way. To illustrate this, let us consider the following example: take M_0, M_1, M_2 and M_3 as four ERTMS/ETCS modes such that:

- there is a transition from M_0 to M_1 with a priority level p_1 and a guard g_1;
- there is a transition from M_0 to M_2 with a priority level p_2 and a guard g_2;
- there is a transition from M_0 to M_3 with a priority level p_3 and a guard g_3;
- $p_1 < P_2 < p_3$.

when the system mode is m_0 (`mode` = m_0), the mode switch is specified as follows:

```
default {if (g3) next (mode) := M3;}
in default {if (g2) next (mode) := M2;}
in {if (g1) next (mode) := M1;}
```

This way, the first assignment will be processed only if the second and third assignments are not executed (neither g_2 nor g_1 are fulfilled). Similarly, the second assignment will be done only if the third one is not applied (g_1 is not satisfied).

Furthermore, in certain cases some concurrent transitions (issued from the same mode) may have the same priority level. In this case, the `case` statement is used as follows:

Let M be a given mode such that there exist, amongst others, n transitions towards n modes $M_1 \dots M_n$ with as guards $g_1 \dots g_n$ respectively, and with the same priority level p. The assignments relative to these transitions are the following:

`case` {
g_1: `next (mode)` := M_1 ;
...
g_n: `next (mode)` := M_n ;
}

The above statement must be integrated as a `default` block corresponding to priority level p.

Nevertheless, it would be interesting to check if the specifications can lead to actual indeterminism (cf. Section 4), i.e. whether the guards associated with such transitions may be simultaneously satisfied or not.

3.3.4. Active functions

Active functions onboard the train depend on the operation mode. Section 4.5.2 of Subset 026/4 defines exactly which functions are active in which operation mode and refers to the pertinent sections along the SRS. Moreover, Section 5.2 offers a classification of all the onboard functions into eight categories.

As these functions must be fulfilled by the ETCS onboard system, and since each function can be either activated or not, active functions will be depicted as boolean output signals:

`output` f_i : **`boolean`**;

Moreover, for each onboard function f_i the following statement will be declared:

$$f_i := (\mathbf{mode} = M_1) \mid \dots \mid (\mathbf{mode} = M_n);$$

where $M_1, M_2, \dots M_n$ are the operation modes in which f_i must be activated.

Note however that specifications (Subset 026/4 – Section 5.2) declare some onboard functions as "optional" in some modes. To take this case into account, two variants will be considered for each occurrence: one with the considered function as activated and the other with the function deactivated.

The general canvas of the generated *SMV* model is given in Appendix A.

4. Model-based verification

In this section, we will illustrate how the formal model relative to mode transitions can be used to check some important properties. These properties need to be expressed formally in the shape of temporal logic assertions. In the two following subsections, we briefly introduce the syntax and semantics of the temporal logics that will be used for expressing properties, namely LTL and CTL.

4.1. LTL temporal logic

LTL is linear temporal logic, which means that an LTL formula must be checked on a predefined run (path) from the system behavior. In other words, at each moment only one possible successor is considered. An LTL formula φ holds in a state s when all the possible runs that start in s satisfy φ. To formulate the LTL syntax, let us consider a transition system with a set AP of atomic propositions.

Syntax. LTL formula over set AP are formed according to the following rule:

$$\varphi ::= true | a | \varphi_1 \wedge \varphi_2 | \neg\varphi | \bigcirc \varphi | \varphi_1 \mathbf{U} \varphi_2$$

where:

- $a \in AP$ is an atomic proposition,
- $\bigcirc$ is the "Next" operator, $\bigcirc\phi$ expresses that property φ holds in the next state of the path,
- U is the "Until" operator, $\varphi_1 U \varphi_2$ expresses that property φ_2 holds somewhere in a future state on the path, and until this state is met φ_1 remains true.

Other logic modalities in LTL such as $\Diamond$ (something holds in the "Future") and $\Box$ (something is "Globally" true from now on) can be also derived. For instance: $\Box\Diamond\varphi$ means "infinitely often" and denotes that at any step j there is a step $i \geqslant j$ at which a φ-state is visited. The dual modality $\Diamond\Box\varphi$ (eventually forever) expresses that from some moment j on, only φ-states are visited.

