Model-Driven V&V Processes for Computer Based Control Systems: A Unifying Perspective

Francesco Flammini[1], Stefano Marrone[2], Nicola Mazzocca[3], Roberto Nardone[3], and Valeria Vittorini[3]

[1] AnsaldoSTS, Innovation and Competitiveness Unit (Italy), via Nuova delle Brecce 260, 80147 - Napoli, Italy
francesco.flammini@ansaldo-sts.com

[2] Seconda Università di Napoli, Dipartimento di Matematica, viale Lincoln, 5, 81100 - Caserta, Italy
stefano.marrone@unina2.it

[3] Università di Napoli "Federico II", Dipartimento di Informatica e Sistemistica, Via Claudio 21, 80125 Napoli, Italy
{nicola.mazzocca,roberto.nardone,valeria.vittorini}@unina.it

Abstract. A recent trend in software engineering is to support the development process by providing flexible tool chains allowing for effective Model-Driven approaches. These solutions are very appealing in industrial settings since they enable the creation of development and verification processes, enhancing abstraction and reuse, and hence improving productivity. This paper addresses advantages and challenges in extending Model-Driven approaches to system engineering and specifically to verification and validation (V&V) of critical computer-based systems. Specifically, the paper highlights the needs for real-world industrial contexts and proposes the definition of a unifying Model-Driven process for V&V of functional and non-functional system properties. Some enabling techniques which aim at improving the reuse of Model-Driven artifacts are addressed to deal with process scalability and effectiveness. Two sample applications are described for ERTMS/ETCS signalling system in order to show the advantages of the approach: formal modeling for performance evaluation of message delivery between train and track controllers and test case generation for the verification of functional requirements of trains outdistancing.

Keywords: Model-Driven Engineering, Verification & Validation, Critical Systems, Domain Specific Languages, Railway Systems.

1 Introduction

Verification & Validation (V&V) processes within critical control systems development must guarantee the fulfillment of both functional and RAMS (Reliability Availability Maintainability Safety) requirements [11]. Two main approaches are employed in order to predict/evaluate the dependability attributes of those systems: the first relies on simulation based techniques, e.g. fault-injection [14] at the hardware level (either physical or simulated) or software testing [8] at the various abstraction and integration levels; the second is based on formal methods, which can be used at any abstraction level (both hardware and software) and at any stage of system development and validation. Both simulative and formal approaches are used in real world applications, for different or same purposes, and can be classified as model-based techniques, as they require designers to generate an accurate model of the system under analysis and of the external environment (i.e. interacting entities). They may be used in combination with formal models possibly interacting with simulative ones (an example of this is the class of approaches known as model-based testing). Formal methods are employed in a variety of industrial applications, from microprocessor design to software engineering and verification (see [9] for a survey of widespread methods and applications). Despite of such a variety of methods, tools and applications, V&V activities are still critical in costs and results. The optimization of V&V processes is the focus of several ongoing national and international projects carried out in industrial settings [2]. A recent trend is to define and develop tool chains to support the developer in the V&V process. They are based on the Model Driven Engineering (MDE) paradigm and rely on the usage of models as the primary artifact in the development cycle. The idea to derive dependability models (e.g. Stochastic Petri Nets models) from high-level specifications of the system to be developed (e.g. expressed by UML) is not new ([5,6,24,22,7]). The cutting edge between those approaches and MDE may be summarized by the following words: integration, automation and traceability. That means a complete suite of integrated tools covering and linking all the stages of the V&V process has to be available, featuring automated generation of artifacts (models, test cases, log files, etc.) and requirements traceability. This paper describes how MDE may provide a unifying framework for V&V activities for critical systems, and specifically how this can be applied to railways control systems. Section 2 introduces some MDE concepts, starting from its primary nest: software development; Section 3 presents a brief state-of-the-art on a) Model-Driven analysis of non-functional properties of systems, according to its main fields of application, b) Model-Driven approach to functional testing. In Section 4 an overview of a Model-Driven approach to V&V is given; it is then applied to the railway domain in Section 5. Section 6 contains a brief discussion on some open issues and challenges.

2 The Model-Driven Approach

MDE approaches are very used in software development to support software production: they are the starting point from which source code, can be generated in an automatic or semi-automatic way. Models of software systems are usually constructed by using visual modeling languages like UML, SysML, MatLab Simulink/Stateflow; successively, transformational approaches are applied to automate the overall process.

