Enabling the Usage of UML in the Verification of Railway Systems: the DAM-Rail Approach

S. Bernardi[a], F. Flammini[b], S. Marrone[c,*], N. Mazzocca[d], J. Merseguer[e], R. Nardone[d], V. Vittorini[d]

[a]*Centro Universitario de la Defensa Academia General Militar, Zaragoza, Spain*
[b]*AnsaldoSTS, Business Innovation Unit, Napoli, Italy*
[c]*Seconda Università di Napoli, Dip. di Matematica e Fisica, Caserta, Italy*
[d]*Università di Napoli "Federico II", DIETI, Napoli, Italy*
[e]*Universidad de Zaragoza, Dpto. de Informática e Ingeniería de Sistemas, Zaragoza, Spain*

Abstract

The need for integration of model-based verification into industrial processes has produced several attempts to define Model-Driven solutions implementing a unifying approach to system development. A recent trend is to implement tool chains supporting the developer both in the design phase and V&V activities. In this Model-Driven context, specific domains require proper modelling approaches, especially for what concerns RAM (Reliability, Availability, Maintainability) analysis and fulfillment of international standards. This paper specifically addresses the definition of a Model-Driven approach for the evaluation of RAM attributes in railway applications to automatically generate formal models. To this aim we extend the MARTE-DAM UML profile with concepts related to maintenance aspects and service degradation, and show that the MARTE-DAM framework can be successfully specialized for the railway domain. Model transformations are then defined to generate Repairable Fault Tree and Bayesian Network models from MARTE-DAM specifications. The whole process is applied to the railway domain in two different availability studies.

Keywords: Availability Analysis, Formal Models, Model-Driven Engineering, Railway Systems, RAM requirements, UML Profiles

*Contact Author
Email address: stefano.marrone@unina2.it (S. Marrone)

1. Introduction

Model-Based approaches and techniques are widely used in verification of complex systems in order to provide formal evidence of requirements. Nevertheless, their systematic application in industrial settings demands for proper model development processes to be integrated in the system development cycle leading to a semi-automated and unified approach to system development, verification, validation and maintenance. In recent years this need has produced several attempts to integrate model-based analysis and Model-Driven Development (MDD): they mainly focus on non-functional properties of systems, such as security [1], performance [2], dependability [3], reliability and availability [4], QoS and its relationship with the other properties [5]. These works demonstrated that the Model-Driven paradigm is very appealing in industrial settings and it may be applied to many different contexts or re-interpreted to support the production of hardware and hybrid systems. However there is still need of works that address specific domains, such as railway systems, especially for what concerns RAM (Reliability, Availability, Maintainability) analysis and the fulfillment of international standards.

This paper specifically addresses the definition of a Model-Driven approach to the evaluation of RAM attributes for railway applications. The approach is based on the usage of UML profiles at the conceptual representation level of the system for the automatic generation of formal models. The paper is partially based on the work described in [6], here we define a general process and introduce DAM-Rail, a Domain-Specific Modelling Language (DSML) [7] for railways modelling. DAM-Rail is derived from the MARTE-DAM profile [8] in order to provide railway modellers with an adequate support to domain concepts allowing on one hand, to reuse the tool chains and workbenches of the base languages (UML and MARTE) and access to a well established set of development practices, and on the other hand to use more direct and familiar expression of domain specific concepts on which a proper Graphical User Interface (GUI) may be easily realized. In doing that, we extend the MARTE-DAM profile with some concepts related to maintenance aspects and service degradation and show that the MARTE-DAM framework can be successfully extended to the railway domain (as well as several other domains). Model-to-Model transformation chains are defined to translate DAM/DAM-Rail specifications into formal models. Here we focus on availability and maintenance attributes, hence the Repairable Fault Tree [9] and the Bayesian Network [10] modelling formalisms are used. The overall process is then applied to two different railway systems in order to show its flexibility. With respect to the work presented in [6], a further case study is also described.

The paper is organized as follows: in Section 2 we briefly introduce the reader to the high-level languages (UML, MARTE, DAM) used in the paper. In Section 3 the overall process is presented, DAM extensions and DAM-Rail are defined. The transformation chains are introduced in Section 4. In Section 5 we apply the DAM-Rail approach to the availability analysis of a modern railway controller and a driverless metro vehicle, according to the ERTMS/ETCS [11] and CENELEC [12] standards respectively. Section 6 closes the paper by pointing out challenges and open issues to be addressed by ongoing work.

2. Background

The Unified Modeling Language (UML) [13] is a well known general purpose standardized modelling language for software system specification. The system structure is typically specified by a Component Diagram and/or a Class Diagram. The behavioural view of the system is instead specified using Use Cases, Activity Diagrams, Sequence Diagrams and State Machines, or a combination of them.

UML is a semi-formal specification language: the semantics of UML diagrams is expressed in natural language while the abstract syntax is provided in terms of UML meta-models. A UML meta-model is actually a Class Diagram that represents the UML concepts and their relationships as meta-classes and meta-associations, respectively. Constraints on the meta-classes and meta-associations are expressed with OCL [14].

UML is also equipped with a *profiling* mechanism that allows to customize UML for a particular domain or platform. The UML profiling is actually a *lightweight* meta-modelling technique to extend UML, since the standard semantics of UML model elements can be refined in a strictly additive manner.

Stereotypes, tags and OCL constraints are the extension mechanisms used to define a UML profile. In particular, a stereotype extends one or more UML meta-classes and can be applied to those UML model elements that are instantiations of the extended meta-classes. For example, in Figure 1, the *Resource* stereotype extends the *Class* meta-class, then the former can be applied to a class in a Class Diagram. Just like classes, a stereotype can have properties which are referred as tag definitions: in the previous example *resMult* and *isProtected* are tags. When a stereotype is applied to a model element, the value assigned to a stereotype property is called *tagged-value*.

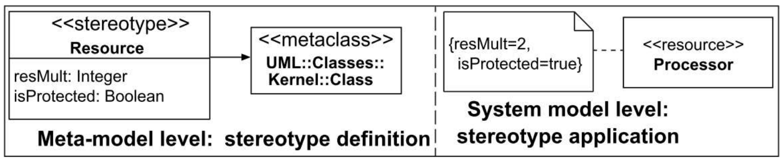


Figure 1: UML stereotypes and their application.

2.1. MARTE and DAM profiles

The *UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE)* [15] is an OMG-standard profile that customizes UML for the modelling and analysis of non-functional properties (NFP) of real-time embedded systems, such as timing or performance-related properties. MARTE provides a general analysis framework called General Quantitative Analysis Model (GQAM) sub-profile. Although analysis domains have different terminology, concepts, and semantics, they also share some foundational concepts which are expressed in GQAM. A specific analysis domain can be defined through specialization of GQAM; concretely, MARTE addresses the schedulability analysis and the performance analysis by defining two separated sub-profiles that specialize GQAM.

A key feature of MARTE is the NFP framework and its Value Specification Language (VSL). The former is used to define new NFP data-types that are necessary to the definition of a specific analysis domain sub-profile. In particular, an NFP data-type is characterized by several properties, such as the origin which allows the modeller to specify whether an NFP is a requirement or a metric to be estimated, the type of statistical measure associated to an NFP (e.g., mean), the type of the quality order relation in the value domain of an NFP, for comparative analysis purposes. Instead, the VSL enables the specification, at model level, of tagged-values according to a well-defined syntax.

The *Dependability Analysis and Modeling (DAM)* [8] profile is a specialization of MARTE-GQAM to enable dependability analysis. DAM is conceived by following a systematic approach, which consists of two main steps:

first, a dependability *domain model* is built; later, the model is mapped onto UML extensions (i.e., stereotypes, tags and OCL constraints).

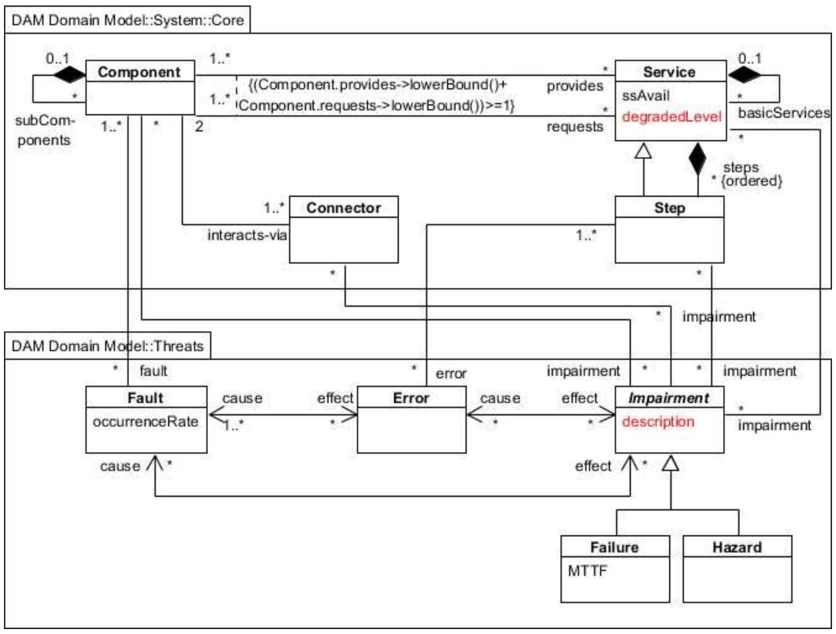


Figure 2: Excerpt of the System Core and Threats domain models.

