An Interoperable Testing Environment for ERTMS/ETCS Control Systems

Gregorio Barberio[1], Beniamino Di Martino[2], Nicola Mazzocca[3], Luigi Velardi[4], Aniello Amato[1], Renato De Guglielmo[4], Ugo Gentile[3], Stefano Marrone[5], Roberto Nardone[3], Adriano Peron[3], and Valeria Vittorini[3]

[1] MATE Consulting s.r.l., Salerno, Italy
{g.barberio,a.amato}@tabit.it
[2] Seconda Università di Napoli, DIII, Aversa, Italy
beniamino.dimartino@unina2.it
[3] Università di Napoli "Federico II", DIETI, Napoli, Italy
{nicola.mazzocca,ugo.gentile,roberto.nardone,
adrperon,valeria.vittorini}@unina.it
[4] AnsaldoSTS, Napoli, Italy
{Luigi.Velardi,Renato.DeGuglielmo}@ansaldo-sts.com
[5] Seconda Università di Napoli, DMF, Caserta, Italy
stefano.marrone@unina2.it

Abstract. Verification of functional requirements of critical control systems requires a hard testing activity regulated by international standards. As testing often forms more than fifty percent of the total development cost, to support the verification processes by automated solutions is a key factor for achieving lower effort and costs and reducing time to market. The ultimate goal of the ongoing work here described is the development of an interoperable testing environment supporting the *system level testing* of railway ERTMS/ETCS control systems. The testing environment will provide a standardized interface to enable the integration testing between sub-systems developed by different companies/suppliers. We present the first outcomes obtained within the ARTEMIS project CRYSTAL which tackles the challenge to establish and push forward an Interoperability Specification (IOS) as an open European standard for the development of safety-critical embedded systems.

Keywords: Functional Testing, Railway Control System, Model-Based System Testing, Model Driven Engineering, Test Case Generation.

1 The RBC Use Case within the Crystal Project

The ARTEMIS Joint Undertaking project CRYSTAL (CRitical sYSTem engineering AcceLeration) [2] takes up the challenge to establish and push forward an Interoperability Specification (IOS) and a Reference Technology Platform (RTP) as a European standard for safety-critical systems. CRYSTAL is strongly industry-oriented and will provide ready-to-use integrated *tool chains* having a mature technology-readiness-level. To achieve technical innovations ("technology

A. Bondavalli et al. (Eds.): SAFECOMP 2014 Workshops, LNCS 8696, pp. 147–156, 2014.

bricks"), CRYSTAL adopts a user-driven approach based on applying engineering methods to industrially relevant *Use Cases* from the automotive, aerospace, rail and health-care sectors [15] and *increases the maturity of existing concepts* developed in previous European and national projects like CESAR [1], iFEST [3], MBAT [4]. The work described in this paper was born in the rail domain, and specifically from the needs expressed by Ansaldo STS (ASTS), an international transportation leader in the field of signalling and integrated transport systems for passenger traffic (Railway/Mass Transit) and freight operation. The industry needs expressed by the ASTS's *Use Case* are oriented to improve the quality and the efficiency of existing Verification & Validation (V&V) processes. In fact, testing activities are time-consuming tasks whose efficiency is a primary issue in a global competitive market and whose quality can not be decreased due to the adherence to international standards.

The ASTS's Use Case is centred on the Radio Block Centre (RBC) system, a computer-based system whose aim is to control the movements of the set of trains on a track area under its supervision, in order to guarantee a safe inter-train distance according to the ERTMS/ETCS specifications. ERTMS/ETCS (European Rail Traffic Management System/European Train Control System) [16] is a standard for the interoperability of the European railway signalling systems ensuring both technological compatibility among trans-European railway networks and integration of the new signalling system with the existing national train interlocking systems. Each ERTMS/ETCS controlled track is usually divided into several sub-tracks, each of them is supervised by a single RBC in charge of *concurrently and continuously* controlling a number of connections with trains. The main objective of the train control system is to timely transmit to each train its up-to-date Movement Authority (MA) and the related speed profile. The MA contains information about the distance the train may safely cover, depending on the status of the forward track. RBC is also in charge of managing emergency situations if the communication with one or more trains is compromised.