In this context, the Object Management Group (OMG) has developed a set of standards providing an advanced meta-modeling architecture (Model Driven

Architecture, MDA [4]) whose primary goals are to cope with complexity and heterogeneity of different platforms and application domains and obtain automation and reuse. These goals are pursued by abstracting at three different levels: the *platform independent model (PIM)* is a model of the system structure and functions which must be independent by specific technical details related to its implementation, it is translated by proper Model to Model (M2M) transformations into one or more *platform specific models (PSMs)* and then to *system code.*

MDA relies on OMG standards including MOF (Meta-Object Facility, which is used to define modeling languages), UML (that is a MOF model), and QVT (Query/View/Transformation, a standard language for model transformation). MDA processes are supported by well established workbenches and tool chains, based on easily extensible plug-in systems such as Eclipse. Nevertheless, MDE is more general than MDA. Its strength is in using abstract representations of concepts and activities that characterize a specific application domain. Hence, it could be applied to several fields and to different purposes. In this paper we show how MDE may be used to define a V&V process for critical control systems according to a unifying perspective able to held together (semi) automatic generation of code, test cases, and dependability models. This may be done by exploiting the usage of proper languages to represent domain specific concepts and solutions at the conceptual level. This is a tricky aspect of the question that is briefly discussed here below.

2.1 Domain Specific Languages

UML is a general purpose modeling language, rich of modeling notations and semantic, which can be applied to a wide class of application domains. Despite of this great advantage, the effective usage of Model-Driven Development solutions in industrial settings asks for the availability of specialized modeling languages for several reasons: Domain Specific Modeling Languages (DSMLs) are small and well focused on domain scope, they simplifies the design process, tracing recurring design patterns in the application domain, and promote communication by standardizing the terminology and the best practices to be used in the specific application domain. Domain specific concepts are grouped into a domain meta-model, which defines the relationships among them and precisely specifies semantics and constraints associated with the domain concepts. The definition of a DSML is an activity performed by "language engineers" and it is still an emerging discipline with few established guidelines and patterns. Three main approaches to the definition of a DSML are reported in the literature [23]: (1) definition of a new modeling language from scratch; (2) extension of an existing modeling language by supplementing it with fresh domain specific constructs; and (3) refinement of an existing more general modeling language, as UML, by specializing some of its general constructs to represent domain specific concepts. Clearly the first one allows a precise characterization of domain specific concepts, but it requires the implementation of the model editors that involves an extra effort when put into practice. The second one suffers from the same problems but it can rely on the experience. The third one is more practical and presents minor development and maintenance costs: it bases on UML profiling techniques when UML is the general modeling language chosen.

The proposed approach is based on the third solution since inheriting UML syntax and semantics avoids the re-definition of existing concepts (e.g. state machines); moreover UML (and its good tool support) shortens the time to realization of languages that is of great interest for industries.

3 Model Driven V&V of Critical Systems

Model-driven approaches have been extended to support V&V activities, both for software and complex systems in general, in order to prove properties (Model-Driven Analysis) and to generate test cases (Model-Driven Testing). This is possible applying the two main principles, described previously, on which a Model-Driven process is based: the definition and usage of an high-level model for the system, and model transformation techniques. Model-Driven Analysis and Model-Driven Testing are separate techniques and are also supported by different tools but, since they rely on the same high-level model, some attempt to integrate them have been tried and are still under study: an ARTEMIS on-going project MBAT [2] is an example of a European project that will provide a new leading-edge V&V technology in form of a Reference Technology Platform (MBAT RTP) for effective and cost-reducing validation and verification, primarily focusing on transportation domain combining Model-Driven Analysis and Testing techniques.

3.1 Model-Driven Analysis

The goal of Model-Driven Analysis is to construct formal models or artifacts, able to verify requirements, from input design models (high-level model) assuring the achievement of system quality, such as safety targets. Several projects addresses the analysis of dependability attributes of complex systems, based on MDE principles (e.g. the projects PRIDE [3], CHESS [1]), even if performance evaluation is perhaps the most addressed feature assessed in the literature by means of Model-Driven Analysis (see for example [18], [20] and [21]) The ArgoSPE tool [12] implements a performance3 evaluation process translating some performance annotated UML diagrams into Stochastic Petri nets models. MARTE [17] is a UML profile which intends to replace the UML profile for Schedulability, Performance and Time, adding capabilities to UML for Model-Driven development of Real Time and Embedded Systems. The MARTE profile is able to annotate, in an high-level model, system non functional properties (NFPs), according to a well-defined Value Specification Language (VSL) syntax. In a recent work [6] the "Dependability Analysis and Modeling" (DAM) profile has been proposed to extend MARTE with dependability concepts (e.g., annotating a UML State Machine transition as a failure step). Hence, DAM is useful to annotate dependability requirements and properties in UML specifications, in particular, reliability, availability, maintainability and safety. The DAM domain model represents the main dependability concepts according to a component-based view of the system under analysis. The system is defined as a set of components and delivers a set of services that can be detailed as a set of steps. Possible hw/sw redundancies are modeled through the redundant structure, made of fault tolerant components which can play different roles. The system can be affected by threats according to the fault, error, impairment (failure or hazard) chain. The maintenance actions are modeled through maintenance model, which includes the concepts necessary to represent components repair and service restore.