The DAM domain model is a set of UML Class Diagrams, structured into packages, that represent the main dependability concepts from the literature, according to a component-based view of the system to be analysed [16]. Figure 2 shows an excerpt of the System Core and Threats domain models. The system consists of a set of *components* bound together through *connectors*, in order to interact. It delivers a set of high-level *services*, that can be detailed - at a finer grained level - by a sequence of *steps*, representing states of components, events or actions. The system can be affected by threats, i.e., *faults, errors, impairments* (e.g *failures*) according to the definition given in [16].

The domain model also includes redundancy and maintenance concepts. A hw/sw *redundant structure*, shown in Figure 3, increases the system fault tolerance (FT) and is made of components, among them *FT components*. Maintenance is related to repairable systems: the Maintenance domain model (Figure 4, white classes) includes concepts that are necessary to support the evaluation of system availability, that is the *maintenance actions* undertaken

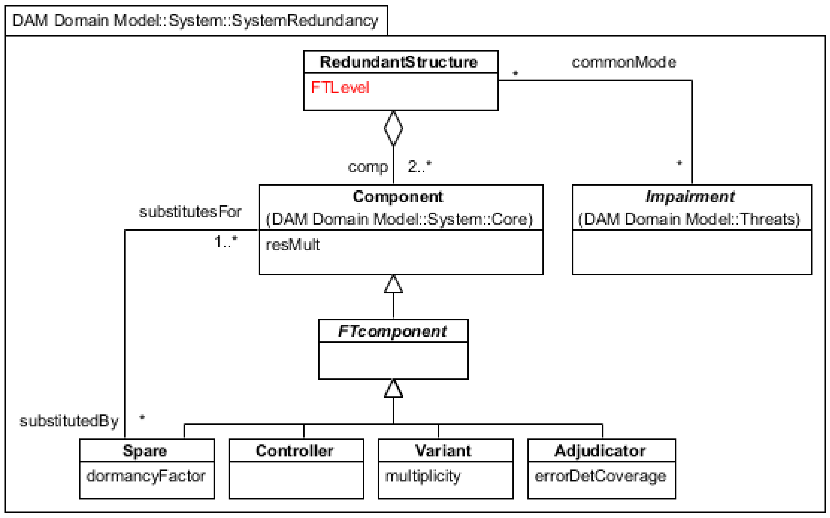


Figure 3: System Redundancy domain model.

to restore the system affected by threats. According to [16], *repairs* of system components involve the participation of external agents (e.g., repairman, test equipment, etc.), while *recovery* of services does not require the intervention of the latter.

The DAM profile has been defined by mapping, systematically, the concepts of the DAM domain model to UML and MARTE. It includes a set of DAM extensions (stereotypes and tags) and a DAM library containing dependability specific types. According to [7], each domain class was examined, together with its attributes, associations and constraints, to identify the most suitable UML base concepts for it. Finally, only a subset of the domain classes were mapped to stereotypes and the rest of the classes were mapped to data-types (e.g., fault, error and failure classes). The attributes and associations of such domain classes were mapped to tags, following the guidelines in [17]. Table 1 shows an excerpt of the DAM stereotypes, which were mapped from the homonymous classes of the *System Redundancy* (Figure 3) and *Maintenance* (Figure 4) domain models.

3. Modelling process

This Section aims at defining the Model-Driven process enabling the verification of railway systems. One of the activities of the whole process is the definition of DAM-Rail, a DSML for modelling and verification of RAM

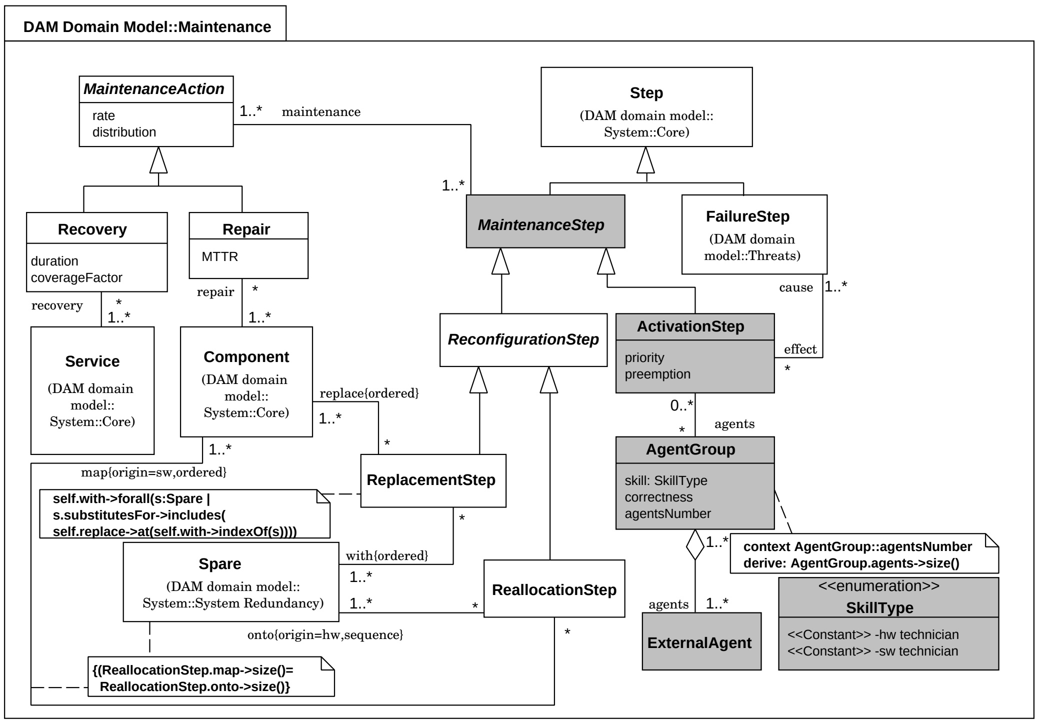


Figure 4: Maintenance domain model.

properties in railway systems. Instead of creating DAM-Rail from scratch, we decided to extend UML in the form of a new *profile*. Therefore, we gained for DAM-Rail the general modelling capabilities UML provides. We also needed dependability modelling capabilities, hence we selected DAM and defined DAM-Rail as a specialization of DAM. On the other hand, we realized that the extension of DAM with few dependability concepts will increase its modelling capabilities for different domains, also for railway systems. The Model-Driven process adopted in this paper is depicted by the UML Activity Diagram in Figure 5. The diagram presents several activities that are related to the phases described in the following of this paper. The process starts with three parallel activities: 1) extending DAM, in order to introduce service degradation modelling capabilities as well as some concepts related to maintenance; 2) defining the meta-models which describe the main elements and relationships on which the formalisms (used to build the formal models) are based on, and 3) UML system modelling. The activities in Figure 5 are grouped in three swimlanes that are related to three process stakehold-

Table 1: DAM stereotypes and tags.

Stereotype	*Inherits / Extends*	*Tags: type*
System Redundancy		
DaAdjudicator	DAM::DaComponent / -	errDetCoverage: NFP_Percentage[*]
DaController	DAM::DaComponent / -	*none*
DaRedundantStructure	- / UML::Package	commonModeF: DaFailure[*] commonModeH: DaHazard[*]
DaSpare	DAM::DaComponent / -	dormancyFactor: NFP_Real[*] substitutesFor: DaComponent[1..*]
DaVariant	DAM::DaComponent / -	multiplicity: NFP_Integer[*]
Maintenance		
DaReallocationStep	DAM::DaStep / -	kind: {reallocation} map: DaComponent[1..*] onto: DaSpare[1..*]
DaReplacementStep	DAM::DaStep / -	kind: {replacement} replace: DaComponent[1..*] with: DaSpare[1..*]

ers respectively: *Meta-Model Developer* (bottom), *Transformations Designer* (middle) and *System Modeller* (top).

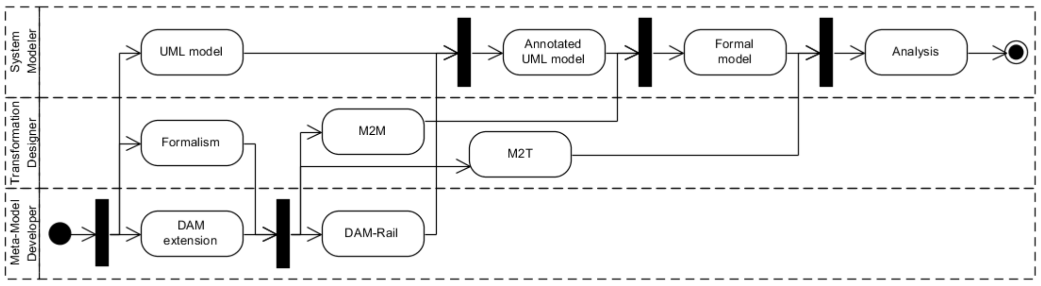


Figure 5: Process schema.

Model-to-Model (M2M) transformations are implemented by defining the proper rule sets to translate DAM/DAM-Rail specifications into availability models, where DAM and DAM-Rail domain models act as source meta-models for the transformations, and formalisms meta-models act as target meta-models. Model-to-Text (M2T) transformations are also provided, they use the formalism meta-models to generate the proper input files for the analysis tools (e.g. JavaBayes [18] for Bayesian Networks, Sharpe [19] for Fault Trees, GreatSPN [20] for Generalized Stochastic Petri Nets). The UML models of the system are annotated with DAM and DAM-Rail stereotypes and the M2M transformations are applied to generate the formal models to be solved in the analysis phase. Of course, some activities are accomplished just once (DAM extension, DAM-Rail definition), whereas the activities involving specific railway applications or the definition of different transformation chains must be performed on each new study. DAM extension and DAM-Rail are presented here below, in Subsection 3.1 and Subsection 3.2 respectively. Model transformations are briefly described in Section 4, where DAM to Repairable Fault Trees and DAM to Bayesian Networks transformations are specifically addressed.

