A generation method of test scenarios based on models: application to the ERTMS/ETCS system

S. Jabri[1], E. M. El Kouris[1], E. Lemaire[1] & T. Bourdeaud'huy[2]
[1]*National Institute for Transport and Safety Research, Lille, France*
[2]*Ecole Centrale de Lille, France*

Abstract

The railroads must be able to offer high-quality services for both the high-speed trains and the conventional ones. So, these objectives imply a search for a harmonization of the European rail network. For this purpose, the European Union set up a European management system of the rail traffic – the ERTMS system (European Rail Traffic Management System) – to ensure, in full safety, trains circulations on different European networks. It is composed of the control-command system, named ETCS (European Train System Control), and the GSM-R, the new radio system for voice and data communication. As the full deployment of this system will be long and expensive, evolutions will be necessary and will raise other technological challenges. This paper presents methods, models, and tools dedicated to the generation of tests scenarios for the validation of ERTMS components based on functional requirements.

Keywords: ERTMS system, UML, Petri Net, model transformation.

1 Introduction

On international trains, onboard equipment for the various national control command systems of the corresponding geographical area generally must be installed, in parallel. This is becoming more and more costly due to the increasing sophistication and expense of equipment. So, in order to remove these obstacles through the European rail network, the European Commission encouraged the development of a signalling and management system common to all member states – the ERTMS system (European Rail Traffic Management

WIT Transactions on the Built Environment, Vol 108, © 2009 WIT Press
www.witpress.com, ISSN 1743-3509 (on-line)
doi:10.2495/SAFE090221

System). This work aims to facilitate the interoperability through the mutual recognition of the ERTMS components between the member states by proposing test scenarios generation. The objective of these tests is to check if ERTMS components comply with ERTMS specifications. A description of the ERTMS system, as the scientific context of this study, follows this introduction. Next, a UML (Unified Modelling Language) modelling of ERTMS/ETCS specifications is presented. Then, a model transformation technique from UML to Petri Net formalism is proposed in order to generate test scenarios. These conformity tests will be considered as a checking technique of the system behaviours compared to those described in the specifications.

2 ERTMS system

The ERTMS system aims at curing the fragmentation of the European rail network, identified as a major obstacle to the development of the international rail traffic. In fact, the system principle is to standardize the several signalling systems currently coexisting in Europe and to produce an economic and technical solution to the railway interoperability [1, 2]. From a functional point of view, interoperability is characterized by four major points: At the borders, the train should not change locomotives, the train should not stop, it should not have change of control agent, and the driver should not carry out control actions other than ERTMS standardized ones [3].

2.1 ERTMS components

The ERTMS system has two basic components:

- ETCS, the European Train Control System, which not only transmits permitted speed and movement information to the driver of the train, but monitors constantly the driver's compliance with these instructions.
- GSM-R (GSM for Railways), the radio system used for the information transmission between the track and the train. It is based on the standard GSM but using different frequencies specific to rail.

2.2 ERTMS application levels

The ERTMS system has to replace many railway signaling systems that currently exist in Europe. To deal with the very different configurations in the signaling equipments in the member states, ETCS has been conceived with three so-called applications levels which are a way to express the possible relationships between track and train [4]. They are summarized in the following paragraphs.

2.2.1 Application level 1

In application level 1, the track transmits to the train information allowing it to calculate constantly its maximum authorized speed. This information is transmitted to the train by means of Eurobalises placed along the track and connected to the existing signaling system [4].

2.2.2 Application level 2

In application level 2, the ETCS uses a continuous radio communication, the GSM-R, to exchange information between the track and the train. The interlocking transfers the status of trains' routes to the RBC (Radio Block Center) which, in turn, calculates the correct movement.

2.2.3 Application level 3

As for Level 2, RBC uses the GSM-R transmission between the track and the trains, but at level 3, the track receives the train location and the train integrity from trains.