3.2 Model Driven Testing

Model-Driven Testing techniques deal with the efficient and automated generation of test cases from different kinds of models. Model-Driven Testing promises higher quality and conformance to the respective functional safety and quality standards at reduced costs through increased coverage, advanced test generation techniques, and increased automation of the process, including support for certification. As depicted in Fig. 1, Model-Driven Testing applies the same abstraction of platform independent model (PIM) and platform specific models (PSM) concepts into the test design model: it have been introduced the concepts of platform specific test design model (PST) that can be derived from platform independent test design model (PIT) ([10]). Both PIT and PST can be refined and enriched with test specific properties and it is possible to obtain from them executable test suites (and code) with the aim to verify the properties. To date transformations between the different abstraction levels (from platform independent to platform specific, and from models to executable test) have been made, but only few progress in the transformations between system models and test models are remarkable, in particular for non-software systems.

A recent OMG standard, the UML Testing Profile (UTP), defines a language for designing, visualizing, specifying, analyzing, constructing and documenting test cases [19]. This language can work with all major object and component technologies and can be applied in various application domains. UTP defines a MOF-based meta-model, enabling compliance between MOF-based tools and UTP standard. In4 of the profile, in which the behavioral aspects are left out because they are considered not relevant and require an inclusion of a significant portion of the UML 2.0 meta-model. For these reasons the UTP can be used standalone or in an integrated way with UML. This profile introduces four logical concept groups, that include test specific concepts, covering the following aspects: *test architecture*, *test behavior*, *test data* and *test time*. The *test architecture* contains the concepts able to describe the organization and the realization of test cases. One or more objects can be stereotyped as the *SUT* (system under

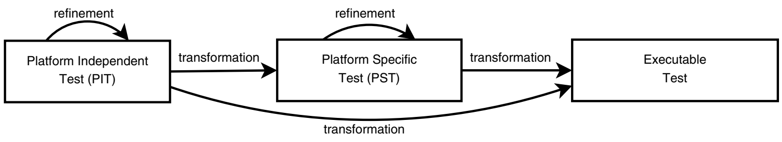

Fig. 1. Model Driven Testing reference schema

test), that refers to a system, subsystem, or component that is being tested. The features and behavior of the SUT is given entirely by the type of the property to which the stereotype is applied; the internal portion of the SUT is not known during the test execution due to its black-box nature. Different test cases can be groupend into a *TestContext*, the *TestContext* is realized by a set of *TestComponent*s able to communicate with the SUT. The *TestContext* is also connected with an *Arbiter* able to determine the final outcome, the *Verdict*, of a test case. The concepts of *text behavior* specify the behavior of test cases: one *Behavior* is included into each *TestCase*. *TestCase*s are connected with *TestLog* entities, able to log information. The concepts of *test data* group are able to specify data values. They include wildcards for a flexible data definition such as special characters for "any value" and "any or omitted value" definition. At last *test time* defines time concepts for a precise time specification using the primitives of *Time* and *Duration* to define respectively time values and duration. In [10] a set of transformations from UML model to UTP model is showed, proposing to generate test cases using three layers of transformations that are UML to PIT, PIT to PSM and PSM to testcase.

4 How It Could Be Used: A Unifying Approach

In this Section a unifying "industry-friendly" Model-Driven approach for V&V processes (for both formal analysis and testing) is presented. In Fig. 2 a reference schema for this approach is provided.

The first step is related to meta-modeling activities: proper languages, if not available, should be created by means of extension and/or merging of existing languages. These may be domain-independent (as MARTE-DAM and UTP) or domain-specific according to their focus on technical or business concepts. In our approach we need both kinds of languages since the first improves reusability while the second improves usability. In the proposed approach the Verification&Validation Profile (VVP) (an "horizontal" language) is created: it extends MARTE-DAM and UTP. Moreover it could be possible to specialize VVP concepts into a specific business domain (*Specific*).

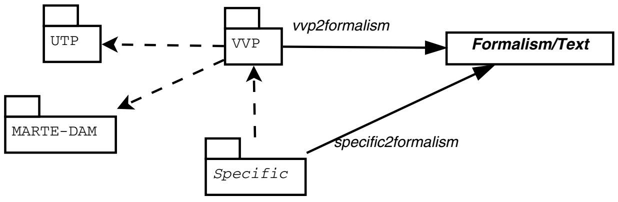

Fig. 2. Unifying Model-Driven V&V approach

Model transformations are defined generating a formal model according to a well specified formalism[1] from an high level model expressed into VVP or Specific. The definition of model transformations can be a modular task too, since it is possible to exploit composition and inheritance techniques [16]. Best results can be obtained if model transformations are defined on the basis of VVP since these transformations allow every specific derived language to inherit them improving reusability. Nevertheless some peculiarities of the Specific language may need to partially create further model transformations in order to best translate these features into the target formalism (*specific2formalism*).

