Adding Dependability Analysis Capabilities to the MARTE Profile*

Simona Bernardi[1], José Merseguer[2], and Dorina C. Petriu[3]

[1] Dipartimento di Informatica, Università di Torino, Italy
bernardi@di.unito.it
[2] Departamento de Informática e Ingeniería de Sistemas, Universidad de Zaragoza, Spain
jmerse@unizar.es
[3] Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada
petriu@sce.carleton.ca

Abstract. Dependability is a non-functional property that should be assessed early in the software lifecycle. Although several UML profiles exist for quantitative annotations of non-functional properties, none of them provides concrete capabilities for dependability analysis of UML system models. In this paper, we propose a dependability analysis and modeling profile. The objective is twofold: to reuse proposals from the literature on deriving dependability models from UML annotated specifications and to be compliant with the recently adopted MARTE profile, which provides a framework for general quantitative analysis concepts that can be specialized to a particular analysis domain. The profile definition process was done in several steps. Firstly, an in depth analysis of the literature has been carried out to collect the information requirements for the profile. Secondly, a domain model for dependability analysis was defined independently of UML. Thirdly, the domain model was mapped to UML extensions by specializing MARTE.

1 Introduction

The *dependability* of a system, as defined in [1] is the ability to avoid failures that are more frequent and more severe than acceptable. The dependability encompasses a set of non-functional properties, called *attributes* of dependability, such as: a) *availability*, the readiness for correct service; b) *reliability*, the continuity of correct service; c) *safety*, the absence of catastrophic consequences on the users and environment; d) *maintainability*, the ability to undergo modifications and repairs.

* This work has been supported by the European IST project CRUTIAL-027513 (CRitical UTility InfrastructurAL resilience), the project DPI2006-15390 of the Spanish ministry of Science and Technology, and the Discovery grant from the Natural Sciences and Engineering Research Council of Canada (NSERC).

K. Czarnecki et al. (Eds.): MoDELS 2008, LNCS 5301, pp. 736–750, 2008.

Although there are several proposals in literature on extending Unified Modeling Language (UML) models with dependability annotations, as reviewed in Sect. 3 of this paper, each covers only a subset of dependability aspects. Compared with the performance and schedulability analysis domains, which are supported by standard UML profiles such as Schedulability, Performance and Time (SPT) [2] and MARTE [3], there is no similar standard profile for the dependability analysis of UML-based models yet. Another Object Management Group standard specifying UML extensions for a variety of non-functional properties, the "Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms" (QoS&FT) [4], provides a flexible but heavy-weight mechanism to define properties such as performance, security or reliability by means of specific QoS catalogs. The annotation mechanism is supported by a two-step process, which implies catalog binding and either the creation of extra objects just for annotation purposes, or the specification of long Object Constraint Language (OCL) expressions.

The main objective of this paper is to propose a UML profile for quantitative dependability analysis of software systems modeled with UML with particular focus on the following facets of dependability: reliability, availability and safety. One of the requirements for such a profile is to reuse the best practices reported in literature, and to unify the terminology and concepts for different dependability aspects under a common dependability domain model. Another requirement is to be compliant with the recently adopted MARTE profile [3] in two ways: a) by using the Non-Functional Properties (NFP) framework and its corresponding Value Specification Language (VSL) for defining dependability-specific data types necessary for the profile definition; and b) by specializing general concepts from MARTE's generic quantitative analysis model (i.e., GQAM sub-profile) for the dependability analysis domain. This paper builds on the previous work [5], where a first attempt to extending UML with dependability analysis capabilities was proposed.

We have adopted a systematic approach for the definition of the dependability profile according to the recommendations from Selic [6] (see also Lagarde et al.[7]). Firstly we define a conceptual domain model for dependability analysis and modeling, which covers different dependability aspects, and reuses and unifies the concepts from previous work, and secondly we map the domain concepts to elements of a UML profile. The new proposed stereotypes extend either UML meta-classes or MARTE stereotypes, and the stereotype attribute definitions use dependability-specific types defined in VSL. Our work proposes a unified DAM profile, but does not discuss the different model transformation techniques for generating different dependability models (Markov chains, Fault Tree, Petri Nets, etc.). The proposed profile provides all the information necessary to produce such models, but the actual model transformations represents future work.