2.3 ERTMS/ETCS architecture for application level 2

Required for any new line in Europe, this paper focuses only on application level 2 of ERTMS. For this application level, as any other level, the ETCS is distributed partly trackside and partly on-board the train. These two sides are more detailed in the following paragraphs.

2.3.1 The on-board sub-system

The train must have a movement authority corresponding to the track distance reserved to it. As the train progresses, this authorization is updated. Based on this concept, the EVC (European Vital Computer) constructs different curves, including the maximum authorized speed curve, the alert curve, the service brake intervention curve, and the emergency intervention curve. It controls the train's speed relative to those curves taking into account the position of the train, its current speed and its braking capability [4].

2.3.2 The track-side sub-system

Regarding the application level 2, the trackside sub-system can be composed of the RBC and the balises. The set of balises constitutes geographical references. The RBC is the computer-based system that delivers all train movement authorities. It uses safety data from the track, such as train positions, to send messages to the Euroradio equipment which, in turn, will transmit the movement authorities by radio to all trains on the line [4].

2.4 ERTMS specifications

The Functional Requirements Specifications (FRS) [5] and the System Requirements Specifications (SRS) [6] constitute the system level ERTMS specifications. They are defined in a text format. The FRS identifies the functions required for technical interoperability. The SRS define the system requirements for the European Train Control System of ERTMS. They are the translation of the mandatory functional requirements defined in the FRS.

2.5 Main issue

As being a real-time distributed system, ERTMS can be considered an industrial complex system. It also uses a great number of actors and, as it is a railway

signaling system, it must comply with very strict safety constraints. It is then necessary to check and validate ERTMS system in order to give confidence to all parts [7]. These checking activities suggest various techniques such as: static analysis, model checking, and conformity test. Conformity tests verify if an implementation satisfies the behaviour described in the specification. This method detects the errors of implementation by performing sequences of actions; in order to determine whether system behaviours are those initially specified. This type of test was developed for the validation of communication protocols [8]. Conformity tests have three steps: (1) *the derivation of tests scenarios* which aims to *generate abstract test scenarios* on the basis of the protocol specification, (2) allows implementation tests scenarios in order to make them executables for a particular implementation, and (3) *apply the implemented tests scenarios* in order to determine the conformity verdict.

The problem is to determine how to use ERTMS specifications in order to produce test scenarios. Regarding ERTMS specifications, FRS and SRS contain functional requirements at system level, which place the core of this study in the functional testing field. However, the structure of the specifications (inputs, outputs, requirements) is not formal enough and therefore not well appropriate to the preparation of test scenarios. The objective of this work consists in producing "standard bricks" which will describe and specify each requirement. Then, a transformation process allows obtaining formalized models which will be used in this work. This formalization requires analyzing, classifying, and structuring of the specifications. They are described hereafter.

3 ERTMS specifications modeling

The ERTMS specifications constitute the basis of this research. Indeed, their analysis is considered to be a first step of the proposed method. As already explained in paragraph 2.4, the description of these specifications is not formal, hence the need to build formal and standard models. At a first time, SADT (Structured Analysis and Design Technique) method was proposed to organize the specifications because it is a functional analysis method [9] and commonly used in railway sectors. This method provides a static, clear, and precise architecture of the ERTMS specifications. However, carrying out checks of the ERTMS/ ETCS system appears to be difficult using SADT models. Indeed, the SRS are evolving while SADT models are static and cannot be easily reused. So, a migration from this functional modeling to an object oriented one is proposed in order to apprehend the dynamic behavior of the system and to allow the re-use of models. The selected object formalism is UML (Unified Modeling Language).

3.1 UML language

The Object oriented analysis (OOA) views the world as objects with data structures and behaviors. The idea that a system can be viewed as a population of interacting objects, each of which is an atomic bundle of data and functionalities, is the foundation of object technology.