3.1. DAM extension

This Section shows several extensions of DAM. The primary one introduces new concepts in the Maintenance domain model, which was described in Section 2 as it was initially proposed by DAM. This extension is targeted to obtain augmented maintenance models in terms of Repairable Fault Trees. The other extensions are minor, we realized their need while developing DAM-Rail, but we considered them useful also for other domains beyond railway systems, so we included them at DAM level.

The primary extension is proposed by the grey classes in Figure 4. We have introduced an abstract *MaintenanceStep* class that is associated to the actual *MaintenanceActions* this step performs. Maintenance steps can be *ReconfigurationSteps*, already present in DAM, or *ActivationSteps*. The latter is introduced to model maintenance actions that are initiated as a consequence of component failures, therefore the activation step is related to the failures that can trigger it. An activation step defines its *priority* as well as a *preemption* policy: the priority is an integer that allows for choosing the maintenance policy to be executed in case of conflicts depending on the failures occurred in the system, while an activation step can be preempted by another activation step with a higher priority according to its preemption policy (*preemption* = *true*). Moreover, an activation step relates to a group of agents with the required *skills* to perform the step. The *AgentGroup* class also defines the *correctness* that is the probability to have a successful maintenance and number of agents needed to carry out the step.

Minor extensions mainly consist in introducing new attributes to existing classes of the DAM domain model: *degradedLevel* (*Service* class of Core model in Figure 2), *FTlevel* (*RedundantStructure* class of System Redundancy model in Figure 3), and *description* (*Impairment* class of Threats model in Figure 2). The *degradedLevel* defines whether the service is working to its full capacity or not. The *FTlevel* specifies the minimum number of components, within a redundant structure, required to still guarantee a service. Finally, the *description* allows the modeller to provide a textual explanation of the impairment (i.e., failure or hazard).

The aforementioned extensions have provided new elements in the DAM profile (i.e., stereotypes and tags), which have been defined following the guidelines commented in Section 2. They are summarized in Table 2. The new attributes are mapped into tags: then, the *degradedLevel* and the *FTlevel* are converted into tags of the already existing stereotypes *DaService* and *DaRedundantStructure*, respectively. Both tags are of MARTE_NFP types. On the other hand, the *description* attribute is mapped onto tags of the concrete complex dependability data-types *Failure* and *Hazard*.

From the new four classes in the Maintenance domain model we have created two stereotypes: *DaAgentGroup* and *DaActivationStep*. The former directly converts its three attributes into three tags, the type of the *skill* tag is an enumeration (*skillType*). The latter is somehow more complicated: the attribute *kind* is inherited from *DaStep*; *cause* maps the association for defining the failures triggering the activation; finally, *agents* maps the other association in the domain modell defining the agents involved.

Table 2: New DAM extensions.

Stereotype	***Inherits / Extends***	***Tags: type***
	System::Core	
DaService	MARTE::GQAM::GaScenario / -	degradedLevel: NFP_Real[*]
	System::Redundancy	
DaRedundantStructure	- / UML::Package	FTlevel: NFP_Integer[*]
	Maintenance	
DaAgentGroup	- / UML::Classifier (e.g., Actor, Class)	skill: skillType correctness: NFP_Real[*] agentNumber: NFP_Integer[*]
DaActivationStep	DAM::DaStep / -	kind:{activation} preemption:NFP_Boolean[0..1] cause: DaStep[1..*] = (kind=failure) agents: DaAgentGroup[1..*]
Datatypes	***Is a***	***Tags: type***
	Threats	
DaFailure	MARTE::VSL::DataType	description: String[0..1]
DaHazard	MARTE::VSL::DataType	description: String[0..1]

Tables 1 and 2 solve the issue of which model elements in a UML diagram can be stereotyped (e.g., a state in a State Machine Diagram or a component in a Component Diagram). In the second column of these tables each stereotype may either specialize a previously defined MARTE-DAM stereotype or directly extend a UML meta-class. For example, *DaAdjudicator, DaController, DaSpare* and *DaVariant* specialize *DAM::DaComponent*, then they can be applied to UML elements representing system software and hardware resources (e.g., classes, instances, components, nodes) because *DAM::DaComponent* can do it. On the other hand, the different *step* stereotypes (e.g., *DaReallocation*, *DaReplacement*, *DaActivation*) inherit from *DaStep*, which can be applied to a wide set of behaviour-related elements, such as messages in Sequence Diagrams, and transitions, state, trigger events, effect actions in State Machine Diagrams. Finally, the stereotypes *DaRedundantStructure* and *DaAgentGroup* directly extend the *Package* and *Classifier* UML meta-classes, respectively. While the former can be applied to package elements, the latter can be applied to different kind of structure-related elements, such as actors in Use Case Diagrams and classes in Class Diagrams.

3.2. DAM-Rail

DAM-Rail adds railway concepts to MARTE-DAM. Its meta-model is organized as depicted in Figure 6. It consists of two main packages: *System* and *O&M*. *System* describes the structural features of the railway domain, *O&M* deals with the modelling of the railway operation and maintenance. The *System* package is, in turn, structured in several packages, each one addressing a specific technology. According to a classical railway organization, we identify eight sub-packages: (1) *RollingStock* includes concepts and their attributes for a complete vehicle description (e.g. vehicle length and engine power); (2) *Track* contains concepts related to the infrastructure as well as track elements (e.g. segment, crossing and point); (3) *Signalling* encompasses concepts of the signalling system (e.g. fixed and moving block); (4) *TP* includes concepts related to traction power, (e.g. traction transformer, catenary and third rail); (5) *PS* contains concepts related to power supply (e.g. station cabinet and medium-voltage converter); (6) *SCADA* deals with supervisory control and data acquisition modelling (e.g SCADA rack and server); (7) *PSD* includes concepts related to platform screen doors subsystem (e.g doors engine and screen); (8) *TLC* contains concepts related to telecommunications technologies (e.g. terrestrial trunked radio and on-board system).

Hence a single package can be considered a “technological” DSML itself. As an example, an excerpt of the *RollingStock* package is shown in Figure 7: it describes the portion of domain concepts that will be used in Section 5 to model the metro driverless case study.

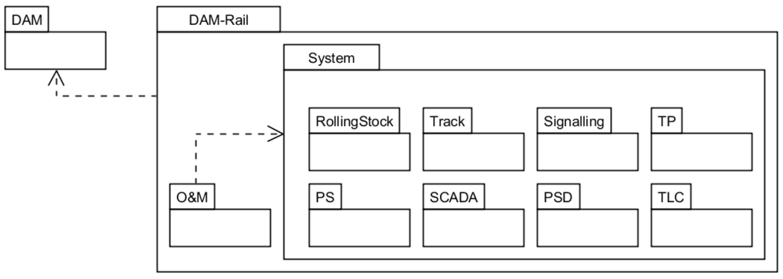


Figure 6: DAM-Rail package structure.

The main concept included in this package is the *Vehicle*, which is characterized by the number of cars, the length and other attributes (not shown in the Figure). The *RollingStock* package also contains concepts related to the railway vehicle sub-components: *Pantograph*, *Engine*, *Wheel*, *AirCompressor* and *GasTank*. Each of them extends the *Component* class of DAM domain model, by adding some specific features: i.e., the maximum distance between vehicle and catenary for the pantograph; the power for the engine; the diameter and the border thickness for the wheel; the pressure for the air compressors; the power number (measured in British Thermal Units - BTU) for the gas tank. The description of the other sub-packages of *System* is out of the scope of this paper.

O&M deals with Operation and Maintenance, focusing primarily on the features to be considered in the evaluation of railway services. It is important to consider that, in the railway domain, a common approach defines degradation categories for each service. Figure 7 depicts an excerpt of this package including the degradation categories (Service Mode - SM) related to the services provided by vehicles. *DelaySM* may be used to specify the expected minimum and maximum delays at the end of the trip, *SpeedSM* may be used to specify the maximum speed (ratio) reached by the vehicle, *CoolnessSM* may be used to specify the refrigeration ratio inside a car and *PowerCaptureSM* may be used to specify the power capture capability ratio of a vehicle from the electrified line. Notice that, by inheritance, the relationships between components and services are derived from DAM, as well as their threats characterization. Similarly, the impact of impairments on services is already fully modelled in DAM and it does not require to be specialized in DAM-Rail.