The paper is organized as follows: Sect. 2 gives an overview of the approach followed for the profile definition; Sect. 3 gives a brief survey of related work from literature; Sect. 4 introduces the dependability analysis (DA) conceptual

model, composed of different packages that cover separate dependability aspects; Sect. 5 presents the proposed dependability profile; Sect. 6 gives an example of profile application and Sect. 7 presents the conclusions.

2 Approach Overview

This section presents the process followed to define the dependability analysis profile, in order to ensure that we produced a technically correct quality UML profile that covers the necessary concepts according to the best practices reported in literature.

Literature Review. The existing standard UML profiles for the analysis of non functional properties of software systems have been analyzed, in particular the SPT [2], QoS&FT [4] and MARTE [3]. None of them provides a comprehensive support for dependability analysis, especially from a quantitative point of view. We investigated the literature on dependability main concepts and taxonomy (e.g., Avizienis et al. [1], Leveson [8], Lyu [9,10]) as well as on standard methods used for the quantitative assessment of dependability (e.g., [11,12]). We also surveyed the works from the literature proposing dependability modeling and analysis of UML system specifications (about twenty papers). The output of this preliminary step is a checklist of information requirements that a UML profile for dependability analysis should satisfy reported in detail in [13].

Definition of Dependability Analysis (DA) Conceptual Model. We defined a DA conceptual model to represent the main dependability concepts from the literature. Its construction required several refinement steps to consider all the surveyed works. The final domain model is described in Sect. 4.

Completeness Assessment of the DA Domain Model. We verified whether all the concepts considered in the survey have been included. If a concept was not considered, we either repeated the refinement step or provided a motivation for its exclusion.

Definition of the Dependability Analysis Modeling (DAM) Profile. Using the DA conceptual model we defined: a) the DAM extensions (stereotypes and tags), and b) DAM library containing dependability specific types. The objective is to introduce a small set of stereotypes that can be easily used by the software analyst. The DAM library has been defined by importing the MARTE library and consists of basic and complex DA types.

DAM Profile Assessment. We verified whether the information requirements from [13] are satisfied. If a requirement was not met, we went back to the previous step in order to refine it.

3 Related Works

The brief survey from this section focuses on existing work from literature that provides support for the quantitative dependability analysis of UML-based designs. (A more detailed survey can be found in [13]).

Pataricza [14] extends the General Resource Modeling package of the SPT profile with the notion of faults and errors to support the analysis of the effect of local faults to the system dependability. The work includes permanent and transient faults in the resources and error propagation to estimate which fault may lead to a failure. Explicit fault injection behavioral models are also proposed to represent faults as special virtual clients.

Addouche et al. [15] define a profile for dependability analysis of real-time systems compliant with the SPT resource modeling. The UML extensions are used to derive probabilistic time automata for the verification of dependability properties via temporal logic formulas. The static model of the system is enriched with new stereotyped classes associated with resources (but has the disadvantage that new classes are introduced in the system model for dependability analysis purposes).

Bernardi et al. [16] propose a set of UML Class Diagrams structured in packages for collecting dependability and real-time requirements and properties of automated embedded systems with the use of COTS fault-tolerance (FT) mechanisms. The approach provides support for a semi-automatic derivation of dependability analysis models, such as Stochastic Petri Nets and temporal logic. In [17] is proposed a method to assess the quality of service of FT distributed systems by deriving performability models from UML+SPT models.

The most comprehensive approach so far for reliability and availability analysis of UML specifications has been proposed by Bondavalli et al. [18]. A profile for annotating software dependability properties compliant with the taxonomy and basic concepts from [1] is proposed. A model transformation process derives Timed Petri Net models via an intermediate model from the annotated UML models. The approach supports the specification of error propagation between components, as well as independent and dependent failures. It is possible to discriminate between normal and failure states and events. The main drawback of this work is the introduction of unnecessary redundant information in the UML model, as sometime the joint use of more than one stereotype is needed.