In this context, this paper presents an interoperable testing environment supporting the *system level testing* of railway ERTMS/ETCS control systems being developed within CRYSTAL. The paper is organized as follows: Section 2 provides a bird-eye view of the testing environment and related technological bricks involved in the ASTS *Use Case*. The subsequent Sections contain more information about the state of development of each Brick: Section 3 addresses modelling and test case generation carried out in the *Rail Model* Brick; Section 4 and Section 5 describe the architecture of the *IOP Test Writer* and *Log Analyzer* Bricks, respectively. Finally Section 6 describes how the Bricks are integrated into the CRYSTAL RTP/IOS.

2 An Environment for ERTMS/ETCS Interoperable Testing

The user needs expressed by ASTS within the CRYSTAL project are oriented to the automation of the system level testing activities, and to the realisation of

a tool chain providing full support to interoperable testing. In this Section the complete workflow of the automated testing process is described as well as the components of the tool chain and their relationships.

The proposed workflow complies with the ASTS *Use Case* requirements and will improve the current testing process, starting from the definition of the system specification to the generation of test reports. In detail, it enables semi-automatic generation of test cases from a set of test specifications, relying on a Model-Driven methodology. The generated test cases are then automatically transformed into executable test scripts, which can be executed on the real system or on simulation environments. Test logs are then analysed and test reports are automatically generated.

In Fig. 1 (a) the complete workflow is represented by an activity diagram, the tool chain and the involved technological bricks are described in Fig. 1 (b). The same figure also reports the links between each activity of the workflow and its supporting brick.

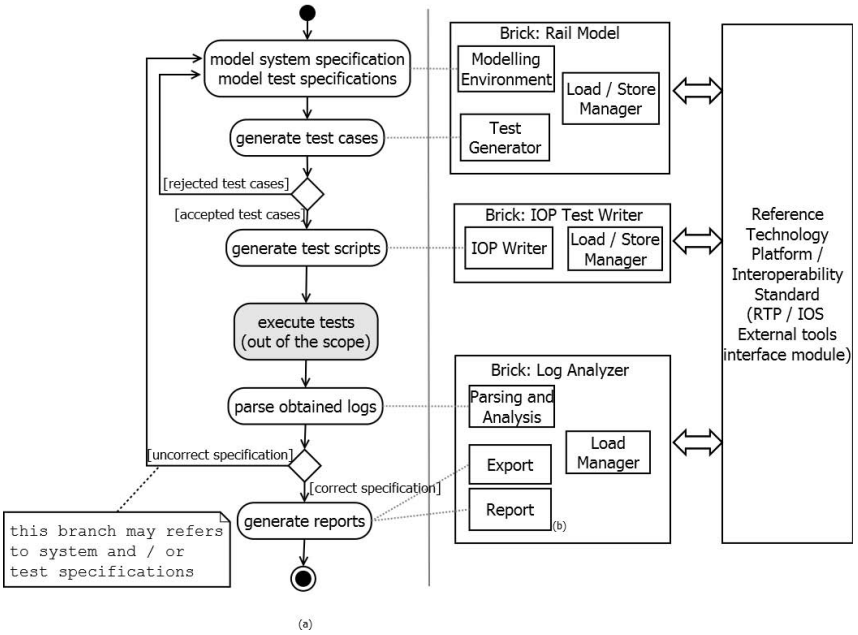

Fig. 1. CRYSTAL enhancements: the testing workflow (a) and the bricks implementing the tool chain (b)

With respect to the current process implemented in ASTS, this workflow introduces three main advantages:

– *automatic generation of test cases from test specification*: in the current process, adopted by ASTS, test cases are manually generated by domain experts

which are able to control the high complexity of these systems. This activity is heavy and error prone, in addition the training of new testers is very expensive as they must have a great experience, that could be acquired only working on several different projects.
- *generation of test script in IOP notation*: due to the heterogeneity of simulation environments, system level testing requires an interoperable testing environment where different simulators can exchange information. The generation of test scripts in IOP notation, under development by UNISIG, allows for the execution of interoperable tests in a multi-suppliers environment.
- *tool supported analysis of test logs*: the decision about the outcomes of test cases is currently performed by inspecting big log files; the adoption of a tool which is able to parse generated logs reduces the efforts of navigating them and generating reports.

The first activity of the testing workflow is the realization of a system specification, performed manually by V&V engineers by using a proper modeling language and a graphical modeling environment. The specification of the system is given by a high-level description of the system behaviour and by the set of functional requirements the system shall satisfy. By the same environment, the test specifications are defined in order to describe the essential features that a test case must accomplish (e.g., the sequence of transitions that the test case shall stress). Test cases are generated from the system and test specifications in semi-automatic way.