Semantics. Let us recall that LTL formulae need to be fulfilled by all the system paths. Moreover, for a given path $\pi = s_0 s_1 \ldots$ the satisfaction relation $\models$ for a given LTL path formula is defined by:

- $\pi \models \Phi$ iff $s_0 \models \Phi$
- $\pi \models \Phi_1 \wedge \Phi_2$ iff $\pi \models \Phi_1$ and $\pi \models \Phi_2$
- $\pi \models \neg\Phi$ iff not $\pi \models \Phi$
- $\pi \models \bigcirc\Phi$ iff $\pi[1..] \models \Phi$
- $\pi \models \varphi_1 \mathbf{U} \varphi_2 \iff (\exists j \geqslant 0)(\pi[j..] \models \varphi_2 \wedge ((\forall 0 \leqslant k < j)(\pi[k..] \models \varphi_1)))$

where for a path $\pi = s_0 s_1 s_2 \cdots$ and an integer $i \geqslant 0$, $\pi[i..]$ denotes the suffix of π from index i on.

4.2. CTL temporal logic

In contrast with LTL, CTL is a branching temporal logic, which means that a CTL formula must be checked while taking into account, at each step, all the possible successor states. Phrased differently, checking is performed without defining paths a priori.

Syntax. CTL formula over set AP are formed according to the following syntax:

$$\Phi ::= true | a | \Phi_1 \wedge \Phi_2 | \neg\Phi | \exists\varphi | \quad \forall\varphi$$

where $a \in AP$ and φ is a *path* formula. CTL path formulae are formed according to the following syntax:

$$\varphi ::= \bigcirc\Phi | \Phi_1 \mathbf{U} \Phi_2$$

where φ, φ_1 and φ_2 are state formulae. Note that by convention the Greek capital letters denote CTL state formulae, whereas lowercase Greek letters denote CTL path formulae.

One can note here that two path quantifiers are used with CTL formulae:

- $\exists$ which denotes that there exists at least one path satisfying the subsequent formula,
- $\forall$ means that all the paths satisfy the subsequent formula.

Semantics. The semantics of CTL formulae can be deduced similarly as explained in the case of LTL formulae.

5. Model-based verification, illustrations

As explained earlier in this paper, the main motivation behind the present work is to develop a rigorous framework that holds all the necessary information relative to ETCS modes and transitions, which can serve as a basis for formal verification. In this section, some examples will be given to explain how the developed model can be used for checking important properties. The kinds of properties that are checked through the following examples are far from being exhaustive, and should be seen just as illustrative pointers.

5.1. Reaching The full supervision mode

Let us check the following statement: "*Provided that the ETCS equipment is not isolated, it is always possible to reach a Full supervision mode (FS)*". It is worth recalling here that while in the FS mode, the ETCS onboard has the most responsibility.

5.1.1. Formula

The above statement can be formalized in the shape of the following CTL property:

$$\forall\Box\ [\neg(\mathbf{mode} = \mathtt{IS}) \ \Rightarrow\ \exists\Diamond\ (\mathbf{mode} = \mathtt{FS})]$$

This formula can be interpreted as follows: whatever the run of the system and all along this run ($\forall\Box$), or in other words at each state of the system, if the current mode is not IS, i.e. if the ETCS onboard equipment is not isolated, then there exists at least a path ($\exists$) in which a state where the mode is FS can be eventually ($\Diamond$) reached. This formula can be seen as a "reachability" property.

5.1.2. Checking results

SMV takes a little time to check this property and shows that the stated requirement is met.

5.2. Leaving train trip mode

Let us now express the requirement stating that "*the ETCS onboard never leaves the TRIP mode (*TR*) before the driver has acknowledged the train trip and the train is stopped*".