DalCin [19] proposes a UML profile for specifying dependability mechanisms, aimed at supporting the quantitative evaluation of the FT strategy effectiveness. However, the profile lacks support for modeling the interactions between dependability mechanisms and system components. Pai and Dugan [20] present a method to derive dynamic fault tree from UML system models. The method supports the modeling and analysis of sequence error propagations that lead to dependent failures, redundancies and reconfiguration activities.

The papers [21,22,23] address specifically the reliability analysis of UML-based design. D'Ambrogio et al. [21] define a transformation of UML models into fault tree models to predict the reliability of component-based software. Cortellessa and Pompei [22] propose a UML annotation for the reliability analysis of component-based systems, within the frameworks of the SPT and QoS&FT profiles. The annotations defined in [22] are used by Grassi et al. [23] where a model-driven transformation framework for the performance andreliability analysis of component-based systems is proposed. The method uses an intermediate model

that acts as bridge between the annotated UML models and the analysis-oriented models. In particular, discrete time Markov process models can be derived for the computation of the service reliability.

Jürjens et al. define a safety [24] and reliability [25] check list, based on UML extension standard mechanisms, to support the identification of failure-prone components in the software design.

The approaches [26,27,28] support the safety analysis of UML-based system models. Pataricza et al. [26] use UML stereotypes to identify erroneous states and error correcting transitions in state machine diagram, integrating the normal and the faulty behavior of a system component in a single state machine. Goseva et al. [27] devise a methodology for the risk assessment of UML models at architectural level; a Markovian model is constructed to estimate the scenario risk factor from risk factors associated to software components and connectors. Hassan et al. [28] introduce a methodology for the severity analysis of software systems modeled with UML, which integrates different hazard analysis techniques (Functional Failure Analysis - FFA, Failure Modes and Effects Analysis - FMEA - and Fault Tree Analysis - FTA) to identify system level and component/connector level hazards and to evaluate the cost of failure of system execution scenarios, software components and connectors.

4 Dependability Analysis Conceptual Model

The DA conceptual model has been constructed considering the main dependability concepts from the literature as well as standard methods used for the dependability assessment. It is organized into a set of packages, as shown in Fig. 1. The top-level package includes:

- The *System Core* model. It provides a description of the system to be analyzed, according to a component-based view of the system [1,9]. The model includes also additional concepts for the description of redundancy structures that may characterize a fault tolerant system [10].
- The *Threats* model introduces the threats [1,9,8] that may affect the system at different levels as well as the relationships between the threats.
- The *Maintenance* model introduces the repair/recovery actions that are undertaken in case of repairable systems [1,10].

Figures 2,3,4, and 5 show the Class Diagrams of the DA conceptual model; the markups used for the classes will be explained in Sect. 5. The *Core* model (Fig. 2) represents the *context* for carrying out dependability analysis. Actually, it is a component-based description of the system to be analyzed that includes both structural and behavioral views. From the structural point of view, the system consists of a set of *components* that are bound together through *connectors*, i.e., logical/physical links, in order to interact. A component can be a sub-system consisting of sub-components. The structure of the system is what enables it to generate the behavior. The system delivers a set of high-level *services*, in response to user *service requests*. Each high-level service is, then, the system

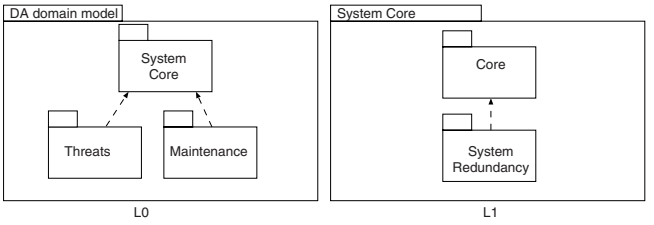

Fig. 1. Top-level package (L0), System Core package (L1)