According to the proposed workflow, test cases can be analyzed (at the state this is not automated) and, if they are rejected by engineers, some updates on the source specifications can be performed. Test cases are then transformed into executable test scripts, through a transformation in the IOP notation. This language supports the creation of interoperable and multi-supplier testing environments.

Test scripts are executed and proper test logs are generated. The execution of test cases has not been considered in the CRYSTAL project, since each railway operator is interested in using proprietary testing environment. However test logs can be parsed in order to detect possible inconsistencies between planned test case and test execution. These inconsistencies can be due to wrong specification (and it is necessary the feedback to the source model), otherwise they trace bugs in the developed system. Finally test reports can be generated.

Fig. 1 (b) shows the tool chain that will support the execution of the proposed workflow. Three technological bricks will be developed in the CRYSTAL project, they are: *Rail Model*, *IOP Test Writer* and *Log Analyzer*. The first one provides the modeling environment to describe the system and the test specifications, it is also able to generate test cases implementing the appropriate transformations to and from the specific model checker, as explained in the next Section. The second brick supports the generation of test scripts written in IOP Language (according to the concept of an interoperable testing environment). The last one is able to parse the test logs and to generate test reports from them. Some details about these technological bricks are given in the following Sections, as well as some technological hints.

3 Rail Model: Model-Based Test Sequences Generation

The inputs of the *Rail Model* brick are a formal state based specification of the system behaviour and of its requirements. From them, chains of model transformations allows to obtain Test Cases by applying model checking techniques. This approach is based on the previous experiences on extending UML with the V&V UML Profile [10,14]. The approach under development envisages a) the definition of a proper formal state-based language (DSTM4Rail) to be used for modeling the behaviour of the system under test and to formalize the requirements from which the test specifications are obtained; b) the implementation of the transformation chains in order to obtain the test cases; c) and the definition of a proper set of test specification patterns which will provide general reusable models for recurrent classes of requirements.

The CENELEC standards [6,7] explicitly recommend the usage of state-based formalisms, as the dynamic of critical control systems, based on a sequential computation, can be abstracted as a state-transition system. Despite the great number of works addressing the usage of state machine and their extensions, within CRYSTAL the railway industry expressed the need for a concise formal modeling notation, able to easily capture some characteristic features of the specific domain, to be used in model-driven test automation environments. In particular, at the state, ASTS keeps out the possibility of using several UML diagrams and prefers an ad-hoc formal language, developed from scratch, with the objective to be as simple as possible and as rich as needed for modeling the behaviour and the requirements of a railway control system for system testing purposes.

DSTM4Rail extends Hierarchical State Machines [5]. Its peculiarity mainly resides in the semantics of fork-and-join which allows dynamic (bounded) instantiation of machines (processes) and parallel execution of machines inside a box. Each state machine may be parametric over a finite set of dynamically evaluated parameters; in addition the same machine may be dynamically instantiated many times without explicitly replicating its entire structure.

DSTM4Rail also allows to model the requirements and add proper information to the behavioural models for implementing requirement traceability. ATL (Atlas Transformation Language) [13] is used to translate the DSTM4Rail model of the system to a NuSMV [8] or Promela [12] specification and the DSTM4Rail model of the requirement to verify into a CTL/LTL specification or into a Promela property (e.g., a never claim) which are added to the NuSMV or Promela model, respectively. Indeed, it is well known that test case generation may be obtained by using the ability of a model checker to construct counterexamples to violated properties: a counterexample defines the sequence of steps which are interpreted as a test case [11].

Fig. 2 shows a DSTM4Rail specification model of a particular functionality of RBC (i.e., *Management of the train movement* in the box) and its realization in the prototype modeling environment.

