Model-Driven Availability Evaluation of Railway Control Systems

Simona Bernardi[1], Francesco Flammini[2], Stefano Marrone[3], José Merseguer[4], Camilla Papa[5], and Valeria Vittorini[5]

[1] Centro Universitario de la Defensa, Academia General Militar (Spain)
simonab@unizar.es
[2] AnsaldoSTS, Innovation and Competitiveness Unit (Italy)
francesco.flammini@ansaldo-sts.com
[3] Seconda Università di Napoli, Dip. di Matematica (Italy)
stefano.marrone@unina2.it
[4] Dep.to de Informática e Ingeniería de Sistemas, Universidad de Zaragoza (Spain)
jmerse@unizar.es
[5] Università di Napoli "Federico II", Dip. di Informatica e Sistemistica (Italy)
{camilla.papa,valeria.vittorini}@unina.it

Abstract. Maintenance of real-world systems is a complex task involving several actors, procedures and technologies. Proper approaches are needed in order to evaluate the impact of different maintenance policies considering cost/benefit factors. To that aim, maintenance models may be used within availability, performability or safety models, the latter developed using formal languages according to the requirements of international standards. In this paper, a model-driven approach is described for the development of formal maintenance and reliability models for the availability evaluation of repairable systems. The approach facilitates the use of formal models which would be otherwise difficult to manage, and provides the basis for automated models construction. Starting from an extension to maintenance aspects of the MARTE-DAM profile for dependability analysis, an automated process based on model-to-model transformations is described. The process is applied to generate a Repairable Fault Trees model from the MARTE-DAM specification of the Radio Block Centre - a modern railway controller.

Keywords: Automated Model Generation, ERTMS/ETCS system, Model Transformation, Repairable Fault Trees, UML profiles.

1 Introduction

The development of mission-critical systems has to tackle several challenges, including the evaluation of RAMS (Reliability, Availability, Maintainability, Safety) attributes since early stages of system life-cycle till the possible final certification phase. Evaluation approaches by means of formal models have proven to be effective in assessing RAMS attributes. However, many formalisms (e.g., Fault Trees) suffer from limited expressive power when dealing with complex repairable systems, or they are limited in usability and solving efficiency (e.g., Stochastic Petri Nets). To solve those issues, recently some "hybrid" approaches have been proposed, trying to combine the advantages of different formalisms [10].

One further step toward the simplification of the model-based RAMS evaluation is the use of high-level modeling languages, derived from the Unified Modeling Language (UML), and model transformation, which is the basis of the Model Driven Engineering (MDE) methodology. Model transformations processes transform a source model into a target model based on transformation rules [18]. A first attempt of defining such a process, in the dependability context, has been performed within the HIDE project [3]. More recently, specific profiles have been developed to specify non-functional properties (NFP) on UML diagrams, such as the OMG standard MARTE (Modeling and Analysis of Real-Time and Embedded Systems) UML profile [20].

An important aspect of the MDE related work is the availability of open source case tools and workbenches. These MDE platforms integrate tools which support the main technologies to implement Model-to-Model (M2M) transformations, such as ATL [11] or QVT [14]. The integration of formal methods and techniques into MDE based development process is still an open issue. MARTE introduces the possibility of annotating models in order to cope with NFPs but it does not provide support to dependability analysis. A recent work [2] proposes an extension of MARTE for dependability and modeling (MARTE-DAM). Nevertheless MARTE-DAM does not define M2M transformations for the automated generation of formal analysis models from MARTE artifacts.

In the past, several research efforts have focused on the derivation of formal models from UML diagrams, as surveyed in [2]. As specifically regards fault trees (FT) and their extensions, Pai and Dugan developed a method to derive Dynamic Fault Trees from UML system models [15] and D'Ambrogio et al. defined a method to generate FT models from a set of sequence diagrams in order to predict software reliability [5]. All these works are partial solutions to the problem and their systematic application appears to have been absent in the subsequent evolution of UML.