UML is a standardized object modeling language. It is a semi-formal model and a powerful mean of communication. It is used to produce a systems model through diagrams. Each diagram defines and graphically visualizes a model representing one view or aspect of the modeled system. For instance, the classes diagram expresses the static structure of a system in terms of classes and relations between classes. The object diagrams are more concrete than class diagrams, and are often used to provide examples or act as test cases for the class diagrams [10]. Sometimes, there is information, such as system functional constraints, which cannot be presented in diagrams. In this case, OCL (Object Constraint Language) can be used to enrich the concerned diagrams [3].

The main objective of this research is the generation of test scenarios to check the behavior of an on-board ERTMS component (EVC) compared to specifications. Then, it is necessary to model the dynamic view of ERTMS system by using different UML diagrams [11].

3.2 UML modeling of ERTMS dynamic behavior

ERTMS system is composed of two parts, on track and on board equipment. It is necessary to consider two system points of view in order to identify external actors. In the first point of view, the considered system is the on-board one, so actors are defined as driver, RBC, and GSM-R. In the second point of view, the considered system is the trackside one, so actors are defined as trains (EVCs), traffic manager, and interlocking. This allows defining system use cases which are a sequence of actions carried out by the system and producing an observable result to a particular actor. The first point of view was chosen in order to comply with the research aim and to check EVC component. Therefore, a complete use case must be modeled in order to use it for verification and validation. The procedure Start of Mission (SOM) allows the start of a train. It is considered a use case for the on-board system, fig.1.

The use case implemented in fig.1 needs to be detailed in a State machine diagram in order to understand the dynamic behavior of the EVC (fig.2), for example. The UML state machine diagram determines the various states that an object may be in and the transitions between those states. In fact, a state represents a stage in the behavior pattern of an object, and like UML activity diagrams, it is possible to have initial and final states. An initial state is the one that an object is in when it is first created, whereas a final state is one in which no transitions lead out. A transition is a progression from one state to another and can be triggered by an event and conditioned by a constraint. The State machine diagram presented in this section (fig.2) describes the behavior of the EVC during the identification of SOM data. However, the lack of a formal and well-defined semantic leaves the notation open to many interpretations [12]. This research aims to check and to validate an ERTMS component. So it is not possible to apply, directly, mathematical techniques on the State machine diagram for system validation. UML language is used, in this paper, in order to model ERTMS specifications and to document the steps of test scenario preparation. To reach the study goal, a transformation of UML State machine diagram to Petri Nets model is proposed [12].

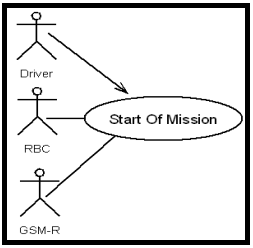

Figure 1: Use case diagram of the SOM procedure.

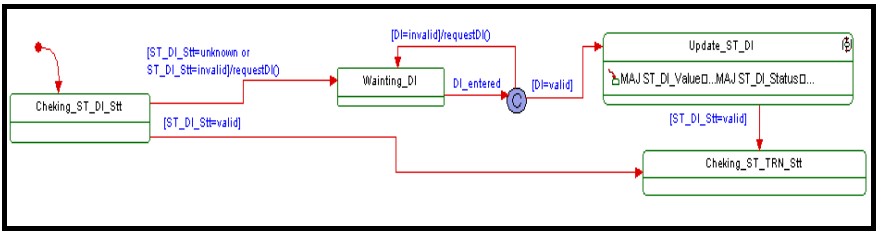

Figure 2: A UML State machine part of identification SOM data (Driver ID).

4 Transformation of UML State machine to Petri net

4.1 Petri net

Petri nets are a promising tool for describing systems that are characterized as being concurrent, asynchronous, distributed, parallel, non-deterministic, and/or stochastic. As a graphical tool, it is used like a means of communication and it is similar to state machine diagram. It consists of places, transitions, and arcs that connect them. Input arcs connect places with transitions, while output arcs start at a transition and end at a place. In addition, tokens are used in places in order to simulate the dynamic and concurrent activities of systems. The transitions model the activities which can occur, thus changing the state of the system. They are only allowed to fire if they are enabled, which means that all the preconditions for the activity must be fulfilled. As a mathematical tool, it is possible to set up state equations, algebraic equations, and other mathematical models describing the behavior of systems [13].