The DAM-Rail packages are directly mapped into the DAM-Rail pro-

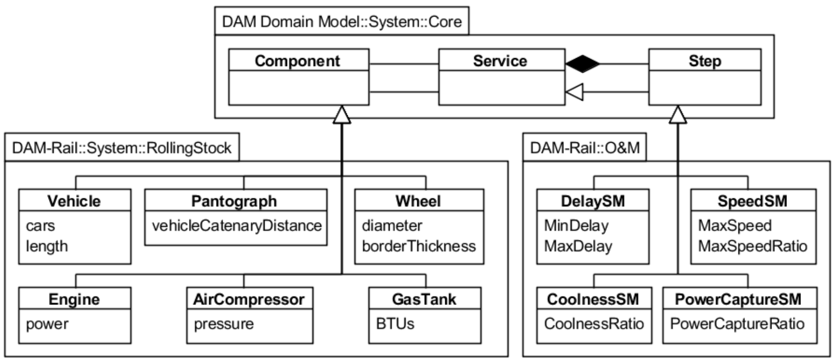


Figure 7: DAM-Rail *RollingStock* and *O&M* packages (portion).

file. The concepts introduced by *RollingStock* and *O&M* are mapped one to one into stereotypes: the classes inherit from *DAM::DaComponent* and *DAM::DaStep* stereotypes respectively. For example, *Engine* belonging to the DAM-Rail meta-model is translated into the *Engine* stereotype in the DAM-Rail profile inheriting from *DAM::DaComponent*, as well as *DelaySM* is translated into the *DelaySM* stereotype inheriting from *DAM::DaStep*. In Table 3 a partial list of the DAM-Rail stereotypes (those shown in Figure 7) and their attributes is reported.

Table 3: DAM-Rail stereotypes and tags (portion).

Stereotype	***Inherits / Extends***	***Tags: type***
System::RollingStock		
Vehicle	DAM::DaComponent / -	cars: Integer[0..1] length: NFP_Length[0..1]
Pantograph	DAM::DaComponent / -	vehicleCatenaryDistance: NFP_Length[0..1]
Engine	DAM::DaComponent / -	power: NFP_Power[0..1]
Wheel	DAM::DaComponent / -	diameter: NFP_Length[0..1] borderThickness: NFP_Length[0..1]
AirCompressor	DAM::DaComponent / -	pressure: NFP_Real[0..1]
GasTank	DAM::DaComponent / -	BTUs: Integer[0..1]
O&M		
DelaySM	DAM::DaStep / -	MinDelay: NFP_Duration[0..1] MaxDelay: NFP_Duration[0..1]
SpeedSM	DAM::DaStep / -	MaxSpeed: NFP_Real[0..1] MaxSpeedRatio: NFP_Percentage[0..1]
CoolnessSM	DAM::DaStep / -	CoolnessRatio: NFP_Percentage[0..1]
PowerCaptureSM	DAM::DaStep / -	PowerCaptureRatio: NFP_Percentage[0..1]

The ultimate goal of the DAM-Rail profile is to provide railway modellers with specific concepts, close to their experience and skills, that cannot be represented at DAM level. Of course, DAM stereotypes must still be used to model general structures; for example the railway controller in Section 5.1 is completely modelled by using DAM stereotypes, since involved concepts are generally enough to be stereotyped with *DAM::DaComponent*, *DAM::DaRedundantStructure*, *DAM::DaSpare*, etc., neither the required analysis refers to specific domain features. On the contrary, the metro driverless vehicle model, in Section 5.2, makes use of both DAM and DAM-Rail stereotypes since, in this case, the focus is on railways service degradation categories.

4. Model transformations

MARTE-DAM and DAM-Rail models may be automatically translated into formal models: this translation is performed by means of M2M transformations; in addition, according to the process described in Section 3, some M2T transformations are needed in order to serialize the formal models into a concrete syntax, specific of the analysis tools. M2M transformations are defined on source and target meta-models. Here the source meta-model is MARTE-DAM, whereas we choose Repairable Fault Trees (RFTs) and Bayesian Networks (BNs) as destination formalisms. Let us define the meta-models for both RFTs and BNs. RFTs have been introduced to allow complex repair policies modelling and evaluation [9]. RFTs integrate Generalized Stochastic Petri Nets (GSPNs) and Fault Trees (FTs): repair policies are represented by nodes, called Repair Boxes (RBs), which encapsulate a GSPN. The RBs are connected to a FT which describes the faults that may happen and their contribution to the occurrence of a failure. A RB is applied to an event and models a repair action that can be performed on the related system component. A simplified RFT meta-model is shown in Figure 8.

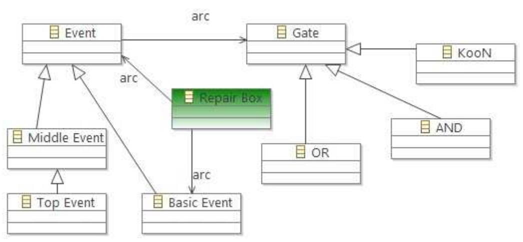


Figure 8: The RFT simplified meta-model

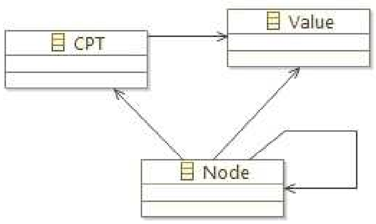


Figure 9: The BN simplified meta-model

BNs allow to easily model common modes of failure and multi-state systems [10]. A BN is a probabilistic graphical model which is used to represent a set of random variables and their conditional dependencies through a Directed Acyclic Graph. Each node in the graph represents a random variable and the edges between nodes represent probabilistic dependencies among the corresponding random variables. The conditional dependencies in the graph is captured by Conditional Probability Tables (CPTs) and may be estimated by well known statistical and computational methods. A simplified BN meta-model is shown in Figure 9.

One of the most critical issue in the application of transformational processes to critical systems is the correctness assessment of the transformations themselves. Different approaches have been proposed in scientific literature spanning from testing [21] to formal verification [22] but the problem still remains an open issue. This work does not focus on transformation verification: the transformations here introduced have been tested on some significant examples and then applied to the case studies. Future research effort will be spent on transformation assessment.

4.1. DAM2RFT transformation

DAM2RFT generates RFTs from UML models stereotyped with DAM. It exploits the strict relationship existing between FTs and RFTs and the dependency relations existing among the DAM packages. Hence, we adopt a *divide-et-impera* approach which exploits transformation composition and reuse, by applying **module superimposition** [23], a widespread mechanism well supported by the most important transformation languages, including Atlas Transformation Language (ATL) [24]. In practice, superimposition allows for defining a new transformation from existing ones by the union of their set of rules. Hence, first a transformation from DAM to FT (*dam2ft*) is realized with the aim to generate the Fault Tree structure of the RFT model from the System and Threats domain models; then the proper rule set for the generation of Repair Boxes and related arcs from the Maintenance domain model is defined and *dam2rft* is obtained by superimposition.

In details, *dam2ft* works as follows: the Top Event is associated with the failure of the system which provides the service (specified by the Use Case Diagram). Events and gates are created from the Component Diagram by recursively applying the following rules:

1. for each *DaComponent*: if *fault* is null, one Middle Event is created;

however, if *fault* is not null, one Middle Event and m Basic Events are created where m is the resource multiplicity (*resMult*);

2. for each *DaSpare*: if *fault* is not null, m Basic Events are created where m is the resource multiplicity (*resMult*);
3. for each Middle Event (created from a *DaComponent*) a Gate is created and arc to the Middle Event is set:
    - an OR gate if *DaComponent* does not belong to a *DaRedundantStructure*;
    - an AND gate if *DaComponent* belongs to a *DaRedundantStructure* and $FTlevel = 1$;
    - a KooN gate if *DaComponent* belongs to a *DaRedundantStructure* and $FTlevel = k$, $k > 1$;
4. when a sub-component relationship exists between a *DaComponent* X and a contained *DaComponent* Y and Y contains other elements, an arc is created from the Middle Event generated by Y to the Gate that is input of the Middle Event generated by X;
5. when a sub-component relationship exists between a *DaComponent* X and a contained *DaComponent*/*DaSpare* Y and Y contains no other element, an arc is created from the Basic Event generated by Y to the Gate that is input of the Middle Event generated by X.

The transformation implemented by *dam2rft* works under the hypothesis that the DAM specification includes a model of the repairing process of a sub-component and information about its steps (this can be expressed by means of a State Machine Diagram, an Activity Diagram, or a Sequence Diagram). The existence of a repair model associated to a *DaComponent* is annotated by a *DaActivationStep* stereotyped element. First an RB is generated for each replica of *DaComponent* and of its *DaSpare*, if any. The RB is filled with information about the MTTR and the resources (the repairmen) needed to accomplish the activity. This information is retrieved by the diagram used to describe the repair dynamics and by navigating the Component Diagram.

Finally, each RB is connected to the Fault Tree generated by *dam2ft* through repair arcs between: 1) RB and its triggering event, i.e. the Middle Event or Basic Event from which the RB has been generated (the sub-system to be repaired); 2) RB and all the Basic Events that are present in the sub-tree whose root is the RB triggering event. Once the RFT model has been generated, a GSPN model is derived by applying another M2M transformation in order to allow easy analysis of this formal model. The description of this last transformation is reported in [25].