4.1 Focus on the V&V Profile

In this subsection we define a V&V domain model to merge the concepts represented in the two cited UML profiles: MARTE-DAM and UTP. The first is used to model both performance and dependability aspects and the second allows the modeling of system and software testing. Fig. 3 depicts the V&V domain model we constructed to obtain the VVP.

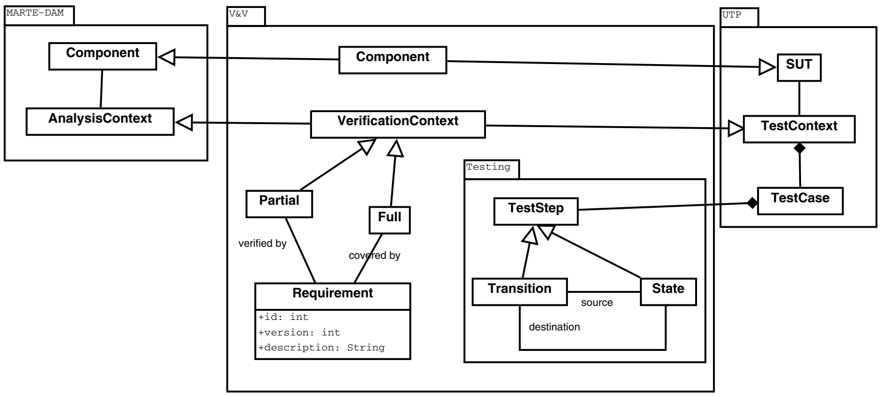

Fig. 3. Extract of V&V domain model

Notwithstanding these two languages need to be extended since they do not provide to all the features required in real industrial processes. In general three kinds of operations can be applied to existing profiles: *merge*: resulting domain model contains all the concepts of existing ones; *extension*: resulting domain model is refined by adding new concepts; *synthesis*: redundant concepts are reduced refactoring the resulting domain model. In the construction of the V&V domain model, first we extended UTP domain model by adding some important features in the modeling of the behaviour of system under test: under the hypothesis to model the behaviour of a component in terms of state machines (a

[1] If we think about performance analysis, suitable formalisms are: Queueing Networks, Petri Nets, etc.

very common practice in industry), the UTP's *TestCase* is refined by defining *TestStep*, a elementary unit of the *TestCase*, that can be a *State* or a *Transition* of the state machine. Other important added concepts are related to specification of requirements (the *Requirement* class) and the *VerificationContext* (that can be reported to a series of *TestStep*s). Testing specification and case can be modeled by specializing *VerificationContext* in *Partial* and *Full* according to the test detail level. Then some synthesis are made between some elements of MARTE-DAM and UTP (e.g. *Component* for both MARTE-DAM's *Component* and UTP's *SUT*). Finally the VVP is generated by the V&V domain model [15].

5 Application to Railway Signalling

The proposed V&V Model-Driven approach is here instantiated to the railway signalling domain. In particular it will be studied the European Railway Traffic Management System/European Train Control System (ERTMS/ETCS) [25] that is a standard for the interoperability of the European railway signalling systems in charge of providing the safe movement of trains and the optimal regulation of traffic flows.

5.1 The ERTMS/ETCS System

The mission of ERTMS/ETCS is to ensure railway interoperability. To this aim, it provides the specification of a traffic management and train control system that enables the transit of high speed trains through national borders. The ERTMS/ETCS standard ensures both technological compatibility among transeuropean railway networks and integration of the new signalling system with the existing national train interlocking systems. An ERTMS/ETCS system consists of heterogeneous, distributed components that are installed on the trains, along the tracks and in several control centers. A reference schema for ERTMS/ETCS systems is shown in Fig. 4. It consists of the Radio Block Centre (RBC), that is a central computer responsible of an entire track area, and the European Vital Computer (EVC), that is the on board controller. The communication between these two subsystems is provided by the GSM-R network. The control of the movement of the train is realized by means of a message that RBC sends to EVC: the authorization to safely move within a defined area that is called Movement Authority (MA)[2]. Attached to the MA, additional information describing a Temporary Speed Restriction (TSR) inside the length of the MA may be sent to the train.

In this paper we focus on V&V of both non-functional and functional properties of the delivery of the MA. We consider two representative requirements:

1. $U_{TX} < 1.6 * 10^{-5}$ where U_{TX} is the unavailability due to transmission error of communication networks [26];

[2] The MA is built according to the information about train position and speed each EVC periodically sends via GSM-R to RBC (Position Report).