In this paper, we integrate the above mentioned approaches: we apply model-driven techniques to generate formal models of critical systems. Starting from a high level specification of the system expressed by an extended UML profile, we define and implement proper M2M transformations in order to automate the generation of availability models of a modern railway controller. The contribution of the paper is twofold: on the one hand, it shows how a model-driven approach may also promote the applicability of formal modeling in industrial settings; on the other hand, we extend the MARTE-DAM profile for dependability modeling and define the M2M transformations to generate Repairable Fault Tree [4] models from the extended profile. In particular, considering MARTE-DAM, we enrich the fault tolerance and maintenance aspects of the profile [2] to enable the specification of complex repairable systems.

It is worth to mention that, beside UML, SysML [19] and AADL [1] have been also considered as source specification languages in M2M transformations.

The work [8] proposes a joint use of UML-MARTE and SysML for the automatic generation of certification-related information in safety domain. Indeed, although SysML provides support to manage requirements and system design together, it lacks of standard concepts for dealing with specific dependability concerns. On the other hand, AADL enables the specification of dependability requirements and properties of software systems. The works [17] and [9] both propose transformation techniques to get formal dependability models from AADL specifications (respectively, Generalized Stochastic Petri Nets and probabilistic finite state-machines).

2 The MARTE-DAM Profile

The MARTE [20] profile provides a *lightweight* extension of UML (i.e., through the use of stereotypes, tagged-values and constraints) to specify system non-functional properties (NFPs), according to a well-defined Value Specification Language (VSL) syntax. Stereotypes extend the semantics of UML meta-classes with concepts from the target domain. They are made of tags whose types can be basic UML types (e.g., integer) or MARTE NFP types (e.g., *NFP_Integer* in Tab.1). The latter are of special importance since they enable the description of relevant aspects of a NFP using several properties, such as *value*, a value or parameter name (prefixed by the dollar symbol); *source*, the origin of the NFP (e.g., a requirement - *req*) *statQ*, the type of statistical measure (e.g., mean).

The "Dependability Analysis and Modeling" (DAM) [2] profile is a MARTE specialization. A MARTE-DAM annotation *stereotypes* a UML design model element, then extending its semantics with dependability concepts (e.g., annotating a UML State Machine transition as a failure step). Moreover, DAM enriches the MARTE types with basic and complex dependability types. The latter (e.g., *DaRepair*) are composed of attributes (e.g., *MTTR*) that can be MARTE NFP types (e.g., *NFP_Duration*) or simple types.

The DAM profile relies on the definition of the DAM *domain model* which represents the main dependability concepts from the literature according to a component-based view of the system to be analyzed [7]. In the domain model, the system is defined by a set of *components* bounded together through *connectors*, in order to interact. The system delivers a set of high-level *services*, that can be detailed - at 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, failures.* A fault is the original cause of errors and it affects system components. Errors are related to steps and they can be propagated from the faulty component to other components it interacts with. Errors may cause failures at different levels: at step level, when the service provided by the component becomes incorrect; at component level, when the component is unable to provide service; at service level, when the failure is perceived by external users.

The domain model includes also redundancy and maintenance concepts. The *Redundancy* model (Figure 1) represents UML hw/sw *redundant structures* to increase system fault tolerance (FT). These structures are made of components, among them *FT components* [12], which can play different roles.

The *Maintenance* model (Figure 2) concerns repairable systems and includes concepts that are necessary to support the evaluation of system availability, that is the *maintenance actions* undertaken to restore the system affected by threats. According to [7], we distinguish *repairs* of system components, that involve the participation of external agents (e.g., repairman, test equipment, etc) and *recovery* of services, which do not require the intervention of the latter.

In this paper, we aim at increasing modeling and analysis capabilities of DAM regarding redundancy and maintenance, as explained in the next Section.

3 A DAM Extension for Maintenance and Fault Tolerance

The DAM domain models of redundancy and maintenance provide the basis for the definition of proper extensions, i.e., stereotypes, tagged-values and OCL constraints. In particular, the stereotypes and tagged values will be used to annotate UML designs with fault tolerance and maintenance requirements/properties, while OCL constraints are assigned to UML extensions to guarantee UML annotations compliant to the DAM domain concepts.

The extension of DAM domain model consists in: 1) augmenting the maintenance model w.r.t. the one presented in [2] (grey classes in Figure 2 account for the extension). *Activation steps* initiate maintenance actions as consequence of component failures. An activation step defines its *priority* as well as a *preemption* policy, moreover it relates to a group of agents with the required *skills* to perform the step. The activation step also relates to the failures that caused it; 2) improving the redundancy model (Figure 1), keeping its original shape and only adding the *FTlevel* attribute in the redundant structure stereotype.