4.2. DAM2BN transformation

DAM2BN generates BNs from UML models stereotyped with DAM under the hypothesis that the DAM specification includes, beyond a Use Case and a Component Diagram, also a behavioural model of the service, and specifically a State Machine Diagram modelling the state transition caused by classes of failures. In details, *dam2bn* works according to the following rules:

1. for each *DaComponent* one BN node is created; the values of the random variable associated with the node are the elements of the *DaComponent*'s *failure* array plus one (the nominal service mode); the Conditional Probability Table (CPT) is built according to its tagged values:
    - if a *failure* element has a not null *occurrenceRate*, this rate is used to determine the probability associated with the value of the node;
    - if a *failure* element has a not null *condition*, this condition defines the truth table used to determine the CPT;
2. for each *DaRedundantStructure* one BN node is created; the associated random variable has just two values corresponding to *nominal* and *failure* service modes and its CPT is built according to the rules defined in [26]. An arc is added from the node corresponding to the *DaRedundantStructure* and the node corresponding to the *DaComponent* which contains the *DaRedundantStructure*;
3. for each *DaService* one BN node is generated; the values of the random variable values and/or the CPT associated with the node are calculated from the states in the State Machine Diagram associated with the *DaService*;
4. if a *usedResources* tagged value of *DaService* is assigned, it is used to determine the BN nodes that must be connected to the one BN node translating the *DaService*; then proper arcs between such nodes are added;
5. if a UML dependency between two *DaService*s is found, it is used to connect their associated BN nodes;

6. for each *DaRedundantStructure*, the CPT of the related BN node is determined (in analogy to what specified at point 1.b) from the *condition* tagged-value in the *DaStep*s contained in the State Machine.

4.3. Model-to-Text transformation

M2T transformations are necessary to finalize the M2M transformation chains in order to allow the usage of existing solvers. In this work, two M2Ts are necessary: the former transforms GSPN models into GreatSPN [20] input file format, the latter generates JavaBayes [18] compatible files from a BN model. The two transformations are implemented by means of ATL queries; they are not further described for sake of space.

5. Modelling railway systems

In order to demonstrate the DAM-Rail approach, in this Section two railway systems are modelled: the Radio Block Centre (RBC), that is a railway controller, and a driverless metro vehicle. These two examples address different aspects of availability analysis in railway industrial settings: RBC availability analysis is conducted to provide formal evidence that it meets the requirements dictated by international standards; vehicle availability analysis is conducted in order to formally state the effect that different failures categories have on the performability attribute of a metro system.

The former availability study is carried out at the component level and the contribution to the unavailability of the overall RBC has an upper bound fixed by the standards. Here the focus is on the impact of different maintenance policies on the component availability, that is an important and neglected issue. The latter availability study is carried out at subsystem level and the focus is on the impact that vehicle unavailability has on the metro service whose performance requirements may vary, according to the specific project or tender to be answered. In both cases, the goal of this Section is to show how the approach supports the verification process and the automated construction of availability models starting from the high-level model expressed by means of DAM and DAM-Rail profiles. These two different examples better place the approach in the railway context and show its potentialities.

5.1. Radio Block Centre

The Radio Block Centre (RBC) is the vital core of the European Railway Traffic Management System / European Train Control System (ERTMS /

ETCS) which is the reference standard of the European railway signalling and control systems [27] ensuring the safe running of trains on different European railway networks. RBC is a computing system which controls the movements of the set of trains travelling across the track area under its supervision. To this aim, RBC elaborates messages to be sent to the trains on the basis of the information received from external track-side systems and exchanged with on-board sub-systems. The unavailability of RBC is critical, as there is no way for the signalling system to work without its contribution. In case of RBC failure, all the vehicles under its supervision are compelled to brake and proceed in a staff responsible mode. This would lead to the most critical among the ERTMS/ETCS safe failures, that is the so called Immobilising Failure (IF). The ERMTS/ETCS standard requires compliance with the RAMS requirements [11] whose fulfillment has to be properly demonstrated. Specifically, the quantifiable contribution of the RBC system to operational unavailability must be not more than 10^{-6} (see [11], §2.3.3). The standards do not impose constraints on the system architecture so that different implementations are possible. In order to comply with the target, a reference architecture of the RBC must exhibit a high level of redundancy to improve the fault tolerance of the system. The system consists of three commercial CPU-RAM cards and a redundant FPGA based voter in a Triple Modular Redundancy (TMR) configuration. The GSM-R and WAN communication sub-systems are also chosen as COTS (Commercial Off The Shelf). The considered RBC configuration is completed by three commercial power supplies and a redundant standard backbone (used as system BUS). Maintenance policies are a fundamental aspect of RBC life-cycle for their impact on system availability. ERTMS/ ETCS gives no restrictive requirements for the maintainability parameters and this leaves much freedom in designing repair policies. Of course, it must be proved that the system still meets availability requirement.

5.1.1. RBC DAM model

The DAM specification of RBC used to generate the RFT consists of a Use Case Diagram, a Component Diagram and a set of State Machine Diagrams. The diagrams are annotated with the DAM extensions introduced in Section 3.1. The Use Case Diagram depicted in Figure 10 represents the main functionality of the RBC: the *train outdistancing*. The use case is stereotyped *DaService* to explicitly indicate (by means of the *usedResources* tagged value) which is the component in charge of providing the service, hence identifying the source of the failures that may cause a service interruption. The availability requirement is captured by the *ssAvail* tagged-value. The actor stereotyped *DaAgentGroup* represents the set of hardware technicians (*skill* tagged-value) who participate in the repair process; here two technicians (*agentsNumber*) are assumed to accomplish repair activities correctly (*correctness*).

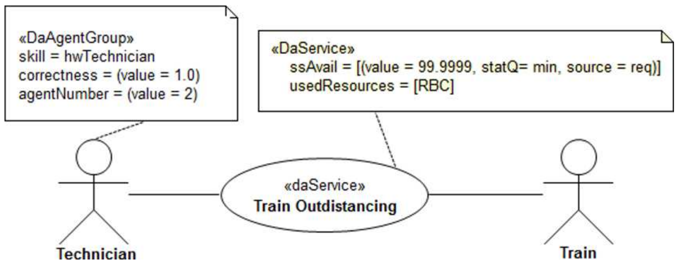


Figure 10: Train Outdistancing Use Case.

The Component Diagram in Figure 11 provides a high level description of the RBC components whose failures affect the system unavailability. The main hardware components of the RBC system are stereotyped *DaComponent*: they can be either simple components (e.g., *MainBus*) or components with an internal structure (e.g., *TMR*). Each redundant sub-system is represented by a package stereotyped *DaRedundantStructure* (e.g., *SystemBus*) which includes several instances of the same hardware component: the *DaComponents* are the active replicas (e.g., *mainBus*), whereas the other components are stereotyped *DaSpare* (e.g., *spareBus*).

A detailed view of the *SystemBus* redundant sub-system is shown in Figure 12, where several tagged-values associated with the stereotyped elements have been specified: the *DaRedundantStructure* requires at least one operative component, either main or spare one, to guarantee the *SystemBus* functionality (*FTlevel* tagged-value); the *SystemBus* includes one *DaComponent* bus instance and one *DaSpare* bus instance (*resMult* tagged-values); the *DaSpare* bus substitutes for the main bus, in case of failure of the latter (*substituteFor* tagged-value); both the main and the spare buses are characterized by the fault occurrence rate (*fault.occurrenceRate*) and the Mean Time To Repair (*MTTR*) values.

Finally, the RBC specification includes several State Machine Diagrams (SM), one for each repairable component. The SM models the dynamics of the repair process of a specific component. In the RBC system, three

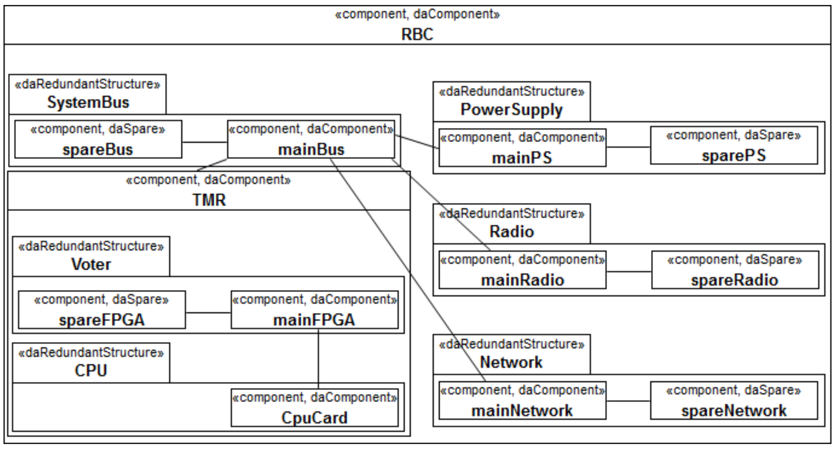


Figure 11: RBC Component Diagram.

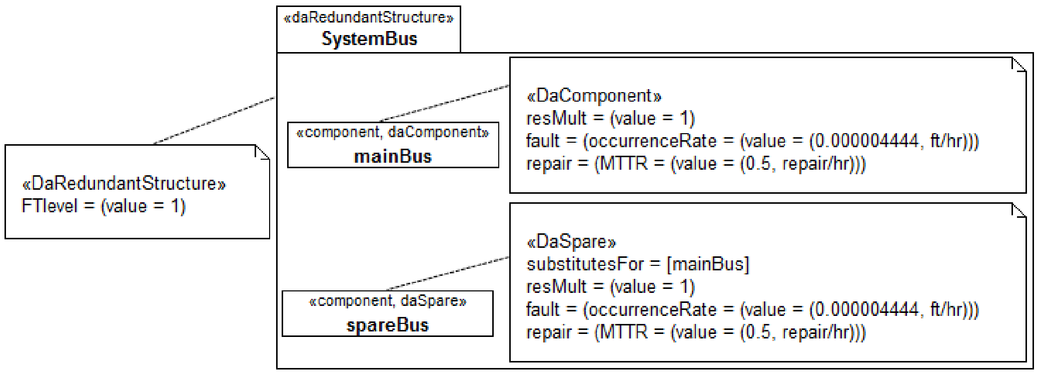

«daRedundantStructure»
SystemBus
«DaRedundantStructure»
FTlevel = (value = 1)
«component, daComponent»
mainBus
«DaComponent»
resMult = (value = 1)
fault = (occurrenceRate = (value = (0.000004444, ft/hr)))
repair = (MTTR = (value = (0.5, repair/hr)))
«component, daSpare»
spareBus
«DaSpare»
substitutesFor = [mainBus]
resMult = (value = 1)
fault = (occurrenceRate = (value = (0.000004444, ft/hr)))
repair = (MTTR = (value = (0.5, repair/hr)))

Figure 12: System Bus.

components are equipped with a diagnostic mechanism (specifically, *RBC*, *mainPS* and *CpuCard*); when one of the latter fails, a repair process may start. The three SMs have a common structure, Figure 13 shows the SM of the *CpuCard*.