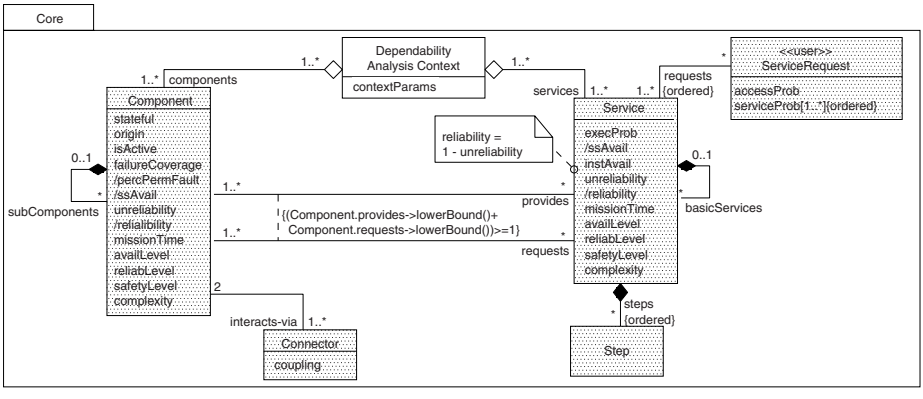

Fig. 2. Core model

behavior as perceived by its users and it is carried out by the interaction of system components, which provide and request basic services to each others. A service is implemented by a sequence of *steps* that represent component states and actions.

The *Core* model acts as a bridge between the DA concepts and the concepts introduced in MARTE for general quantitative analysis. Indeed, several classes of the *Core* model will be mapped to stereotypes that specialize the ones introduced in the GQAM profile.

Observe that some classes of the DA conceptual model have attributes that represent requirements, metrics, properties or input parameters used in dependability analysis, according to the surveyed works from literature. A detailed description of the meaning of the attributes and their types is given in [13].

A system may be characterized by redundancy structures. Software and hardware redundancy are the typical means used to increase the FT of software systems, e.g., by eliminating single points of failure. The *System Redundancy* model (Fig. 3) represents *FT components* [10], e.g., used in [18] to specify the role played by a component within a *redundant structure*. In particular, a redundant structure may consist of several *variants*, i.e, modules with different design that provide the same services, allocated over different *spares*, a *controller*, that is responsible for the co-ordination of the variants, an *adjudicator*, that either

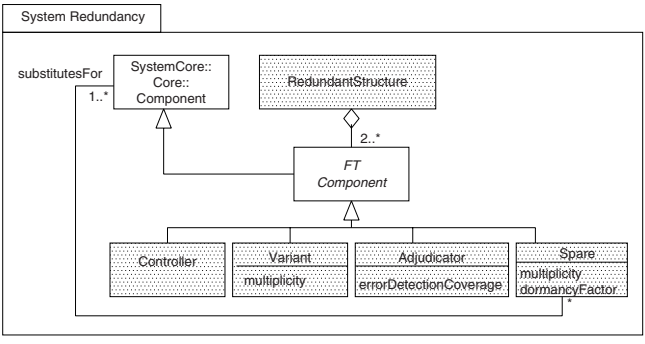

Fig. 3. System Redundancy model

looks for a consensus of two or more outputs among the variants ("N-version software" scheme) or applies an acceptance test to the variant outputs ("recovery block" scheme).

Note that it is out of scope of this work to provide a support for the modeling of FT architectures and, besides, this issue has been tackled by the QoS&FT profile [4]. Rather, the introduction of the *System Redundancy* model is motivated by the objective of providing a specific support for the quantitative dependability analysis of those FT systems characterized by redundant structures.

The *Threats* model (Fig. 4) includes the threats that may affect the system, namely the *faults, errors, failures* [1,9] and *hazards* [8]. We have introduced an abstract concept of *impairment*, that can be specialized depending of the type of analysis domain, i.e., *failure* for reliability/availability analysis and *hazard* for safety. The model represents also the cause-effect relationships between the threats and the relationships between the threats and the system core concepts. Then, a fault is the original cause of errors and impairments, and affects system components. A *fault generator* concept is added to represent a mechanism, used for example in [17,14], to inject faults in the system and to specify the max number of concurrent faults.