The stereotypes associated to the redundancy and maintenance domain models are shown in Table 1 (first column) and they corresponds to concrete classes in Figs. 1 and 2. For reason of space, we omit stereotypes that come from classes defined in the core domain model (e.g., component, service and step). A tag of a stereotype (Table 1, third column) can be derived, together with its multiplicity, from one of the following sources in the domain model: 1) an attribute of the corresponding class, i.e., the attribute *errorDetCoverage* in *Adjudicator* class (Fig.1) has been mapped onto the tag *DaAdjudicator::errorDetCoverage*; 2) an association-end role, i.e., the *substituteFor* role in the association between *Spare* and *Component* classes (Fig.1) has been mapped onto the tag *DaSpare::substitutesFor*. The types of the tags can be either simple types (e.g., the enumeration *skillType* assigned to the *skill* tag of the *AgentGroup* stereotype), MARTE-NFP types (e.g., *NFP_Integer*), or complex dependability types. The latter are data types derived from classes in the domain models, they are characterized by a set of attributes corresponding to the ones of the mapped classes. Basically, they may represent either threat characterization (e.g., the

Table 1. Stereotypes and tags

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

class *Failure* has been mapped onto the *DaFailure* complex type) or concrete maintenance actions (e.g. the *DaRepair* complex type corresponds to the class *Repair*).

Given a UML model of the system under analysis, the main issue - from the software engineer point of view - is which model elements in a UML diagram (e.g., a state in a state-machine diagram or a component in a component diagram) can be stereotyped in order to specify NFPs through tagged-values. As shown in Table 1 (second column), each stereotype may either specialize a previously defined MARTE-DAM stereotype or directly extend a UML meta-class. Then, a given model element can be stereotyped as X if the stereotype X eventually extends the meta-class the former belongs to (either directly or indirectly, through stereotype generalization). For example, all the sub-stereotypes of *DaComponent* can be applied to UML elements representing system software and hardware resources (e.g., classes, instances, components, nodes), since *DaComponent* specializes the MARTE *Resource* stereotype and the latter extends the corresponding UML meta-classes. 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 behavior-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 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.