In particular, the transitions are stereotyped as DAM steps. The *DaStep* transition, triggered by the *CPU fail* event, models the failure occurrence step (*kind* tagged-value), and leads the component from the *running* to the *failed* state. The *DaActivationStep* transition occurs when the activation of a repair action becomes enabled; it specifies the number of agents needed to perform the repair (*agentsNumber* tagged-value) as well as the required repair skills (*agents-skill*). The *DaReplacementStep* transition models the

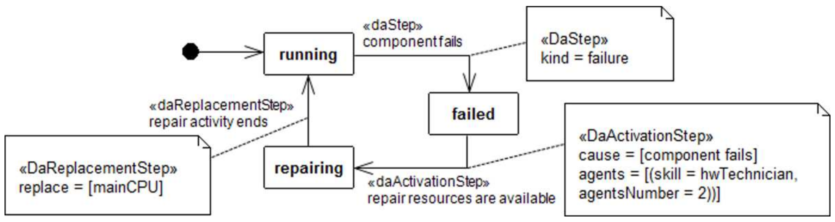


Figure 13: State Machine Diagram.

step of replacing the failed component, then restoring the service.

5.1.2. RBC: Applying DAM2RFT

The application of *dam2rft* transformation to the case study of RBC is depicted in Figure 14, where only a part of the resulting RFT is shown, specifically the sub-tree obtained by translating the *PowerSupply* package.

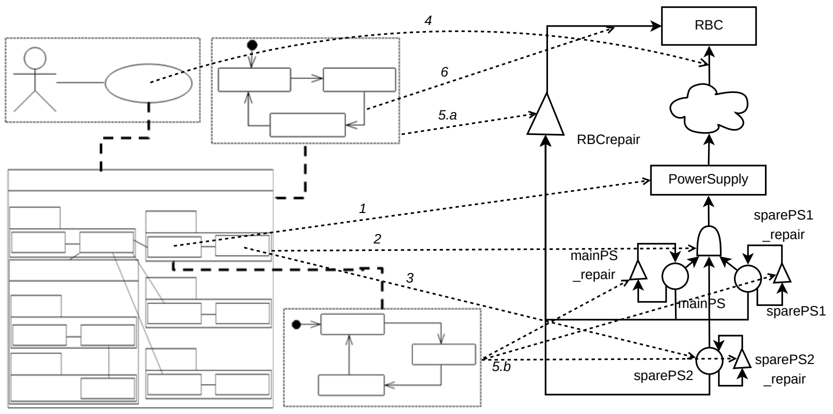


Figure 14: DAM-to-RFT transformation steps.

First *dam2ft* rules are applied to the source model: they are represented by the dotted lines labelled from 1 to 4. Then the *dam2rft* transformation is applied (rules labelled 5 and 6). Rule 1 is applied to the *DaComponent* stereotyped elements of PowerSupply in the RBC Component Diagram and generates a Middle Event for each of them, hence in this case Rule 1 generates the *PowerSupply* FT event. Rule 2 generates the AND input gate because *FTlevel*=1, as described in Subsection 4.1. Rule 3 generates three Basic Events, where three is the number of replicas of *SparePS* (2) plus the number of *mainPS* (1). From the Use Case Diagram, Rule 4 identifies the

DaComponent representing the system to analyse (*DaService*) and generates gate-to-event arcs by recursively looking for sub-components relations. Rule 5 is triggered by the *DaActivationStep* transitions present in the state machines, it generates one RB for the RBC component and three RBs from the *PowerSupply* package. These RBs are filled with relevant data (MTTR, necessary resources) by extracting maintenance related information from the State Machine Diagram and the Component Diagram. Rule 6 links RBs and events by recursive exploration of the sub-tree.

Once the RFT model has been generated it can be solved to perform the availability analysis and efficiently evaluate the probability of an immobilizing failure in presence of different repair policies. The solution process of RFT models is described in [9], the results of the availability analysis of RBC are reported in [28], where a hand-made RFT was proposed.

5.2. Driverless Metro Vehicle

The main service provided by a metro system is the *transport* of passengers. This service is evaluated through mathematical expressions, defined by the customer needs and specified in a tender document. As an example, in this case study, we adopt the Operating System Service Availability (A) as a service measure, that is: $0 \leqslant A \leqslant 1$, $A = 1 - \frac{\sum_i d_i}{Nsec}$, where d_i is the delay, recorded in seconds, measured at terminal station at the end of the i-th trip (varying i from 0 to the number of trips performed in a day); and $Nsec$ is the total daily service time expressed in seconds. In this case study let us assume a target value for the A of 97.0%, with a 18 hours daily service and a headway between trips of 360 seconds (180 nominal trips). When the system is up, we need to demonstrate that a trip arrives on time with a ratio of 98.8% and delayed with a percentage of 1.2%. This last contribution is a sum of three percentages: a 0.6% when the delay is shorter than 3 minutes, a 0.45% when the delay in longer than 3 minutes but shorter than 10 and a pecentage of 0.15% when the delay is longer than 10 minutes. With the aim of classifying the effects of a failure on the *transport* service, three failure consequence classes are hence defined. Note that these classes, depending on the specific formula, are not common to different projects. The three failure consequence classes here defined are reported in Table 4.

The main service *transport* is performed by the cooperation of a set of basic services, offered by the different subsystems (e.g. power supply, signalling and vehicle fleet). Without losing generality, here we suppose that the set of basic services, provided by the vehicle fleet, are *movement*, *conditioning* and

Table 4: Transport: failure consequence classes.

FCC	***Definition***
Negligible (NG)	A failure that causes a delay to service shorter than 3 minutes.
Tolerable (TL)	A failure that causes a delay to service longer than 3 minutes and shorter than 10 minutes.
Intolerable (NTL)	A failure that causes a delay longer than 10 minutes.

powerCapture. As for the *transport* service, also for the basic services some failure consequence classes are defined (see Table 5).

Table 5: Vehicle services: failure consequence classes.

FCC	***Definition***	***Effects***
***movement* service**		
Return To Depot (RT)	A failure that causes the return to the depot at the end of the trip without service delays (100% of max speed).	Transport.NG
Partial Block (PB)	A failure that causes the landing of passengers at the first station and return to the depot at reduced speed (40% of max speed).	Transport.TL
Total Block (TB)	A failure that causes a total block of the vehicle (0% of max speed) and the need for a thrust.	Transport.NTL
***conditioning* service**		
Acceptable (AC)	A failure affecting one conditioner of a single car (coolness ratio reduced at 60%).	No Effect
Down (DW)	A failure affecting both conditioners of a single car (coolness ratio at 0%).	Transport.NG
***power capture* service**		
Down (DW)	A failure where vehicle is not turned on by power supply	Movement.TB Conditioning.DW

Each vehicle consists of several components, whose failures have an impact on the vehicle availability, and consequently on A. For brevity's sake here we restrict our study to the components and failure modes listed in Table 6 which reports a fragment of the FMEA (Failure Modes and Effects Analysis) document of the vehicle.

5.2.1. Vehicle DAM-Rail model

Modelling this case study requires the application of stereotypes coming from DAM and DAM-Rail, in particular to annotate service modes and vehicle component features. The Use Case Diagram representing services (stereotyped as *DAM::DaService*) is shown in Figure 15. The actor *user* of the metro system is associated with the *transport* service, which is the only service perceived by him. Indeed, as said before, the *transport* service strictly depends on the vehicle *movement* and vehicle *conditioning* services, which in

Table 6: Vehicle FMEA Fragment.

Level	**Component**	**Failure Mode**	**Failure rate** $[h^{-1}]$	**num / tot**	**Effects**
1	Electric Unit	-	-	-/1	-
2	Pantograph Unit	-	-	-/2	-
3	Pantograph	Loss of contact	2.5E-6	1/2	PowerCapture.NM
				2/2	PowerCapture.DW
1	Traction Unit	-	-	-/1	-
2	Engine Unit	-	-	-/2	-
3	Engine	Short circuit	8.3E-6	1/2	Movement.PB
				2/2	Movement.TB
		Mechanical breakdown	1.2E-5	1/2	Movement.PB
				2/2	Movement.TB
1	Undercarriage	-	-	-/1	-
2	Wheel Unit	-	-	-/12	-
3	Wheel	Crack	2.5E-7	1/2	Movement.PB
				2/2	Movement.TB
		Border breakdown	2.5E-7	1/2	Movement.PB
				2/2	Movement.TB
1	Air Conditioning	-	-	-/1	-
2	Air Unit	-	-	-/3	-
3	Air Compressor	Electrical	5E-7	1/2	Conditioning.AC
				2/2	Conditioning.DW
		Over Temperature	2.5E-7	1/2	Conditioning.AC
				2/2	Conditioning.DW
	Gas Tank	Pressure Loss	2.7E-8	1/1	Conditioning.DW

turn depend on the *powerCapture* service. The tagged value *usedResources* denotes, for each service, which components provide that service. The *transport* does not specify this tagged value because this service is provided by the cooperation of the basic ones.