During the movement of the train, RBC periodically sends the Movement Authority (MA) to the train (Section 1). Concurrently, RBC has to monitor the

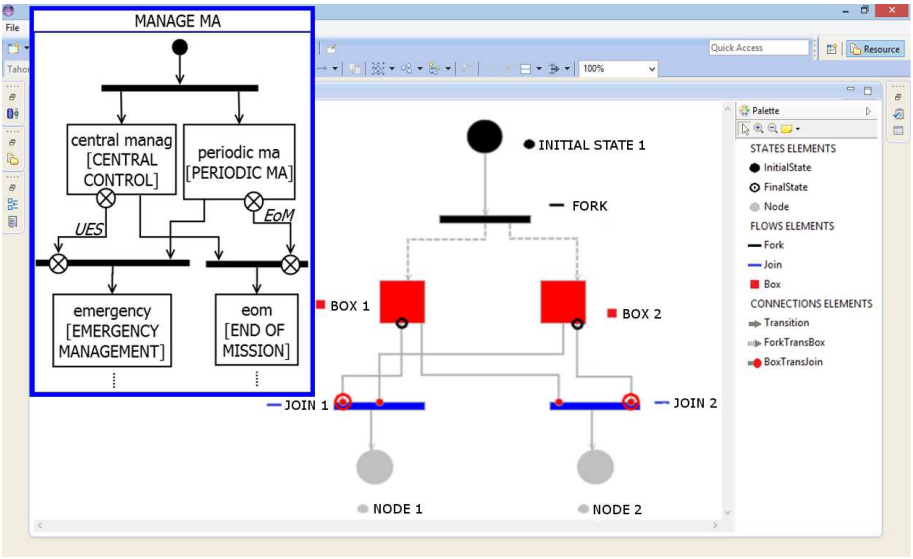

Fig. 2. A DSTM4Rail model and a tool screenshoot

commands that come from the Centralized Traffic Control (CTC) where a human operator may raise an alarm which requires the train to brake: in this case an Unconditional Emergency Stop (UES) message is sent to the train. On the other hand, when the train successfully ends its trip, it performs the End of Mission (EoM) procedure. This scenario needs for representing concurrently executing machines, one of whom may force the termination of the others. DSTM4Rail models this situation by a preemptive join, as shown in Fig. 2 where the processes CENTRAL CONTROL and PERIODIC MA are executed concurrently but, when the first machine reaches the UES exiting node, the join on the left forces with preemption the process PERIODIC MA to terminate. In this case the machine EMERGENCY MANAGEMENT is instantiated. On the contrary, if the process PERIODIC MA terminates in the EoM exiting node, the join on the right forces with preemption the CENTRAL CONTROL to terminate, and the END OF MISSION machine is instantiated.

State of the development : up to date a first and stable version of the DSTM4Rail formalism has been implemented, within the Eclipse Modelling Framework (EMF), by an Ecore metamodel. To represent DSTM4Rail in a user-friendly way, a graphical editor has been realized through functionalities provided by the Eclipse Graphical Modeling Framework (GMF). A screen shot of the graphical environment is shown in Fig. 2.

The implementation of the transformations is an ongoing activity. At the state, a set of test specification patterns has been also defined, based on the notion of Dwyers's property specification patterns [9].

4 IOP Test Writer

The *Rail Model* brick, described in Section 3, generates the sequence of the steps specifying the test case. These traces need to be translated into an concrete notation in order to be executed on simulated environments. Since ERTMS/ETCS based infrastructures are composed by different subsystems that can be supplied by different technology providers, the testing and simulation environments reflect this heterogeneity being a federation of different simulators developed by different teams. One of the requirements for a an interoperable testing environment is that each component must speak a common "testing" language. The IOP Notation is developed by the UNISIG (Union Industry of Signaling). Properly interpreted by vendor-specific adapters, IOP can support the creation of integrated testing environments. The aim of the IOP notation is not limited to simulated environments as it can be used to give commands and interpreting the states of real systems: the scenario depicted in the Fig. 3 exemplifies its usage in on-field testing.