5.2.1. Formula

The LTL formula expressing this property is as follows:

$$\Box\,[\mathbf{mode} = \mathbf{TR} \Rightarrow \Box(\mathbf{mode} = \mathbf{TR}) \vee (\mathbf{mode} = \mathbf{TR}\ \mathbf{U}\ (\mathbf{ack\_train\_trip} \wedge \mathbf{train\_stopped}))]$$

"⇒" is the `imply` operator which is not *native* in *LTL*, but is supported as a shortcut in most temporal logics and model-checking tools, given that $A \Rightarrow B$ is equivalent to $\neg B \vee A$.

The above formula can be read as follows: Globally (□), i.e. at each state, once in the `TRIP` mode (`mode = TR`), the ETCS onboard remains indefinitely (□ `mode = TR`) in this mode, or remains in this mode (`mode = TR`) until (`U`) both of the following conditions are fulfilled: (1) the driver has acknowledged the train trip and the train is stopped ((ack_train_trip ∧ train_-stopped)). Let us recall that an *LTL* formula must hold at each path in the system behavior; consequently the formula after the □ operator must be fulfilled indefinitely and the whole formula can be seen as a "safety" property.

5.2.2. Checking results

The obtained verification outcome shows that the requirement is not fulfilled. By analyzing the counter-example generated by *SMV*, one can notice that there is a possibility that the `TR` mode can be left without the two indicated conditions having been fulfilled. This happens when a failure affecting safety is detected. In this case, the ETCS onboard directly switches to the `SF` (System Failure) mode. It is worthwhile to recall that the model-checking process stops as soon as a counter-example violating the property under verification is encountered. Let us therefore modify the requirement in such a way as to rule out this case. The corresponding formula is then:

$$\Box\,[(\mathbf{mode} = \mathbf{TR}) \Rightarrow \Box(\mathbf{mode} = \mathbf{TR}) \vee ((\mathbf{mode} = \mathbf{TR})\,U\,((\mathbf{ack\_train\_trip} \wedge \mathbf{train\_stopped}) \vee \mathbf{system\_failure}))]$$

Checking this property shows that it is violated as well and the analysis of the trace generated by *SMV* shows that when in `TR` mode, it is possible to leave this mode even without a fault affecting safety has been detected. This occurs if the ETCS onboard equipment is isolated, in which case the system switches to the `IS` mode (isolation). Let us move the analysis further while setting aside this case as well. The new formula is as follows:

$$\Box\,[(\mathbf{mode} = \mathbf{TR}) \Rightarrow \Box(\mathbf{mode} = \mathbf{TR}) \vee ((\mathbf{mode} = \mathbf{TR})\,U\,((\mathbf{ack\_train\_trip} \wedge \mathbf{train\_stopped}) \vee \mathbf{system\_failure}\ \mathbf{system\_isolated}))]$$

The generated results show that the property is not satisfied and thanks to the counter-example generated by *SMV*, one can see that while in the trip mode, in case the ETCS onboard is not powered anymore, the system switches to the NP mode. This case will also be eliminated then and the formula becomes:

$$\Box\,[(\mathbf{mode} = \mathbf{TR}) \Rightarrow;\Box(\mathbf{mode} = \mathbf{TR}) \vee ((\mathbf{mode} = \mathbf{TR})\,U\,((\mathbf{ack\_train\_trip} \wedge \mathbf{train\_stopped}) \vee \mathbf{system\_failure} \vee \mathbf{system\_isolated} \vee \neg\mathbf{powered}))]$$

The outcome obtained from *SMV* shows that the property is in fact satisfied and the corresponding requirement can thus be literally stated as: "*the ETCS onboard never leaves the TRIP mode (`TR`) before the driver has acknowledged the train trip and the train is stopped, provided that there is no power interruption, the ETCS onboard is not isolated and no safety failure is detected*". The three excluded cases can thus be seen as the authorized exceptions for the considered safety requirement.

Note finally that the generated *SMV* model holds several hundreds of variables (boolean, enumeration, tables) and of assignments including various constructs. Moreover, the above properties have been checked while considering tens of fairness constraints introduced as LTL properties, which implement realistic considerations on the analyzed behavior. Despite this, *SMV* takes less than one second to check each of the above properties.[7] This shows the effectiveness of using symbolic representation for model-checking (Burch et al., 1994).