Errors are related to steps (i.e., states or actions) of the basic services provided by the faulty components. When an error affects an external state of a faulty component, that is the service interface of that component, then *error propagations* may occur from the faulty component to the interacting ones, via the corresponding connectors. Error propagations may be related to each others, for example they occur according to a given order [20]: the *error propagation relation* indicates the sequence of error propagations. Errors may cause impairments at different system level: 1) at service step level, then leading to *failure/hazard steps*, when the service provided by the component becomes not correct, 2) at component level, when the component is not able to provide any basic service, 3) at system level, when the impairment is perceived by the system users. Finally, multiple dependent impairments can affect a redundant structure, such as when several redundant components fail in the same mode (i.e., *common mode failures* [10]).

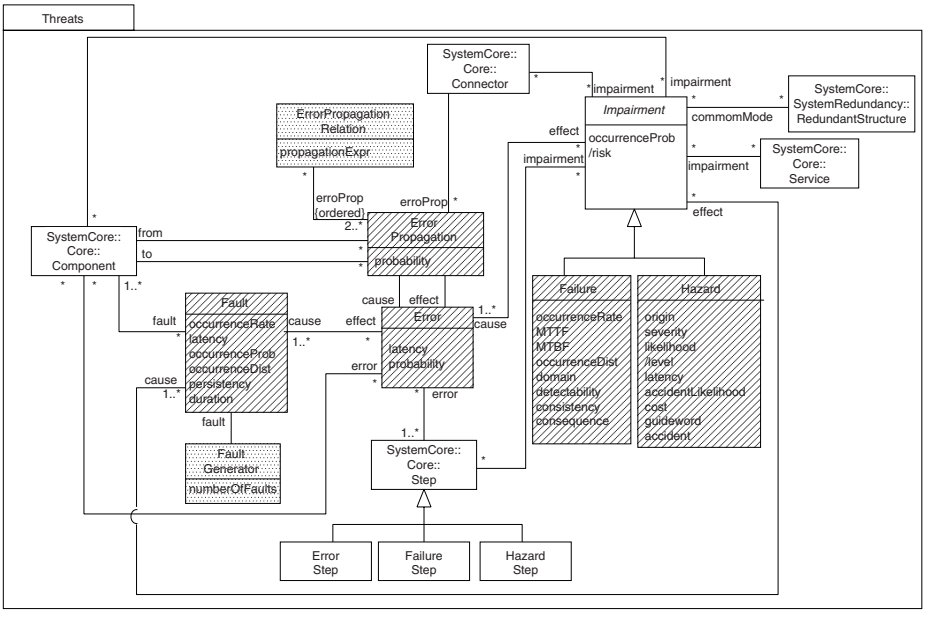

Fig. 4. Threats model

The *Maintenance* model (Fig. 5) 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 [1,10], we distinguish *repairs* of system components, that involve the participation of external agents (e.g., repairman, test equipment, etc) and *recovery* strategies, usually implemented in FT systems, that aim at transforming the system anomalous states into correct states. In particular, the reconfiguration steps imply the use of spare components [20]. The model represents the *replacement steps*, in which faulty components are replaced by spares, and the *reallocation steps*, in which software components are reallocated onto spares.

5 Dependability Analysis Modeling Profile

The process of mapping the conceptual model elements to profile elements has been an iterative process, in which each class has been examined, together with its attributes, associations and constraints, to identify the most suitable UML base concepts for it. Such activity has been carried out by following the general guidelines in [6] and by specializing the UML extensions of the MARTE profile [3].

Figure 6(a) shows an overview of the DAM profile. It consists of an UML extension package, including the set of dependability stereotypes and attributes, and of a model library, in which basic and complex dependability types associated to the attributes are defined. Most of the dependability stereotypes specialize the ones of the GQAM sub-profile of MARTE.