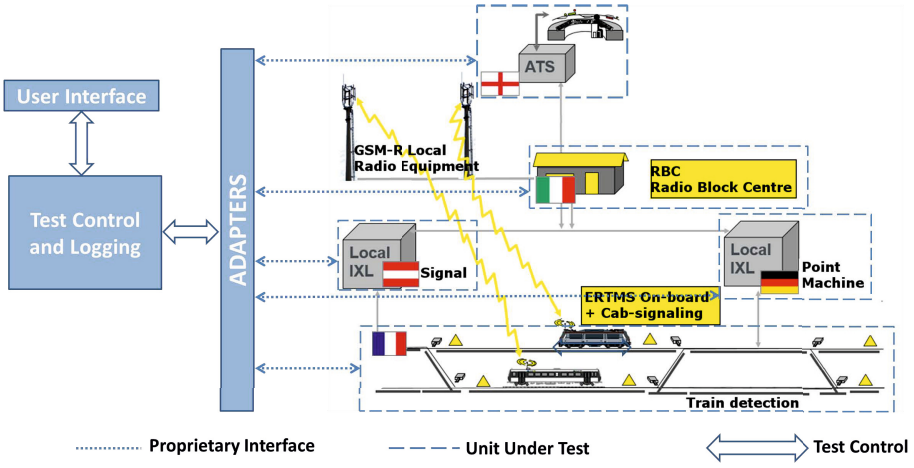

Fig. 3. IOP Test Writer

Since all the ERTMS technology providers are interested in increasing the level of interoperability of their products, ASTS is moving its testing tool set and environments to the IOP notation. According to this wish, the IOP Test Writer can extremely accelerate the test implementation phase by automatically generating the test scripts written in IOP language from the test cases created by Rail Model.

The IOP Test Writer tool will consists of two modules according to software engineering best practises: the *TestWriter* and the *Load/Store Manager*. Each module performs some specific tasks and shows a set of interfaces used to interact with the other module. More specifically:

- the *Load/Store manager* module provides the interfaces to interact with the external modules (for example the RailModel) and with the other technologies within the Crystal IOS. *Load/Store manager* module loads the test sequences produced by the Rail Model Brick and stores their results in the system: such interaction is accomplished by means of IOS/RTP;
- the *IOP Writer* module provides the transformation of the test cases expressed in a Ecore language to a test cases expressed in IOP notation by means of a Model-to-Text (M2T) transformation.

State of the development: up to date, the IOP Test Writer is in its design phase. The idea is to develop it in Java as an Eclipse plug-in to make the integration with the Rail Model brick easier.

5 Log Analyzer

The Log Analyzer brick aids the V&V engineer to state if the execution of a specific test passes/fails. Hence, the Log Analyzer has two different sources: (1) a test case as generated by the Rail Model brick, (2) the logs created by the execution of a test case (after its translation into the IOP notation) on the specific testing environment. According to such inputs, the Log Analyzer may find if logs and the test case match. Such operation would be pointed out to the V&V engineer who is able to decide if the test passes, fails (due to an error in the system model) or fails (due to a misinterpretation of the requirements).

The Log Analyzer has a modular architecture according to software engineering best practises:

- the *Load manager* module provides the interfaces to interact with the external modules and with the other technologies within the Crystal IOS. The *Load manager* loads the test sequences produced by the Rail Model Brick: such interaction is accomplished by means of IOS/RTP;
- the *Parsing and Analysis* module is used to parse the logs of the test executions. Moreover, each log is analyzed according to the inputs and then the fail/pass decision is taken for the single log.
- the *Report* module focuses on building up summary information about the entire testing campaign and in generating supporting tables for traceability, coverage, etc.;
- the *Export* module has the role to export the report of a testing campaign and the related execution logs. The target report will be conform to widespread document formats (e.g. pdf files or spreadsheets).

State of the development: up to date, the Log Analyzer is in its design phase. As it is not called automatically after the first two bricks, the design teams of this tool is evaluating the possibility to implement it as a stand-alone tool.

6 Integration in RTP/IOS

As shown in Figure 1, each Technology Brick interacts with an External Tool Interface module. This External Tool Interface module implements a custom solution for the integration of RailModel, IOP Test Writer and Log Analyzer or a flexible solution for a generic integration, based on the technologies available in Crystal Reference Technology Platform / Interoperability Specifications (RTP/IOS). The RTP is a generic platform for the integration of model-based tools. It is composed by a set of interoperable tools, methods and processes designed to increase the quality of development processes of safety critical embedded systems. This integration platform will host tools coming from different stakeholders (vendors, industrial & academic partners, etc.) that realise Bricks within the project. Therefore there is the need for a common non-proprietary standard to realise this interoperability functionality within the RTP.

The CRYSTAL IOS would accomplish this task by adopting the Open Services for Lifecycle Collaboration (OSLC), a framework that moves to the integration of data, workflows and processes among product lifecycles. OSLC is divided in several workgroups each of which addressing specific integration scenariosThe set of scenarios and specifications are named *OSLC Domain*. The presence of different domains introduces the necessity to manage the coherence among them. This need is satisfied by a set of standard rules and patterns, contained in the OSLC Core Specification, and all the domain groups must adopt these rules for the specifications. The union of a OSLC Core Specification and a OSLC Domain constitutes a OSLC protocol that is used in order to add interoperability to a specific tool chain, as in CRYSTAL. Some work packages in CRYSTAL are devoted to the study of existing standards and to the proposition of proper technological solutions in order to integrate the Technology Bricks with the RTP/IOS.