6. Discussion and future directions

Railway interoperability is a major challenge for the European Union toward an increasing economic integration, and needs knocking down the borders created by the disparity of the current European railway networks. The ERTMS standard can be seen as the response of the EC to this concern. As for any complex system, ERTMS specifications are generated through successive versions, still in the process of evolving. Due to the railway system criticality, these specifications need to be rigorously checked especially for the parts pertaining to safety issues. However, checking complex rough specifications expressed in natural language remains a burdensome task and a refinement process is often needed before the establishment of a rigorous checking basis.

In the present work, first a generic UML model which gathers and structures all the information pertinent to ETCS modes and transitions is established. Then, the ingredients necessary for developing formal specifications from the subset pertinent to ETCS modes and transitions have been developed. The formal specifications thus obtained allow us to check various types of properties. It is worthwhile to emphasize that, although the SMV syntax has been targeted here for generating a formal

[7] Verification has been performed on a PC Intel with a dual-core processor having a 3.06 GHz clock.

model, translating the developed model into other notations such as automata and state machines, can be performed in a straightforward way. This allows for using different checking tools and developing further refinement models.

On the other hand, given that new versions of SRS specifications are regularly generated, and as a slight modification made on these specifications may break several properties at once, all the properties that must be fulfilled need to be checked "from scratch" on each new version. Nevertheless, since the changes made in each new version are rigorously traced and thanks to the mechanizable translation developed in the present study, developing a formal model respectively to a given new specifications' version can be easily obtained from the model of the previous specifications' version just by mapping the corresponding changes into this model. The study lead in this paper showed that formal methods can offer valuable support for V&V of some ERTMS/ETCS SRS subsets. Nevertheless, specification analysis shall be backed by high expertise and a special care must be taken for model generation which potentially needs several iterations before obtaining a trustworthy translation.

It is worth noticing that the subset handled in this study deals with requirements which remain quite convenient for using formal verification techniques. Some other SRS may require longer refinement processes in order to obtain a corresponding formalized model (Peres et al., 2012). As future directions, we aim to extend our formalizing technique to cope with further ERTMS/ETCS SRS subsets. In addition, the formal models we have developed can serve as a basis for conformance test generation. Such tests allow for ETCS equipment qualification, but can also be used by ETCS equipment manufacturers for V&V purposes. This can be actually performed by elaborating abstraction models for the implementation solutions and checking them against the developed formal model. Another interesting perspective of the present study, would be the integration of the RailML standard (Nash, 2004), which is an initiative to define a standard open language that allows for the interconnection of various railway IT applications, such as rail operation simulators and traffic management centers. This indeed would help the spreading of such verification techniques through the railway community.

Acknowledgments

The present research work has been partially supported by the International Campus on Safety and Intermodality in Transportation, the Nord-Pas-de-Calais Region, the European Community, the Regional Delegation for Research and Technology, the Ministry of Higher Education and Research, and the National Center for Scientific Research. The author gratefully acknowledges the support of these institutions.

The author would like to also thank his colleague E. Lemaire, a research engineer at IFSTTAR, for the interesting technical discussions they have had about the ERTMS/ETCS specifications and for his expertise in this domain.

Appendix A. General form of the SMV model

```
/* Declaring constants */
#define CONDITIONS_NUMBER xx
#define NUMBER_ONBOARD_FUNCTIONS yy
...
module main (list_of_inputs)
/* —Inputs declaration— */
/* 1- ERTMS level */
...
/* 2- ERTMS level switch */
...
/* 3- Driver action */
...
/* 4- Information available onboard */
...
/* 5- onboard status: isolation/powering/failure/STM */
...
/* 6- Train speed */
...
/* 7- Train location; */
...
/* 8- Desk closure status */
...
/* 9- DMI request */
...
/* 10- Procedure activation status */
...
```