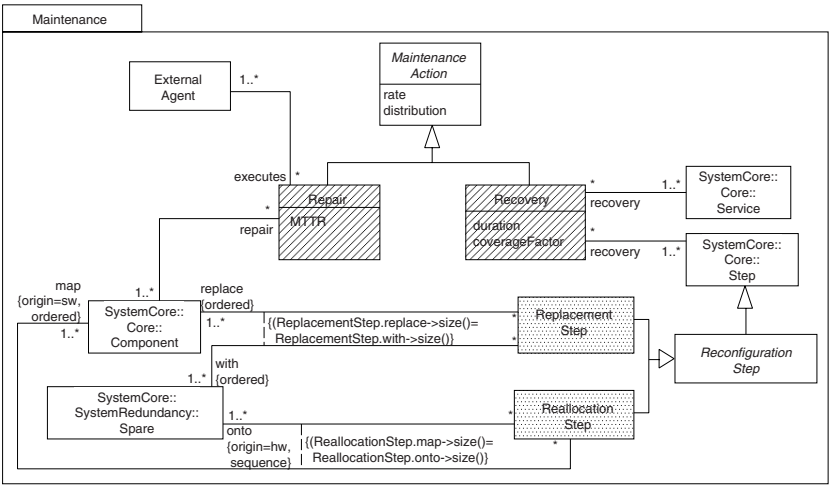

Fig. 5. Maintenance model

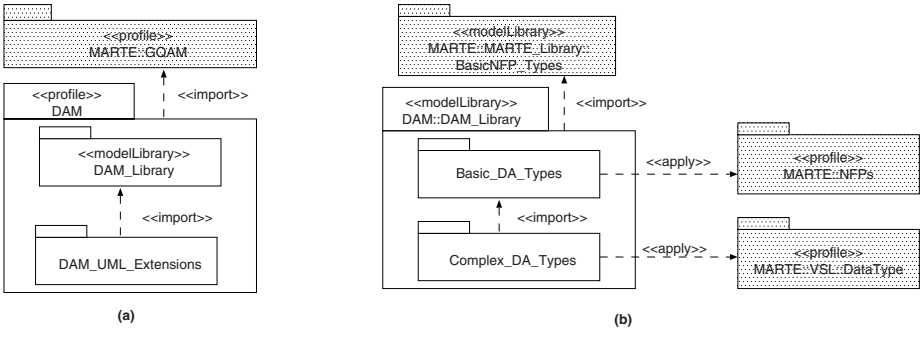

Fig. 6. (a) DAM profile and (b) DAM library

A low level view of the *DAM Library* is shown in Fig. 6(b). Complex dependability types are MARTE data-types which combine different basic NFP types, from MARTE library, and/or basic dependability types. On the other hand, basic dependability types can be either simple enumeration types (such as *CriticalLevel* used to specify a failure consequence or an hazard severity) or, in turn, specialization of basic NFP common types. An example, of the latter is the *DaFrequency*, which is a basic NFP real type, introduced to specify, e.g., a failure occurrence rate as a real value together with a failure frequency unit (e.g., 0.1^{-2} failures per hour). We have also applied the MARTE NFPs profile to define new dependability types, e.g., *DaFrequencyUnitKind* which is an enumeration type including a set of frequency units of fault/failure occurrence rates and of repair/recovery rates (see Fig. 7(c)).

The specialization of the concept of basic NFP common type allowed us to reuse several properties of the super-type that enrich the annotation capabilities

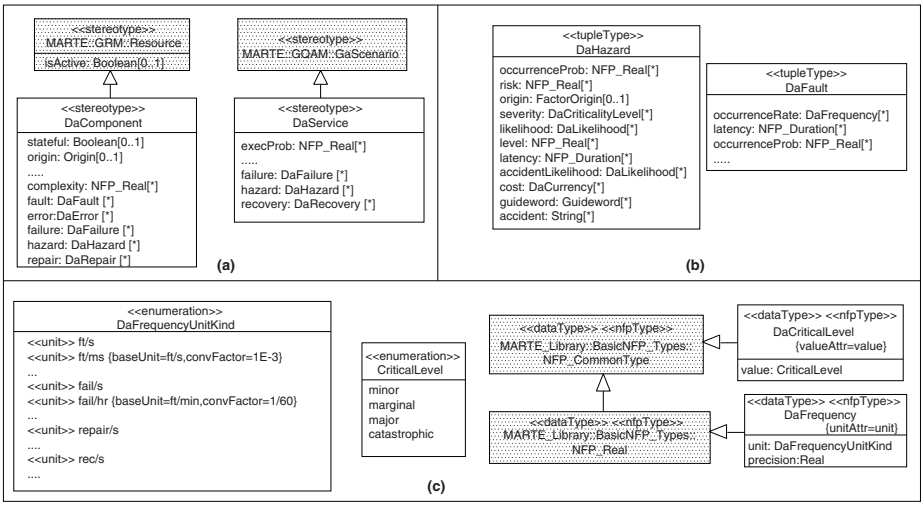

Fig. 7. (a) stereotypes, (b) complex dependability types, (c) basic dependability types