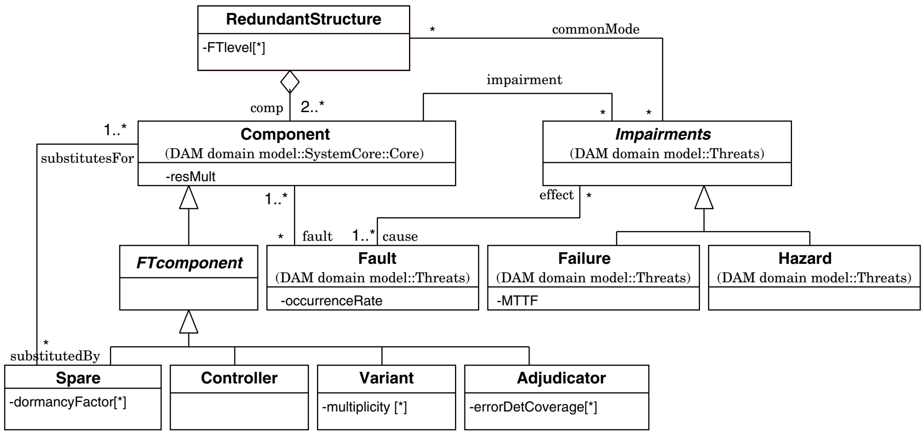


Fig. 1. Redundancy domain model

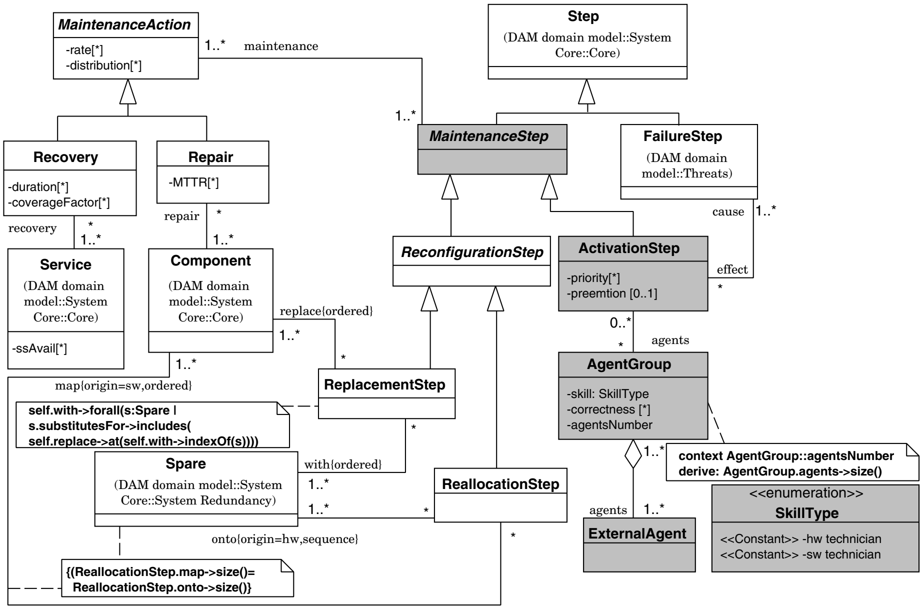


Fig. 2. Maintenance domain model

4 Automated Generation of RFT Models

Automated generation of formal models from UML models has already been studied and such technique can be considered part of the process schema depicted in Fig. 3. The first step is the definition of a design UML model by system designer. At this level dependability aspects have been not considered yet. The next step is constituted by the application of MARTE-DAM profile to the UML model and the definition of dependability parameters that characterize the system. Then a MARTE-DAM model can be automatically translated into a formal model: this translation is conducted by means of model-to-model transformations that are defined on the base of a source and a destination metamodels (i.e. the formalizations of the languages in which source and destination models are expressed). Generated formal models can be finally analyzed, allowing the validation of the model or its eventual refining by parameters tuning and/or redefinition of the architecture. The formal language should be properly chosen according to the specific dependability aspect to be analyzed.

Since we are focusing on maintenance, source formalism is constituted by MARTE-DAM (containing the extension introduced in the previous Section) and destination formalism is the Repairable Fault Tree (RFT) that was introduced to ease the modeler's approach to complex repair policy modeling and evaluation [4]. The RFT formalism [4] *integrates* GSPNs and Fault Trees: repair policies are represented by nodes - called Repair Boxes - which encapsulate GSPN nets, and a Fault Tree describes the faults that may happen and their contribution to the occurrence of a failure. The Repair Box connected to a Fault Tree event models a repair action that can be performed on the related system sub-component. RFT metamodel is given in Figure 4 while the DAM meta-model has been described in Section 2 and Section 3.

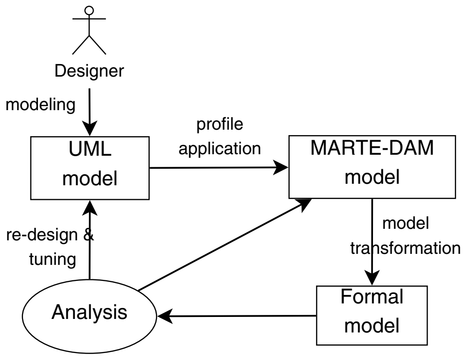


Fig. 3. A reference model-driven process

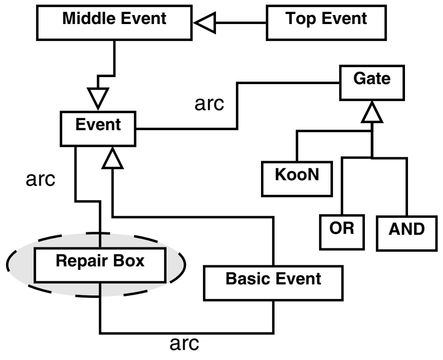


Fig. 4. RFT meta-model

Defining complex model transformations from scratch can be a hard task, so transformations composition and reuse are being widely investigated. In this paper we apply **module superimposition**, a widespread mechanism for coarse-grain composition of transformations which shifts the focus of reuse from rule to set of rules (transformation modules). In practice, superimposition allows for defining a new transformation by the union of the rules set of existing ones. Superimposition is well supported by the most important transformation languages (including ATL). Compositional approaches are enhanced by inheritance relationships between languages. In particular, the RFT language is an extension of FTs obtained by adding the Repair Box element: the hierarchical nature of this formalism and its implications on building model transformations by composition has been already studied in [13]. On the other hand, MARTE-DAM has a decoupled structure due to the dependency relations among packages, as depicted in Figure 5. Hence, M2M transformations from MARTE-DAM to RFT may benefit from a *divide-et-impera* approach. According to the above considerations, the two transformations described in Figure 5 have been defined and implemented in ATL.