(*continued on next page*)

```
/* 11- Mode profile onboard */
...
/* 12- Balise ID consistency */
...
/* –Output declaration– */
...
/* Declaration of onboard functions */
output functions: array
1..NUMBER_ONBOARD_FUNCTIONS of boolean;
/* modes declaration */
output mode:{NP,...,RV};
/* –Variables declaration– */
/* Declaration of transition conditions */
conditions: array 1..CONDITIONS_NUMBER of boolean;
/* –Initialization– */
init (mode) := NP;
...
/* –Defining transition conditions– */
conditions[1] := ...;
...
conditions[CONDITIONS_NUMBER] := ...;
/* –Management of onboard functions– */
functions[1] := (mode = x) | (mode = y) | ...;
...
functions[NUMBER_ONBOARD_FUNCTIONS] := ...;
/* –Encoding transition guards– */
case
{
        mode = NP:
        {
        default {if (conditions[yy]) next (mode) := YY;}
        in default {if (conditions[xx]) next (mode) := UU;}
        in {if (conditions[zz]) next (mode) := ZZ;}
        }
        mode = SB:
        {
        ...
        }
        ...
}
/* –Declaring fairness constraints– */
fair_1: assert ...;
...
fair_n: assert ...;
/* – Properties to be checked– */
prop_1: assert ...;
...
prop_m: assert ...;
/* –Checking prop_1 to prop_m while considering fairness constraints– */
using fair_1,..., fair_n prove prop_1,..., prop_m;
assume fair_1,..., fair_n;
```

References

Abrial, J.R., 1996. The B Book. Assigning Programs to Meanings. Cambridge University Press, Cambridge, ISBN: 0521496195.

Antoni, M., 2012. L'approche probabiliste des risques ne met pas l'abri des catastrophes. A quand la gnralisation des mthodes formelles dans le ferroviaire? Technical Report, 2012.

Badeau, Frédéric, Amelo, Arnaud, 2005. Using B as a High Level Programming Language in an Industrial Project: Roissy VAL, ZB 2005: Formal Specification and Development in Z and B, LNCS, No. 3455, May 7, 2005, pp 334–354.