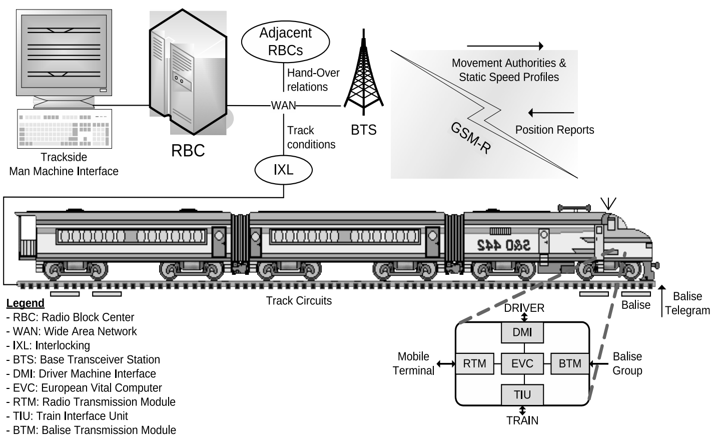
Adjacent RBCs
Hand-Over relations
WAN
Track conditions
IXL
BTS
RBC
Trackside Man Machine Interface
Movement Authorities & Static Speed Profiles
GSM-R
Position Reports
Track Circuits
Balise
Balise Telegram
DRIVER
DMI
Mobile Terminal
RTM
EVC
BTM
Balise Group
TIU
TRAIN
Legend
- RBC: Radio Block Center
- WAN: Wide Area Network
- IXL: Interlocking
- BTS: Base Transceiver Station
- DMI: Driver Machine Interface
- EVC: European Vital Computer
- RTM: Radio Transmission Module
- TIU: Train Interface Unit
- BTM: Balise Transmission Module

Fig. 4. ERTMS/ETCS reference schema

2. the message containing the TSR is sent periodically to EVC until EVC does not ack; if EVC does not send any ack message, RBC must send a braking command (Unconditionally Emergency Stop - UES) [25].

The first requirement can be verified by a performance evaluation of MA message delivery while the second by means of a functional test. The performance analysis is provided by automatic generation of Generalized Stochastic Petri Nets (GSPN) in Subsection 5.2; the functional test case is generated by model checking techniques in the Subsection 5.3 and is supported by the definition of a model transformation into Promela language [13]. Hence the general schema depicted in Fig. 2 is instantiated into the one depicted in Fig. 5.

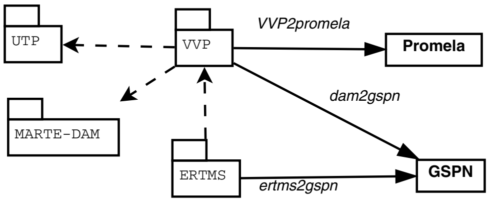

Fig. 5. ERTMS specific Model-Driven V&V

5.2 A DSML for GSM-R Unavailability Analysis

The MA is also used in some implementations as a channel monitoring message. If a train does not receive a new MA within a chosen number of seconds after the last received message, EVC tries to re-establish the connection within a specified timeout period and the following situations may happen: if EVC does not receive any valid message within a timeout, it brakes and passes in a degraded mode from which can exit after a reconnection procedure. VVP is specialized into ERTMS/ETCS domain specific language. A sample of this domain model is depicted in Fig. 6 where the three cited components are represented by three UML classes, each of them containing proper attributes. The meaning of the parameters of EVC and RBC classes are briefly described in the following:

– *numRetry*: number of reconnection attempts by the EVC;
– *timeToRestore*: mean time from a disconnection to the next balise group commanding a recall to the RBC;
– *timeToRetry*: time between reconnection attempts;
– *timeToBrake*: time-out after that a received message is no more valid;
– *messageCycle*: time between monitoring messages sent by RBC;

GSM-R networks does not need to be fully characterized by specific attributes since some quantitative parameters needed for a performance analysis are contained in some clases derived from MARTE and MARTE-DAM:

– *ssAvail* (from MARTE-DAM): unavailability of GSM-R connection;
– *packetTime* (from MARTE): mean message transmission time (in milliseconds) on the GSM-R network;
– *trasmissionError* (newly added): probability of a messages being corrupted during transmission.

According to this specific domain model (and the relative UML profile), we can describe the situation where a EVC and a RBC are connected by a redundant GSM-R network as in Fig. 7. The tagged value *ftLevel* is the level of fault tolerance of the *RedundantStructure*: if set to 1, it means that at least one operating GSM-R is needed to accomplish delivery service.

Dam2gspn and *ertms2gspn* are defined after the definition of a metamodel for GSPN language, omitted for brevity. These transformations are implemented in Atlas Transformation Language (ATL): for clarity the rules defining the transformations are described by means of the generated GSPN subnets. These rules translate: the redundancy of GSM-R networks, single GSM-R behaviour and RBC. The GSPNs are respectively depicted in Fig. 8 (*DaRedundantStructure* with *ftLevel* = 1), Fig. 9 (a) (*GSMR* stereotype) and Fig. 9 (b) (*RBC* stereotype).