at system model level, such as the *expression* property, that supports the specification of expressions using the VSL syntax, and the *source* property that can be used to define the origin of the specification (e.g., a requirement, a metric or an input parameter).

In the DAM profile definition, we have applied several suggestions and patterns proposed in [7] that enable the creation of a profile from the conceptual model which is consistent with the UML meta-model. Moreover, we adopted the best practise of MARTE to keep track of the mapping between the conceptual model and the DAM profile. In particular, the name of each DAM extension (stereotype, complex/basic dependability type) is the name of the corresponding conceptual model element prefixed by *Da*, namely Dependability Analysis.

Domain classes are good candidates to become stereotypes but, eventually, only a subset of them have been mapped to a stereotype (i.e., the dotted classes in Figs. 2, 3, 4 and 5). Indeed, as in [7], we aimed at providing a small set of stereotypes that will be actually going to be used in practical modeling situations. Then, abstract classes, e.g., *FT Component* (Fig. 3), have not been considered in the mapping process. Nevertheless, if an abstract class carries information and it is specialized with concrete classes, such information has been considered in the mapping of the latter as well as of the concrete classes associated with the abstract class. This is the case, for example, of the abstract class *Impairment*, where its attributes have been mapped to attributes of the complex dependability types *DaFailure* and *DaHazard*. Moreover, the association-ends *impairment* have been renamed and mapped to attributes of the stereotypes *DaComponent, DaService* corresponding to concrete classes associated with the abstract class (compare Figs. 4 and 7(a,b)). On the other hand, the classes modeling threats and maintenance actions (i.e., classes with diagonal stripes in Figs. 4 and 5) have been defined to characterize complex dependability concepts through their

attributes. Then, such classes have been mapped to complex dependability types of the DAM library (Fig. 7(b)).

Each stereotype extends either a set of UML meta-classes or MARTE stereotype. To define extension associations for a given stereotype, we have applied the general guidelines in [6], based on similarity of the semantics of the UML meta-classes and of the stereotype. To facilitate the extension process we have exploited the proposals in the surveyed literature, to identify the UML model elements annotated with the same dependability properties as the ones characterizing the stereotype. Finally, if a semantically equivalent stereotype exists in MARTE, then we have defined the dependability stereotype as a sub-stereotype of the former.

Attributes of a conceptual class have been mapped to attributes of the corresponding UML extension (stereotype/complex dependability type). This mapping activity implied the definition of a basic dependability type for each attribute as well as the definition of its multiplicity. The mapping of associations has been less trivial than that of attributes. We have often applied the *reference association* pattern of Lagarde et al.[7]. An exemplification of such pattern is given in Fig. 4, where the *Component* class is characterized by an association with the *Fault* class supported by the association-end *fault*. The latter is used to define the attribute *fault* of *DaComponent*, of complex dependability type *DaFault* (Fig. 7(a)). Observe that association multiplicities have been mapped to attribute value multiplicities.

It is worth to note that the conceptual model is characterized by several constraints written in OCL. Such constraints have been assigned to the DAM extensions and represent constraints for the use of the profile at model specification level. An excerpt of the DAM profile is depicted in Fig. 7: due to space limitation, not all the attributes are shown. The whole set of DAM extensions can be found in [13].