4.2 Transformation technique between UML State machine and predicate transition Petri net

In the presented work, the idea consists in modeling the developed UML use cases with Petri nets as they present a powerful mathematical and graphical tool to model the systems dynamic behavior [14]. The Petri net model is built on the basis of the UML State machine. It offers the possibility to check all existing

paths within the states graph, where one path constitutes one scenario to be tested [15].

This translation between two models need some rules to match the elements modeled in UML to other elements, which compose the Petri net. To formalize this transformation [16], the first step is that the structure, so called metamodel, of both UML and Petri Net models shall be known. The UML metamodel of the State machine diagram is available on the reference [10], and figure 3 shows the metamodel of Predicate Transition Net [17]. Based on these metamodels, the second step consists in defining some rules to match the elements modeled in UML to elements composing the Petri Net.

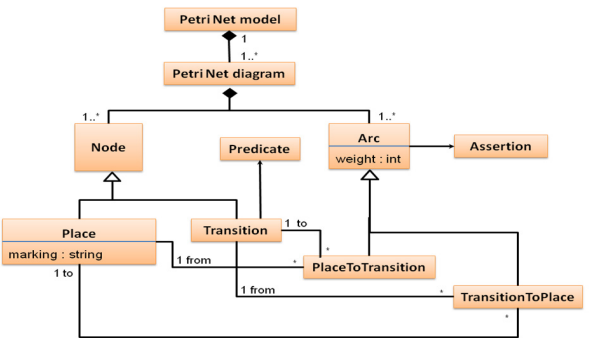

Figure 3: A simple metamodel for predicate transition net.

A State machine diagram [10] contains states (simple and composite) and transitions (events, guards, and actions). A state has several parts, namely **Name** (textual string for identification), **Entry action** (action executed on entering the state), **Do action** (action executed in the state), and **Exit action (**action executed on exiting the state). A transition has five parts, namely **Source state** (state affected by the transition), **Event Trigger** (event whose reception makes the transition fireable), **Guard Condition** (Boolean expression that is evaluated before a transition fires, the transition can fire only if the condition evaluates to true), **Action** (executable atomic computation), and **Target state** (state that becomes active after the completion of the transition).

Predicate Transition net consists of places, transitions, arcs, predicates, and assertions. **Transitions** are active components. They model activities which can occur, thus changing the state of the system. Transitions are only allowed to fire if they are enabled, which means that all the preconditions for the activity have been fulfilled. **Places** are placeholders for **tokens**. The current state of the system being modelled is called marking. A **predicate** is associated to a transition and allows expressing a condition guard. An **assertion** allows performing an action and is associated to an arc between a transition and a place. The following figures summarize the mapping between the different elements composing the state machine and those composing the Predicate Transition net.

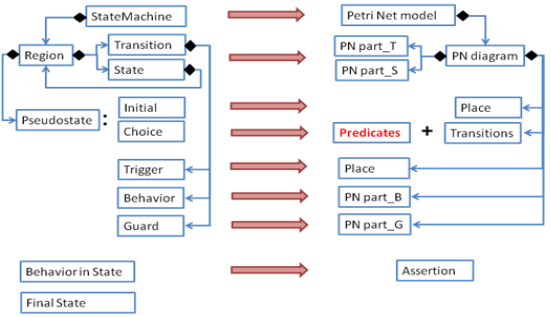

Figure 4: Mapping between State machine and Petri net.

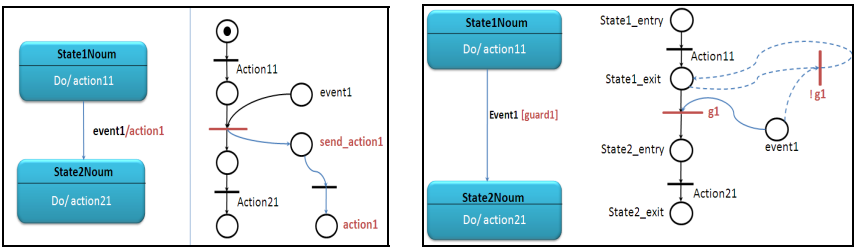

Figure 5: Transformation rules between State machine and Petri net.