The i-th "cloud" in the GSPN of Fig. 8 is filled with one of the GSPN of Fig. 9 (a) by means of the superposition of transition couples (*OK*,*Replicai_out*) and (*FROM_RBC*,*Replicai_in*), then the *in* transition of RedundantStructure

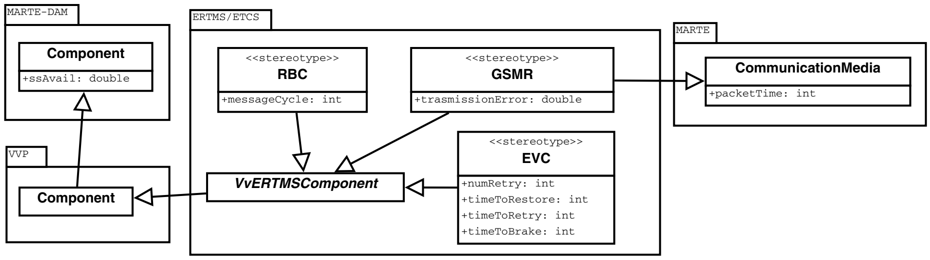

Fig. 6. ERTMS/ETCS domain model

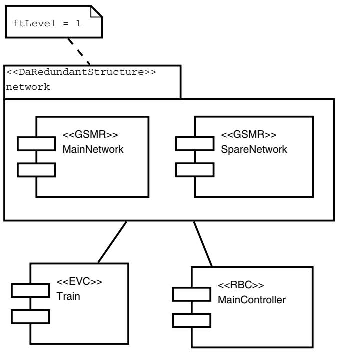

Fig. 7. ERTMS/ETCS Performance Model

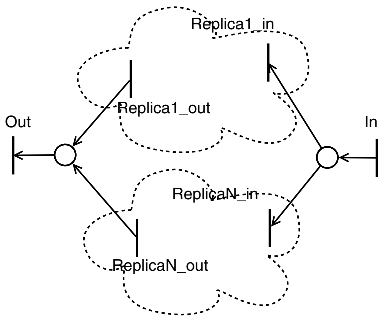

Fig. 8. RedundantStructure GSPN pattern

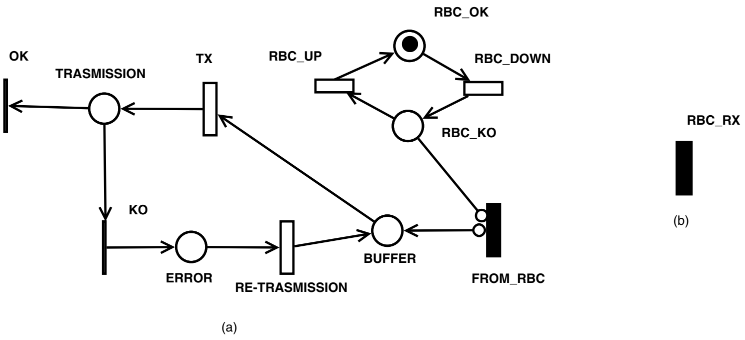

Fig. 9. GSPN patterns of GSM-R (a) and RBC (b)

net is superposed with *RBC_RX* of RBC GSPN model Fig. 9 (b) while the *out* transition is then linked to the EVC GSPN model. We can now apply the *dam2gspn* and *ertms2gspn* transformations to this model, generating a complete GSPN model not fully represented for sake of space.

5.3 Temporary Speed Restriction Behaviour Testing

In order to be industrial appealing, the verification of functional requirements needs automatic test case generation. First steps concern with modeling of both system behaviour and property to be tested.

Fig. 10 models the behaviour of the RBC in presence of TSR. This model is based on state machines according to the VVP language. With respect to the functional requirement expressed in Subsection 5.1, when a TSR must be sent to EVC, RBC starts a timer and sends such kind of message until it does not receive an ack from EVC or three attempts has not been made. In the last case, an Unconditionally Emergency Stop message is sent to the EVC. A model of the verification of the requirement is represented in Fig. 11: an UML activity diagram is stereotyped with *Partial VerificationContext* and contains