Each use case is associated to a State Machine Diagram, which models the state passing between the different service modes. The service modes expected for a single service correspond to the failure consequence classes of the service, i.e. each state depends on the consequences of a failure, hence the nominal state represents the service mode free of failures. As an example, the top of Figure 16 depicts the State Machine Diagram of the *transport* service: the four states model the *Nominal (NM)* service mode and the other

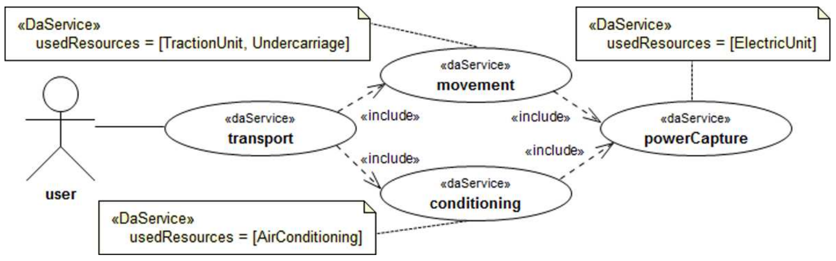


Figure 15: Use Case of Services.

three service modes saying the degradation level of the service. Similarly, the bottom of the Figure depicts the State Machine Diagram for the *conditioning* service. The *entry action* of the latter diagram expresses the effects of the vehicle *conditioning* service degradations on the *transport* service, according to Table 5; in the same diagram, the states representing degraded service modes need to specify the tagged values *failure* to express the relationships between the failure occurrences and the state passing.

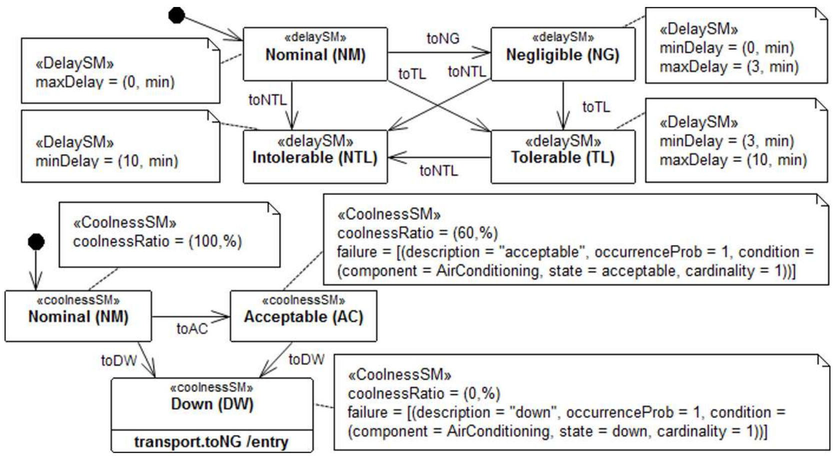


Figure 16: *Transport* and *Conditioning* State Machine Diagrams.

The Component Diagram in Figure 17 provides the description of the vehicle structure previously described in Table 6.

A slice of the complete model is reported which refers to the *AirConditioning* subsystem (Figure 18). The slice points out the tagged values. At the lowest level the basic components are *GasTank* and *AirCompressor*: some tags, in particular failures modes, including their occurrence rate, have

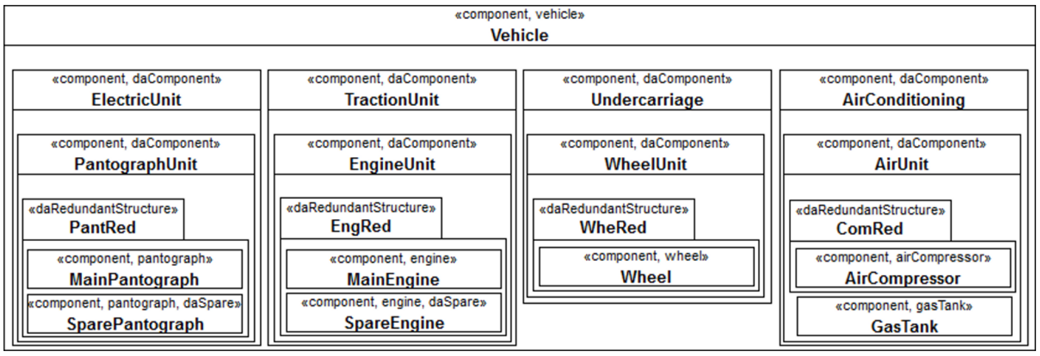
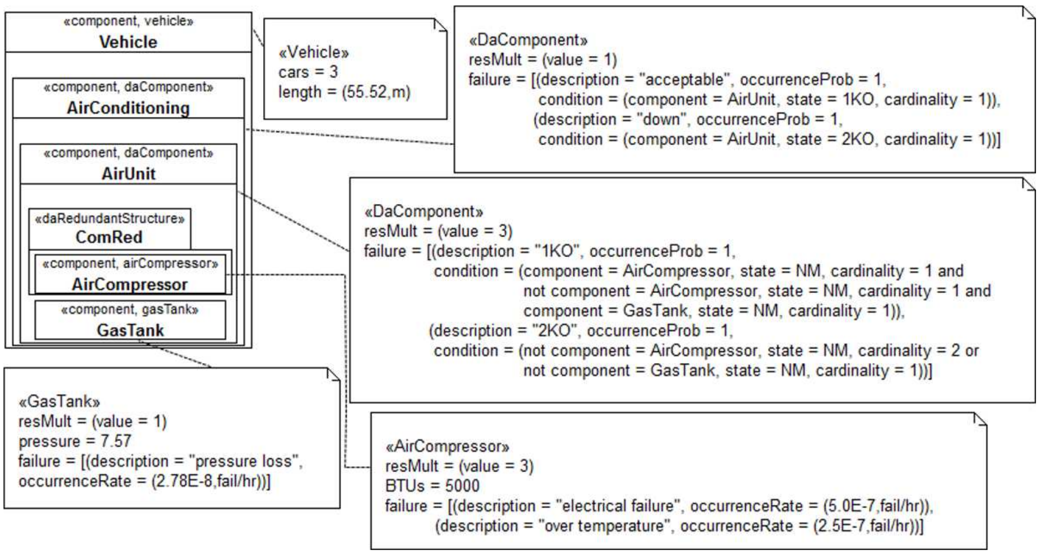


Figure 17: Driverless vehicle Component Diagram.

been annotated for both these components. The *AirCompressor* is enclosed in the *ComRed* redundant structure. Leveling up, at the *AirUnit* level, two failures have been annotated. Notice that the dependency on the failures of the basic components is modelled by a "condition" expression in the DAM Backus-Naur Form (BNF) syntax. At last, *AirConditioning*, which is the *usedResource* of the *conditioning* service (Figure 15), has been annotated with two failures which reference to the state of the air conditioning unit to the *conditioning* service, provided by itself.



«Vehicle»
cars = 3
length = (55.52,m)

«DaComponent»
resMult = (value = 1)
failure = [(description = "acceptable", occurrenceProb = 1,
condition = (component = AirUnit, state = 1KO, cardinality = 1)),
(description = "down", occurrenceProb = 1,
condition = (component = AirUnit, state = 2KO, cardinality = 1))]

«DaComponent»
resMult = (value = 3)
failure = [(description = "1KO", occurrenceProb = 1,
condition = (component = AirCompressor, state = NM, cardinality = 1 and
not component = AirCompressor, state = NM, cardinality = 1 and
component = GasTank, state = NM, cardinality = 1)),
(description = "2KO", occurrenceProb = 1,
condition = (not component = AirCompressor, state = NM, cardinality = 2 or
not component = GasTank, state = NM, cardinality = 1))]

«GasTank»
resMult = (value = 1)
pressure = 7.57
failure = [(description = "pressure loss",
occurrenceRate = (2.78E-8,fail/hr))]

«AirCompressor»
resMult = (value = 3)
BTUs = 5000
failure = [(description = "electrical failure", occurrenceRate = (5.0E-7,fail/hr)),
(description = "over temperature", occurrenceRate = (2.5E-7,fail/hr))]

Figure 18: Air conditioning.

5.2.2. Vehicle: Applying DAMRail2BN

Figure 19 depicts the application of *dam2bn* to the *AirConditioning* component of the vehicle. It works according to the rules described in Subsection 5.1.2: Rules 1 and 2 are applied to the Component Diagram in Figure 17, Rules from 3 to 5 are applied to the Use Case Diagram in Figure 15, Rule 6 is applied to the State Machine Diagram of the service in Figure 16. For each basic component, Rule 1 generates a number of nodes equal to the value specified by *ResMult*, hence *GasTank* is translated into one node (dotted line 1.a) and *AirCompressor* generates two nodes. According to the *failure* tagged value, the random variable values associated to the *GasTank* node are two (nominal service mode and pressure loss), while the random variable values associated to the *AirCompressor* node are three (nominal service mode, electrical failure and over temperature). Rules 1 and 2 also generate the *AirUnit* nodes: the value 3 specified in the tagged value *ResMult* replicates three times the structure, one for each *AirUnit* (dotted line 1.b). Rule 1 and 2 also generate the arcs between nodes. Similarly the nodes *AirConditioning* and *Vehicle* and their related arcs are generated from the Component Diagram. The *usedResources* tagged value of the *conditioning* service is used to generate the node conditioning and the link from the *AirConditioning* node to the *conditioning* node. Through the application of the Rules from 3 to 5, the same node also takes a contribution from the *powerCapture* node, according to the relationships between services. Analogously *dam2bn* works to generate the BN model of the movement service. Finally the CPTs are generated according to the application of the Rule 6. The entire Bayesian Network is created for the service *transport*, on which, in a second step, the M2T transformation has been applied in order to generate the concrete file for the JavaBayes tool. Analysis results and their discussion are not reported for sake of space.