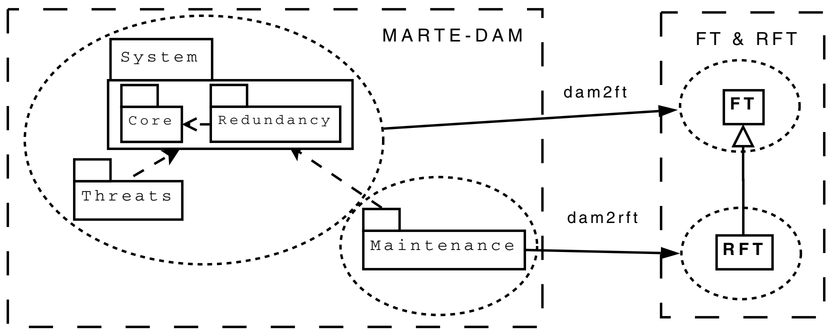


Fig. 5. DAM-to-RFT transformation schema

Starting from a DAM specification, *dam2ft* generates the Fault Tree from the System and Threats domain sub-models, and *dam2rft* adds the RBs and related arcs from the Maintenance domain sub-model.

The transformation implemented by *dam2ft* works as follows: the Top Event is associated to the failure of the system which provides the service (specified by the use case diagram). From the component diagram, events and gates are created by recursively applying the following rules:

1. *DaComponent* is translated into: a) one Middle Event or b) one Middle Event and as many Basic Events as specified by the *resMult* value (i.e., resource multiplicity) if the *fault* tagged value is not null;
2. *DaSpare* is translated into as many Basic Events as specified by *resMult* value if the *fault* tagged value is not null;
3. An input gate is generated for each Middle Event: an OR gate if *DaComponent* does not belong to a *DaRedundantStructure*, an AND gate if *DaComponent* belongs to a *DaRedundantStructure* with *FTlevel*=1, while a KooN gate is generated if *FTlevel*> 1;
4. An input arc is generated to specify an input Middle Event of gate if a sub-component relationship exists between *DaComponent* associated to the Middle Event and the one associated to the output Middle Event of the gate;
5. An arc is always generated from a Basic Event to a gate whose output Middle Event comes from *DaComponent* (point 1.b)) or from *DaComponent* substituted by *DaSpare* (point 2).

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 chart 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 a 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. These information are 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: 1) between 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) between RB and all the Basic Events that are present in the sub-tree whose root is the RB triggering event. Once RFT model has been generated, a Generalized Stochastic Petri Nets 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 [13].

5 The 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 new European railway signalling and control systems [6] 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 traveling across the track area under its supervision. At this aim, RBC elaborates messages to be sent to the trains on basis of information received from external trackside systems and on basis of information exchanged with on-board sub-systems. The unavailability of a RBC is critical, as there is no way for the signalling system to work without its contribution. In case of a RBC failure, all the trains 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[1]. The ERMTS/ETCS standard requires compliance with the RAM requirements [16] whose fulfillment has to be properly demonstrated. Specifically, the quantifiable contribution of RBC system to operational unavailability must be not more than 10^{-6} (see [16], §2.3.3). The standards do not impose constraints on the system architecture. Hence, different implementations are possible. A reference architecture of RBC must exhibit a high level of redundancy to improve the fault tolerance of the system. In this paper the system consists of three commercial CPU-RAM cards and a redundant FPGA based voter in a TMR (Triple Modular Redundancy) configuration. The GSM-R and WAN communication sub-systems

[1] An Immobilising Failure occurs when at least two trains are no more under ERTMS/ETCS supervision [16].

are also chosen as COTS (Commercial Off The Shelf). The RBC configuration in figure is completed by three commercial power supplies and a redundant standard backbone (used as system BUS).