5 Tests scenarios generation

The tests generated on the Petri Nets models will be used to measure the cover and also to detect non-covers. The idea is to consider the test as a scenario that is a firing sequence. So, in order to generate tests on the Petri Net model, there are two possibilities: building the marking graph (in this case, browsing the graph will be considered like a firing sequence) or generating directly the most relevant firing sequences (in this case, combinatorial optimization approaches will be used, such as Constraint Programming [18] or Linear Programming [19]).

References

[1] Commission of the European Communities, Council Directive 2001/16/EC of the European Parliament and of the Council on the interoperability of the trans-European conventional rail system, *Official Journal of the European Union*, (2001).

[2] European Commission. ERTMS – Delivering flexible and reliable rail traffic, *DG TREN*, 16p, (2006).

[3] Jabri, S. & Lemaire, E., Modeling of the European railway system for automatic checking. *15th International Symposium EURNEX- ZEL 2007: Towards more competitive European rail system*, (2007).

[4] website: http://www.ertms.com/
[5] UNISIG, ERTMS Users Group, FRS V4.29, Functional System Requirements Specification, FRS, (1999).
[6] UNISIG, ERTMS Users Group, Subset026, System Requirements Specification, SRS, (2002).
[7] El Koursi, E.M. & Kampmann, B., Qualitative and quantitative safety assessment of ERTMS Operating rules, *Comprail*, pp. 671-680, (2002).
[8] Samuel P., Mall R. & Kanth P., Automatic test case generation from UML communication diagrams, *Information and Software Technology*, Vol. 49, pp. 158-171, (2007).
[9] Cummins R., Functional Analysis, *Journal of Philosophy*, Vol. 72, n° 20, pp. 741-765, (1975).
[10] Object Management Group, Unified Modelling Language: superstructure, version 2.0, http://www.omg.org/docs/formal/05-07-04.pdf, (August 2005).
[11] Knollmann V. & Lemmer K., Combined UML-Based system and test models as back-bone for the development process, *FORMS/FORMAT 2007*, pp. 77-87, (2007).
[12] Lopez-Garo, J.P., Merseguer J. & Campos J., From UML activity diagrams to stochastic Petri Net models: Application to Software Performance Engineering, *Proceedings of the 3rd international workshop on Software and performance*, pp. 25-36, (2004).
[13] Murata T., Petri nets: Properties, analysis and applications, *Proceedings of the IEEE*, Vol. 77, N°4, pp. 541-574, (1989).
[14] Saldhana J. & Shatz, S.M., UML Diagrams to Object Petri Net Models: An Approach for Modeling and Analysis, *International Conference on Software Engineering and Knowledge Engineering*, (2000).
[15] Pettit, RG. & Gomaa, H., Validation of Dynamic Behavior in UML Using Colored Petri Nets, *Workshop on Dynamic Behavior in UML Models: Semantic Questions*.
[16] Lopes, D., Hammoudi S., Bezivin, J. & Jouault, F., Mapping Specification in MDA: From Theory to Practice, *Proceedings of INTEROPESA*, pp. 253-264, (2005).
[17] INRIA, ATL Transformation Example, PathExpression to PetriNet, (2005).
[18] Bourdeaud'huy, T., Hanafi, S. & Yim, P., Efficient Reachability Analysis Of Bounded Petri Nets using Constraint Programming, *International Conference on Systems, Man and Cybernetics*, pp. 10-13 (2004).
[19] Bourdeaud'huy, T., Hanafi, S. & Yim, P., Mathematical Programming Approach To The Petri Nets Reachability Problem, *EJOR, European Journal of Operational Research*, Vol. 177, pp. 176-197, (2007).