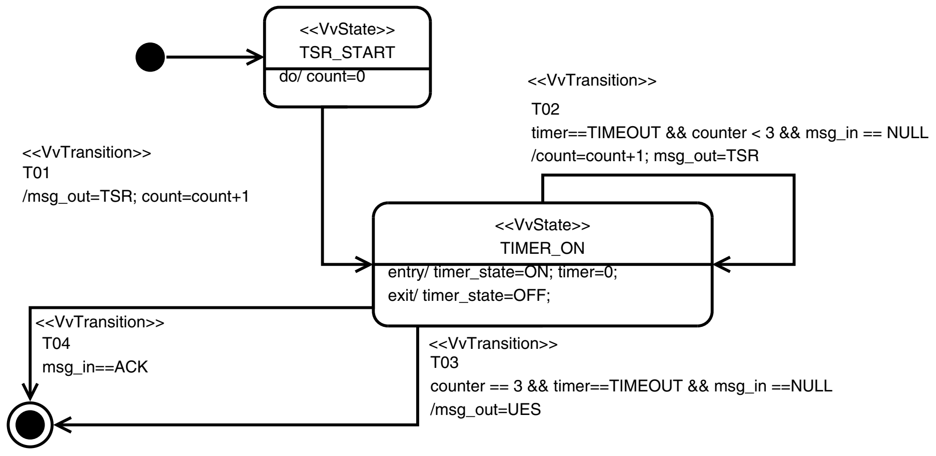
<<VvState>>
TSR_START
do/ count=0
<<VvTransition>>
T02
timer==TIMEOUT && counter < 3 && msg_in == NULL
/count=count+1; msg_out=TSR
<<VvTransition>>
T01
/msg_out=TSR; count=count+1
<<VvState>>
TIMER_ON
entry/ timer_state=ON; timer=0;
exit/ timer_state=OFF;
<<VvTransition>>
T04
msg_in==ACK
<<VvTransition>>
T03
counter == 3 && timer==TIMEOUT && msg_in ==NULL
/msg_out=UES

Fig. 10. Model of the RBC behaviour of TSR mechanism

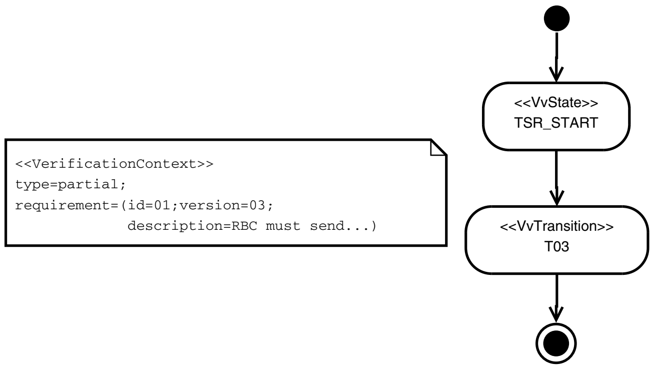

Fig. 11. Specification of TSR mechanism requirement

the necessary *TestStep*s to develop in a full specified *Full VerificationContext* (the object of the automatic generation).

In order to generate the related test case (full specification of input and output conditions), we use model checking technique. In particular we rely on SPIN model checker [13] and Promela language. After the definition of a metamodel for Promela language, both M2M and M2T tranformations can be defined starting from VVP. These transformations can be used to translate both the system behaviour and the requirement for which generate the test case; then the two parts are merged into a single Promela model and translated into Promela concrete syntax. Formal definition of the model transformations is out of the scope of the paper; notwithstanding we report a snippet of the Promela file generated by the TSR model.

```
...
:: (state == TIMER_ON) ->
  atomic {
    if
      :: (timer==TIME_OUT && count < 3 && msg_in == NULL) ->
        // exit - state
        timer_state = OFF;
        // applicable transition
        transition = T02;
        state = TIMER_ON;
        // transition action
        msg_out = TSR;
        count = count + 1;
        // entry - state
        timer_state = ON;
      :: (count == 3 && timer==TIME_OUT && msg_in ==NULL) ->
        // exit - state
        timer_state = OFF;
        // applicable transition
        transition = T03;
        state = END_STATE;
        // transition action
        msg_out = UES;
      :: (msg_in == ACK) ->
        // exit - state
        timer_state = OFF;
        // applicable transition
        transition = T04;
        state = END_STATE;
    fi;
  }
...
```

The code snippet is the result of the translation of the *TIMER_ON* state: it's possibile to see the three transitions that start from this state with trigger conditions as specified in the high level. For each transition a "case" statement is generated containing all the actions that must be accomplished: *state exiting activities*, *transition activation*, *new state entering* tasks.

If a specified test is feasible, SPIN finds a counterexample and a full detailed trace containing all the changes in Promela variables can be generated. This trace can be used to extract the sequence of states-transitions on the state machines inducing the sequence of inputs to give to system during the test execution phase.

6 Conclusions and Open Issues

This papers has presented a novel approach in Verification&Validation of critical railway systems that exploits the benefits of formal analysis and sofwtware/system testing. An important point is related to the capability of defined process to be both theoretically unifiying and be appealing in real industrial contexts. The methodology has been applied to the railway domain specifically addressing the two different aspects: two applications of ERTMS/ETCS signalling control systems show complementary features and advantages of the proposed approach. Indeed the applications show how to develop novel techincal and business oriented specific languages and mode transformations both improving language usability and transformation reuse. It's important to remark that this is part of an ongoing work and the VVP is currently in development phase: future research efforts will investigate on extends VVP in particural in the interaction between the analysis and testing subparts of the approach.