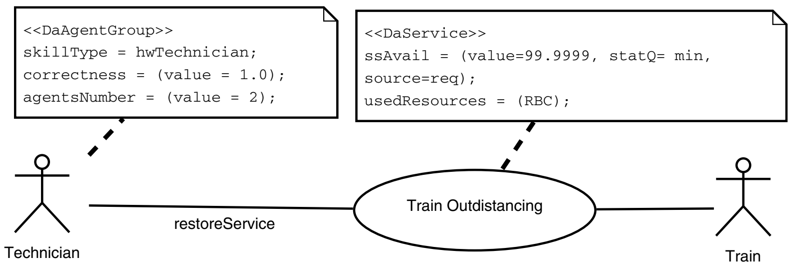
Maintenance policies are a fundamental aspect of RBC life-cycle for their impact on system availability. ERTMS/ETCS gives no restrictive requirements for the maintainability parameters and this leaves much freedom in designing repair policies. Of course, it must be proved that the system still meet the availability requirement. The rest of this Section applies introduced modeling and transformational approach to the RBC case study. We limit our study to the hardware contribution to availability: as MARTE-DAM can be applied on both hw and sw UML models, we could apply this process on software systems.

5.1 DAM Model

The DAM specification of RBC used to generate the RFT consists of an use case diagram, a component diagram and a set of state chart diagrams. The diagrams are annotated with the DAM extensions introduced in Section 3.

The use case diagram in Figure 6 represents the main functionality of the RBC: the train outdistancing. The use case is stereotyped *DaService* to explicitly indicate (by 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 (*skillType* tagged-value) who participate in the repair process; here two technicians (*agentsNumber*) are assumed to accomplish repair activities correctly (*correctness*).



<<DaAgentGroup>>
skillType = hwTechnician;
correctness = (value = 1.0);
agentsNumber = (value = 2);
<<DaService>>
ssAvail = (value=99.9999, statQ= min, source=req);
usedResources = (RBC);
Technician
restoreService
Train Outdistancing
Train

Fig. 6. Train Outdistancing Use Case

The component diagram in Figure 7 provides a high level description of the RBC components whose failures affect the system dependability. 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

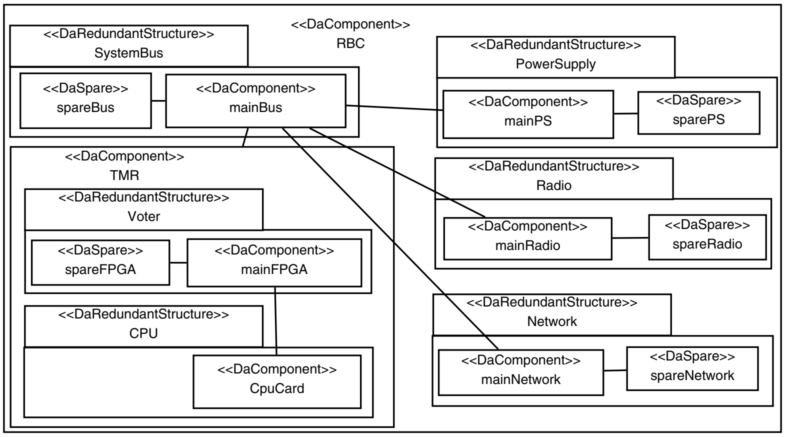


Fig. 7. RBC Component Diagram

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 8, where several tagged-values associated to 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 main *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 fault occurrence rate (*fault.occurrenceRate*) and Mean Time To Repair (*MTTR*) values;

Finally, the RBC specification includes several State Chart diagrams (SC), one for each repairable component. The SC models the dynamics of the repair process of a specific component. In the RBC system, a diagnostic mechanism is present for three components (specifically, `RBC`, `mainPS` and `CpuCard`); when one of the latter fails, a repair process may start. The three SCs have a common structure, Figure 9 shows the SC 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 (*agentSkill*). The *DaReplacementStep* transition models the step of replacing the failed component, then restoring the service.