Here a brief description of how the tool chain presented in this paper will be integrated in the RTP/IOS is reported. According to the Fig. 1, the tool chain uses the RTP by loading/storing the artifacts generated during the process. Each brick interacts with the RTP/IOS via its Load/Store Manager. More specifically:

- Rail Model loads existing DSTM model of the system/test specification previously defined and stores DSTM models created with the Modelling Environment. The module may store test cases generated by the Test Generator module;
- IOP Test Writer loads existing test cases from the RTP/IOS. In addition it stores IOP compliant test cases generated by the IOP Writer;
- Log Analyzer loads existing test cases from the RTP/IOS. Both the logs of the test executions and the report are stored out of the RTP/IOS.

Acknowledgments. This paper is supported by research project CRYSTAL (Critical System Engineering Acceleration), funded from the ARTEMIS Joint Undertaking under grant agreement number 332830 and from ARTEMIS member states Austria, Belgium, Czech Republic, France, Germany, Italy, Netherlands, Spain, Sweden, United Kingdom.

References

1. CESAR: Cost-Efficient methods and proceses for SAfety Relevant embedded systems, http://www.cesarproject.eu/
2. CRYSTAL: CRitical sYSTem engineering AcceLeration, http://www.crystal-artemis.eu/
3. iFEST: industrial Framework for Embedded Systems Tools, http://www.artemis-ifest.eu/
4. MBAT: Combined Model-based Analysis and Testing of Embedded Systems, http://www.mbat-artemis.eu/
5. Alur, R., Kannan, S., Yannakakis, M.: Communicating hierarchical state machines. In: Wiedermann, J., Van Emde Boas, P., Nielsen, M. (eds.) ICALP 1999. LNCS, vol. 1644, pp. 169–178. Springer, Heidelberg (1999)
6. CENELEC. Cenelec, en 50128: Railway applications - communication, signalling and processing systems - software for railway control and protection systems (2011)
7. CENELEC. Cenelec, en 50126: Railway applications - demonstration of reliability, availability, maintainability and safety (rams) - part 1: Generic rams process (2012)
8. Cimatti, A., Clarke, E., Giunchiglia, F., Roveri, M.: Nusmv: a new symbolic model checker. International Journal on Software Tools for Technology Transfer 2 (2000)
9. Dwyer, M.B., Avrunin, G.S., Corbett, J.C.: Patterns in property specifications for finite-state verification. In: Proceedings of the 21st International Conference on Software Engineering, ICSE 1999, pp. 411–420. ACM, New York (1999)
10. Flammini, F., Marrone, S., Mazzocca, N., Nardone, R., Vittorini, V.: Model-driven V&V processes for computer based control systems: A unifying perspective. In: Margaria, T., Steffen, B. (eds.) ISoLA 2012, Part II. LNCS, vol. 7610, pp. 190–204. Springer, Heidelberg (2012)
11. Gargantini, A., Heitmeyer, C.: Using model checking to generate tests from requirements specifications. SIGSOFT Softw. Eng. Notes 24(6), 146–162 (1999)
12. Holzmann, G.: Spin Model Checker, the: Primer and Reference Manual, 1st edn. Addison-Wesley Professional (2003)
13. Jouault, F., Kurtev, I.: Transforming models with ATL. In: Bruel, J.-M. (ed.) MoDELS 2005. LNCS, vol. 3844, pp. 128–138. Springer, Heidelberg (2006)
14. Marrone, S., Flammini, F., Mazzocca, N., Nardone, R., Vittorini, V.: Towards model-driven v&v assessment of railway control systems. International Journal on Software Tools for Technology Transfer, 1–15 (2014)
15. Pflügl, H., El-Salloum, C., Kundner, I.: CRYSTAL, CRitical sYSTem engineering AcceLeration, a Truly European Dimension. ARTEMIS Magazine 14, 12–15 (2013)
16. UIC. ERTMS/ETCS class1 system requirements specification, ref. SUBSET-026, issue 2.2.2 (2002)