References

1. ARTEMIS-2008-1-100022 CHESS - composition with guarantees for high-integrity embedded components software assembly, `https://www.artemis-ju.eu/chess`
2. MBAT: Combined Model-based Analysis and Testing of Embedded Systems, `http://www.mbat-artemis.eu/`
3. PRIDE - ambiente di progettazione integrato per sistemi dependable, transformations for dependability analysis, deliverable 2.1 (February 2003)
4. Model driven architecture guide, Version 1.0.1, OMG document (2003)
5. Bernardi, S., Flammini, F., Marrone, S., Merseguer, J., Papa, C., Vittorini, V.: Model-Driven Availability Evaluation of Railway Control Systems. In: Flammini, F., Bologna, S., Vittorini, V. (eds.) SAFECOMP 2011. LNCS, vol. 6894, pp. 15–28. Springer, Heidelberg (2011)
6. Bernardi, S., Merseguer, J., Petriu, D.C.: A dependability profile within MARTE. Journal of Software and Systems Modeling (2009)
7. Bondavalli, A., Dal Cin, M., Latella, D., Majzik, I., Pataricza, A., Savoia, G.: Dependability analysis in the early phases of UML-based system design. Comput. Syst. Sci. Eng. 16(5), 265–275 (2001)
8. Causevic, A., Sundmark, D., Punnekkat, S.: An industrial survey on contemporary aspects of software testing. In: 2010 Third International Conference on Software Testing, Verification and Validation (ICST), pp. 393–401 (April 2010)
9. Clarke, E.M., Wing, J.M.: Formal methods: state of the art and future directions. ACM Comput. Surv. 28(4), 626–643 (1996)
10. Dai, Z.: Model-driven testing with UML 2.0. In: Proceedings of the 2nd European Workshop on Model Driven Architecture (2004)
11. Flammini, F.: Railway safety, reliability, and security: Technologies and systems engineering. IGI Global (2012)
12. Gómez-Martínez, E., Merseguer, J.: ArgoSPE: Model-Based Software Performance Engineering. In: Donatelli, S., Thiagarajan, P.S. (eds.) ICATPN 2006. LNCS, vol. 4024, pp. 401–410. Springer, Heidelberg (2006)
13. Holzmann, G.J.: The SPIN model checker (September 2003)
14. Hsueh, M.C., Tsai, T.K., Iyer, R.K.: Fault injection techniques and tools. Computer 30(4), 75–82 (1997)
15. Lagarde, F., et al.: Improving UML profile design practices by leveraging conceptual domain models. In: 22nd Int.l Conf. on Automated Software Engineering, Atlanta, USA, pp. 445–448. ACM (November 2007)

16. Marrone, S., Papa, C., Vittorini, V.: Multiformalism and Transformation Inheritance for Dependability Analysis of Critical Systems. In: Méry, D., Merz, S. (eds.) IFM 2010. LNCS, vol. 6396, pp. 215–228. Springer, Heidelberg (2010)
17. UML profile for modeling and analysis of real-time and embedded systems (marte), Version 1.0, OMG document (2009)
18. Moreno, G.A., Merson, P.: Model-driven performance analysis. In: Proceedings of the 4th International Conference on the Quality of Software Architectures, QoSA (2008)
19. UML testing profile, Version 1.1, OMG document (2012)
20. Petriu, D.B., Woodside, M.: A metamodel for generating performance models from UML designs. In: Proceedings of the 7th Int. Conference on the Unified Modeling Language. Modelling Languages and Applications, pp. 41–53 (2004)
21. Petriu, D.B., Woodside, M.: An intermediate metamodel with scenarios and resources for generating performance models from UML designs. In: Software and Systems Modeling, Special Issue, SoSyM, pp. 163–184 (2007)
22. Rugina, A., Kanoun, K., Kaâniche, M.: A system dependability modeling framework using AADL and GSPNs, pp. 14–38. Springer, Heidelberg (2007)
23. Selic, B.: A systematic approach to domain-specific language design using UML. In: 10th IEEE Int.l Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC 2007), pp. 2–9 (2007)
24. Tadano, K., Xiang, J., Kawato, M., Maeno, Y.: Automatic Synthesis of SRN Models from System Operation Templates for Availability Analysis. In: Flammini, F., Bologna, S., Vittorini, V. (eds.) SAFECOMP 2011. LNCS, vol. 6894, pp. 296–309. Springer, Heidelberg (2011)
25. UIC. ERTMS/ETCS class1 system requirements specification, ref. SUBSET-026, issue 2.2.2 (2002)
26. UNISIG. ERTMS/ETCS RAMS requirements specification, ref. 96s1266