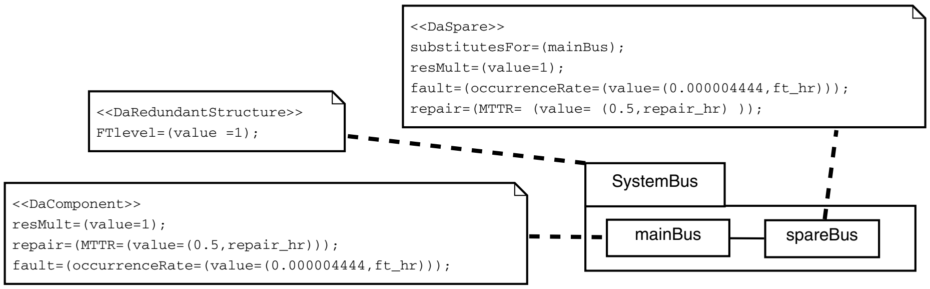

<<DaSpare>>
substitutesFor=(mainBus);
resMult=(value=1);
fault=(occurrenceRate=(value=(0.000004444,ft_hr)));
repair=(MTTR= (value= (0.5,repair_hr)));
<<DaRedundantStructure>>
FTlevel=(value =1);
<<DaComponent>>
resMult=(value=1);
repair=(MTTR=(value=(0.5,repair_hr)));
fault=(occurrenceRate=(value=(0.000004444,ft_hr)));
SystemBus
mainBus
spareBus

Fig. 8. System Bus

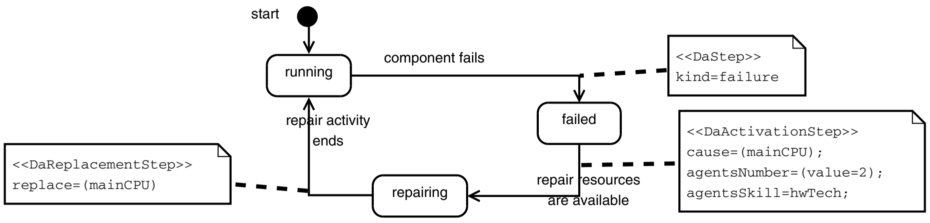


Fig. 9. State Chart Diagram

5.2 Generation of the RFT Model

The process by which transformations generate the RFT model is depicted in Figure 10 where only a part of the resulting RFT is shown, specifically the sub-tree obtained by translating the *PowerSupply* package.

First *dam2ft* rules are applied to the source model: they are represented by the dotted lines labeled from 1 to 4. Then the *dam2rft* transformation is applied (rules labeled 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 Section 4. Rules 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, Rules 4 identifies the *DaComponent* representing the system to analyze (*DaService*) and generates gate-to-event arcs by recursively looking for sub-components relations.

Rules 5 is triggered by the *DaActivationStep* transitions present in the state charts, 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 chart diagram and the component diagram. Rule 6 links RBs and events by recursive exploration of the sub-tree.

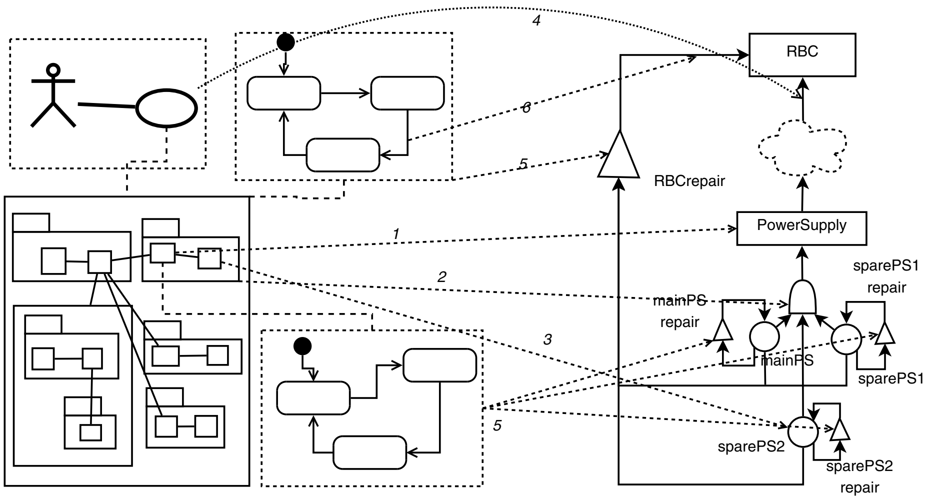


Fig. 10. The RBC Repairable Fault Tree generation

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 [4], the results of the availability analysis of RBC are reported in in [10], where an hand-made RFT is proposed.