6. A remark on industrial application and conclusions

This paper addresses the definition of an approach specifically oriented to integrate the verification of RAM (Reliability, Availability, Maintainability) requirements in Model-Driven development of railway applications. To this aim the paper defines a proper transformational process based on the extension of the MARTE-DAM profile and the definition of DAM-Rail from it, allowing to construct railways-specific models. RAM analyses may be performed by generating formal models from DAM/DAM-Rail specifications,

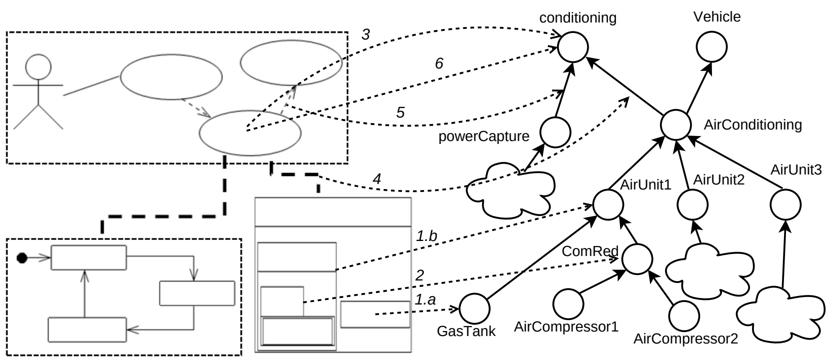


Figure 19: DAM-to-BN transformation steps.

through the definition of Model-to-Model and Model-to-Text transformations. In order to describe the whole process, two availability studies have been developed on a modern railway controller and a metro driverless vehicle.

Model-Driven paradigm promises help in handling the increasing complexity of modern development processes. Raising the abstraction level of models and applying proper transformational techniques it is possible to obtain a high integration and automation level in industrial settings where integration, automation and traceability are key issues. According to Model-Driven development, system models are usually constructed by using visual modelling languages, the most used of them is UML which includes extensibility mechanisms enabling the definition of UML profiles, i.e. UML-based DSML. While there has been a great interest in the last decade on MDE paradigms for complex systems, a certain scepticism has accompanied their adoption in the safety-critical railway industry, limiting UML-based approaches to documentation issues. Such a scenario is going to rapidly evolve in the future as the results of recent research projects have proved the feasibility and the advantages of semantic modelling, making model-based approaches easier and less error prone [29]. MDE approaches should be framed in the general context of formal methods for intelligent transportation systems [30] as “DSL can help to reduce the inconsistencies that can occur between the artefacts created in a manual process because of their contribution to the preservation of consistency and completeness” [31].

Future research, beyond the extension of the DAM-Rail approach to Safety, will also investigate the foundation of DSML definition and its relationships with the Model-Driven Engineering. Indeed, DSML development is still an emerging discipline with few established guidelines and patterns, and on the other hand we have experienced, by defining the DAM-Rail profile, that the relationships between the standard OMG's Model-Driven Architecture and the Model-Driven Engineering meta-modelling stacks have not been clarified enough. The entire framework will be completed by research work on model transformation verification and traceability in order to fully realize the "correct-by-construction" paradigm.

References

[1] J. Jürjens, Secure systems development with UML, Springer, 2005.

[2] D. Petriu, M. Woodside, An intermediate metamodel with scenarios and resources for generating performance models from UML designs, Software and Systems Modeling (SoSyM) 6 (2) (2007) 163–184. doi:10.1007/s10270-006-0026-8.

[3] S. Mustafiz, X. Sun, J. Kienzle, H. Vangheluwe, Model-driven assessment of system dependability, Software and Systems Modeling 7 (2008) 487–502.

[4] V. Cortellessa, H. Singh, B. Cukic, Early reliability assessment of UML based software models, in: Proceedings of the 3rd international workshop on Software and performance, WOSP '02, ACM, New York, NY, USA, 2002, pp. 302–309.

[5] V. Casola, A. Fasolino, N. Mazzocca, P. Tramontana, An AHP-based framework for quality and security evaluation, Proceedings - 12th IEEE International Conference on Computational Science and Engineering, CSE 2009 3 (2009) 405–411.

[6] S. Bernardi, F. Flammini, S. Marrone, J. Merseguer, C. Papa, V. Vittorini, Model-driven availability evaluation of railway control systems, in: SAFECOMP 2011, Vol. 6894 of LNCS, 2011.

[7] B. Selic, 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'07), 2007.

[8] S. Bernardi, J. Merseguer, D. C. Petriu, A dependability profile within MARTE, Software and System Modeling 10 (3) (2011) 313–336.

[9] D. Codetta Raiteri, M. Iacono, G. Franceschinis, V. Vittorini, Repairable fault tree for the automatic evaluation of repair policies, in: Proceedings of the 2004 International Conference on Dependable Systems and Networks, 2004, pp. 659–668.

[10] E. Charniak, Bayesian networks without tears: making bayesian networks more accessible to the probabilistically unsophisticated, AI Mag. 12 (4) (1991) 50–63.

[11] UNISIG, ERTMS/ETCS RAMS requirements specification, ref. 96s1266.

[12] CENELEC, EN50126 railways applications - the specification and demonstration of reliability, availability, maintainability and safety (RAMS) (1999).

[13] OMG, Unified Modeling Language: Infrastructure and Superstructure, version 2.4, formal/11-08-05 (May 2011).

[14] OMG, Object Constraint Language, version 2.3, formal/12-01-01 (January 2012).

[15] OMG, UML Profile for MARTE: Modeling and Analysis of Real-time Embedded Systems, version 1.1, formal/11-06-02 (June 2011).

[16] A. Avizienis, J.-C. Laprie, B. Randell, C. E. Landwehr, Basic concepts and taxonomy of dependable and secure computing, IEEE Trans. Dependable Sec. Comput. 1 (1) (2004) 11–33.

[17] F. Lagarde et al., Improving UML profile design practices by leveraging conceptual domain models, in: 22nd Int.l Conf. on Automated Software Engineering, Atlanta (USA), ACM, 2007, pp. 445–448.

[18] G. Cozman, JavaBayes - User Manual, version 0.346.

[19] R. Sahner, K. Trivedi, A. Puliafito, Performance and reliability analysis of computer systems (an example-based approach using the sharpe software), Reliability, IEEE Transactions on 46 (3) (1997) 441.

[20] S. Baarir, M. Beccuti, D. Cerotti, M. De Pierro, S. Donatelli, G. Franceschinis, The greatspn tool: recent enhancements, SIGMETRICS Perform. Eval. Rev. 36 (4) (2009) 4–9.

[21] F. Fleurey, J. Steel, B. Baudry, Validation in model-driven engineering: testing model transformations, in: First International Workshop on Model, Design and Validation, 2004.

[22] M. Asztalos, L. Lengyel, T. Levendovszky, Towards automated, formal verification of model transformations, in: Proceedings of ICST '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 15–24.

[23] D. Wagelaar, R. V. D. Straeten, D. Deridder, Module superimposition: a composition technique for rule-based model transformation languages, Software and System Modeling 9 (3) (2010) 285–309.

[24] F. Jouault, F. Allilaire, J. Bézivin, I. Kurtev, ATL: A model transformation tool, Sci. Comput. Program. 72 (1-2) (2008) 31–39.

[25] S. Marrone, C. Papa, V. Vittorini, Multiformalism and transformation inheritance for dependability analysis of critical systems, in: Proceedings of 8th Integrated formal methods, IFM'10, 2010, pp. 215–228.

[26] A. Bobbio, L. Portinale, M. Minichino, E. Ciancamerla, Improving the analysis of dependable systems by mapping fault trees into bayesian networks, Reliability Engineering and System Safety 71 (3) (2001) 249–260.

[27] UIC, ERTMS/ETCS class1 system requirements specification, ref. SUBSET-026, issue 2.2.2 (2002).

[28] F. Flammini, N. Mazzocca, M. Iacono, S. Marrone, Using Repairable Fault Trees for the evaluation of design choices for critical repairable systems, in: Proceedings of HASE'05, 2005, pp. 163–172.

[29] A. Fiaschetti et al, On the use of semantic technologies to model and control security, privacy and dependability in complex systems, in: Proceedings of SAFECOMP'11, 2011, pp. 467–479.

[30] A. Fantechi, F. Flammini, S. Gnesi, Formal methods for intelligent transportation systems, in: Proceedings of the 5th international conference on Leveraging Applications of Formal Methods, Verification and Validation: applications and case studies, ISoLA'12, 2012, pp. 187–189.

[31] J. Hutchinson, An empirical assessment of model driven development in industry, Ph.D. thesis, B. Sc. Hons Lancaster University (2011).