Behm, Patrick, Benoit, Paul, Faivre, Alain, Meynadier, Jean-Marc, 1999. Météor: A Successful Application of B in a Large Project, in the Proceedings of FM 99: World Congress on Formal Methods. Lecture Notes in Computer Science. Springer-Verlag.
Bengtsson, Johan, Larsen, Kim G., Larsson, Fredrik, Pettersson, Paul, Yi, Wang, 1995. Uppaal – a tool suite for automatic verification of real-time systems. In: Proceedings of the 4th DIMACS Workshop on Verification and Control of Hybrid Systems, New Brunswick, New Jersey, 22–24 October, 1995.
Beugin, Julie, Marais, Juliette, 2012. Simulation-based evaluation of dependability and safety properties of satellite technologies for railway localization. Transp. Res. Part C: Emerg. Technol. 22, 42–57.
Bozga, Marius., Daws, Conrado., Maler, Oded., Olivero, Alfredo., Tripakis, Stavros., Yovine, Sergio., 1998. Kronos: a model-checking tool for real-time systems. In: Formal Techniques in Real-Time and Fault-Tolerant Systems, Lecture Notes in Computer Science, vol. 1486, 1998, pp 298–302.
Burch, J.R., Clarke, E.M., Long, D.E., McMillan, K.L., Dill, D.L., 1994. Symbolic model checking for sequential circuit verification. IEEE Trans. CAD Integr. Circ. Syst. 13 (14), 401–424.
Clarke, Edmund M., Grumberg, Orna, Peled, Doron A., 1999. Model Checking. The MIT Press.
Commission Decision of 6 November 2012, Amending Decision 2012/88/EU on the Technical Specifications for Interoperability Relating to The Control-Command and Signalling Subsystems of the Trans-European Rail System, Official Journal of the European Union, 10 November 2012.
Council Directive 96/48/EC of 23 July 1996 on the Interoperability of the Trans-European High-Speed Rail System, Official Journal of the European Union L 235, 17/09/1996, 1996, pp. 6–24.
Council Directive 2008/57/EC of 17 June 2008 on the Interoperability of the Rail System within the Community, Adopted by The European Parliament in the 17th of June 2008, Official Journal of the European Union L 191/1, 18/07/2008, 2008, pp. 6–24.
Dobias, R., Konarski, J., Kubatova, H., 2008. Dependability evaluation of real railway interlocking device. In: 11th EUROMICRO Conference on Digital System Design Architectures, Methods and Tools (DSD'08), 2008, pp. 228–233.
ERTMS/ETCS 2007. Functional System Requirements Specification (FRS) V5.0, Reference ERA/ERTMS/003204, Dated 21 June 2007.
Fantechi, Alessandro, Fokkink, Wan, Morzenti, Angelo, 2012. Some trends in formal methods applications to railway signaling. In: Formal Methods for Industrial Critical Systems: A Survey of Applications, first ed. Wiley-IEEE Press, pp. 61–84, ISBN: 9781118459898.
Ghazel, M., Mekki, A., 2010. Assisting specification and consistency-check of temporal requirements for critical systems. In: Proceedings of the 2010 International Conference on Software Engineering Research & Practice, SERP 2010. CSREA Press, Las Vegas, Nevada, USA, pp. 605–611.
Jafarian, E., Rezvani, M.A., 2012. Application of fuzzy fault tree analysis for evaluation of railway safety risks: an evaluation of root causes for passenger train derailment. J. Rail Rapid Transit 226 (1), 14–25.
Kwiatkowska, Marta, Norman, Gethin, Parker, David, 2011. PRISM 4.0: verification of probabilistic real-time systems. In: Proc. 23rd International Conference on Computer Aided Verification (CAV11). LNCS, vol. 6806. Springer, pp. 585–591.
Malavasi, Gabriele, Ricci, Stefano, 2001. Simulation of stochastic elements in railway systems using self-learning processes. Eur. J. Oper. Res. 131 (2), 262–272.
McMilan, K.L., 1999. Getting Started with SMV, Technical Report, Cadence Berkeley Labs, March 23, 1999.
Mekki, A., Ghazel, M., Toguyéni, A., 2012. Validation of a new functional design of automatic. protection systems at level crossings with model-checking. IEEE Trans. Intell. Transp. Syst. 13 (2), 714–723.
Meng, M., Zhongwei, X., Xi, W., Yongbing, W., 2013. Model checking-based safety verification for railway signal safety protocol. J. Comput. Appl. Technol. 46 (3), 195–202.
Nash, Andrew, 2004. RailML a standard data interface for railroad applications. In: Proceedings of Computers in Railways (Comprail IX). Wit Press, Dresden, pp. 233–240,.
Ning, Bin, Tang, Tao, Gao, Ziyou, Yan, Fei, Wang, Fei-Yue, Zeng, Daniel, 2006. Intelligent railway systems in China. IEEE Intell. Syst. 21 (5), 80–83.
Oukhellou, L., Cme, E., Bouillaut, L., Aknin, P., 2008. Combined use of sensor data and structural knowledge processed by Bayesian network: application to a railway diagnosis aid scheme. Transp. Res. Part C: Emerg. Technol. 16 (6), 755–767.
Peres, Florent, Yang, Jing, Ghazel, Mohamed, 2012. A formal framework for the formalization of informal requirements. Int. J. Soft Comput. Softw. Eng. (JSCSE) 2 (8), 14–27.
Pnueli, Amir., 1977. The temporal logic of programs. In: The Proceedings of the 18th Annual Symposium on Foundations of Computer Science, Providence, Rhode Island, USA. IEEE, 1977, pp. 46–57.
Quaglietta, Egidio, 2013. Supporting the design of railway systems by means of a Sobol variance-based sensitivity analysis. Transp. Res. Part C: Emerg. Technol. 34, 38–54.
UNISIG, ERTMS Users Group, 2012. Subset-026 of the System Requirements Specification (SRS), Version 3.3.0, March 07th 2012.
Vale, C., Lurdes, S.M., 2013. Stochastic model for the geometrical rail track degradation process in the Portuguese railway Northern Line. Reliab. Eng. Syst. Safety (RESS) 116, 91–98.
Wang, Jun-Feng, 2011. CTCS-2I: new train control system suitable for trains with speeds up to 350 km/h. J. Transp. Eng. 137 (5), 327–332.