6 Conclusions and Future Work

In this paper, we have presented an enhancement of the MARTE-DAM profile in order to improve the capability of this profile to model complex repairable systems. We have also proposed an approach integrating DAM models and Repairable Fault Trees by means of M2M transformations. The suitability of the profile extension and the proposed transformations has been proved on the real case study of the Radio Block Centre. Next steps in this activity will include the development of meta-modeling, modeling and transformational techniques able to fully generate complex repair policies.

References

1. SAE-AS5506/1 Architecture Analysis and Design Language Annex (AADL): Vol.1, annex E:Error Model, International Society of Automotive Engineers (2006)
2. Bernardi, S., Merseguer, J., Petriu, D.C.: A Dependability Profile within MARTE. Journal of Software and Systems Modeling (2009)
3. Bondavalli, A., Latella, D., Dal Cin, M., Pataricza, A.: High-Level Integrated Design Environment for Dependability (HIDE). In: Proceedings of the Fifth International Workshop on Object-Oriented Real-Time Dependable Systems, WORDS 1999, pp. 87–92. IEEE Computer Society, Washington, DC, USA (1999)

4. Codetta Raiteri, D., Iacono, M., Franceschinis, G., Vittorini, V.: Repairable fault tree for the automatic evaluation of repair policies. In: Proceedings of the 2004 International Conference on Dependable Systems and Networks, pp. 659–668. IEEE Computer Society, Washington, DC, USA (2004)
5. D'Ambrogio, A., Iazeolla, G., Mirandola, R.: A method for the prediction of software reliability. In: Proc. of the 6-th IASTED Software Engineering and Applications Conference, SEA 2002 (2002)
6. ERTMS/ETCS System Requirements Specification (SRS), SUBSET-026, Issue 3.0.0 (2008)
7. Avizienis, A., et al.: Basic concepts and taxonomy of dependable and secure computing. IEEE Trans. on Dependable and Secure Computing 1(1), 11–33 (2004)
8. Cancila, D., et al.: SOPHIA: a modeling language for model-based safety engineering. In: 2nd International Workshop on Model Based Architecting and Construction of Embedded Systems, Denver, Colorado, USA, October 6, pp. 11–26. CEUR (2009)
9. Bozzano, M., et al.: Safety, dependability and performance analysis of extended AADL models. The Computer Journal 54(5), 754–775 (2011)
10. Flammini, F., Mazzocca, N., Iacono, M., Marrone, S.: Using repairable fault trees for the evaluation of design choices for critical repairable systems. In: IEEE International Symposium on High-Assurance Systems Engineering, pp. 163–172 (2005)
11. Jouault, F., Kurtev, I.: On the architectural alignment of ATL and QVT. In: Proceedings of the 2006 ACM Symposium on Applied Computing, SAC 2006, pp. 1188–1195. ACM, New York (2006)
12. Lyu, M.R.: Software Fault Tolerance. John Wiley & Sons, Ltd., Chichester (1995)
13. 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)
14. MOF Query/Views/Transformations. Final Adopted Spec., ptc/05-11-01 (2005)
15. Pai, G.J., Dugan, J.B.: Automatic Synthesis of Dynamic Fault Trees from UML System Models. In: Proceedings of the 13th International Symposium on Software Reliability Engineering, pp. 243–254. IEEE CS, Washington, DC, USA (2002)
16. ERTMS/ETCS RAMS Requirements Specification. Ref. 96s1266 (1998)
17. Rugina, A.-E., Kanoun, K., Kaaniche, M.: A system dependability modeling framework using AADL and GSPNs. In: de Lemos, R., Gacek, C., Romanovsky, A. (eds.) Architecting Dependable Systems IV. LNCS, vol. 4615, pp. 14–38. Springer, Heidelberg (2007)
18. Sendall, S., Kozaczynski, W.: Model transformation: The heart and soul of model-driven software development. IEEE Softw. 20, 42–45 (2003)
19. Systems Modeling Language. SysML, `http://www.sysml.org`
20. UML profile for Modeling and Analysis of Real-Time and Embedded Systems (MARTE), Version 1.0, OMG document formal/2009-11-02 (November 2009)