6 Usage of the DAM Profile

The purpose of this section is to illustrate, through an example, the usage of the DAM profile, as well as to demonstrate how an established methodology [27]

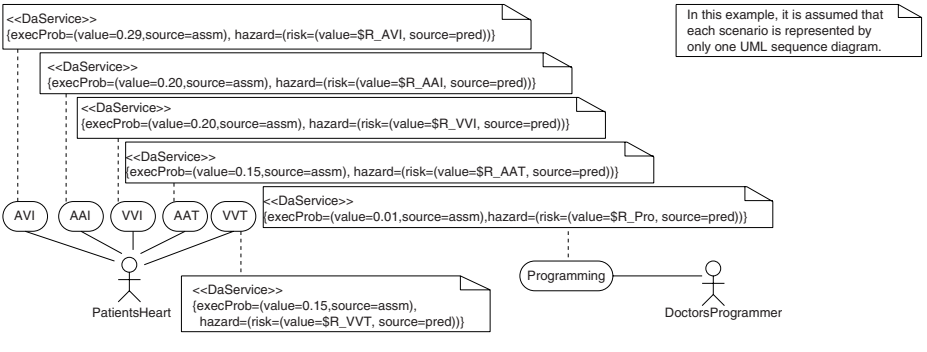

Fig. 8. Use case diagram of the pacemaker

for dependability analysis benefits from the DAM profile. The methodology for risk assessment proposed by Goseva et al. [27] introduces the safety related parameters in a tabular form, i.e., no UML extensions are provided. This section shows how these parameters can be added to a UML diagram using the DAM profile, while the methodology steps proposed in [27] (which cannot be presented here in detail due to space limitations) can still be applied.

Figure 8 depicts the use case diagram of the pacemaker example used in [27] to illustrate the methodology. Each use case (UC) is later realized as a (set of) UML sequence diagram(s) representing system *scenario(s)*. We have attached to the UCs, using the DAM profile, the original annotations that were given by the authors in the tabular form. Note that the UCs are stereotyped as *DaServices*. So, the corresponding attributes can properly describe their dependability properties. As an example, see the annotation in the AVI UC, where its execution probability `execProb`, i.e., a software requirement describing an input dependability model parameter (`source=assm`), is gathered in the UML model. Also, it can be observed how the dependability metrics to be computed in the model (i.e., the hazard risk factors) are considered as predicted values (`source=pred`). Using the VSL syntax, variable names are preceded by the $ symbol, as the $R_AVI variable.

Figure 9 shows the components and connectors that make up the pacemaker architecture[1]. In [27], the dependability input parameters *complexity and coupling* are calculated from the software functional requirements. Indeed, the *coupling* of a connector is derived from the UML SD, while the *complexity* of a component is obtained from its associated state-chart. On the other hand, the characterization of the hazards (*severity, guideword, accident*) affecting a component/connector is the result of the FMEA technique, carried out by a domain expert.

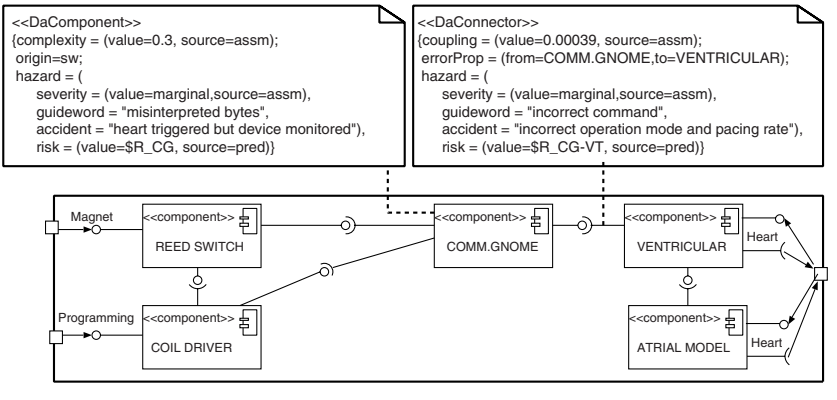

Fig. 9. The architecture of the pacemaker

For each scenario, the dependability metrics (*risk factors*) are estimated as a product of the *complexity* (*coupling*) of the component (connector) and the

[1] To avoid cluttering, only one component and one connector have been annotated.

hazard *severity* (the authors map the *marginal* severities to 0.5). The methodology ranks the critical components/connectors using such information. Moreover, considering these component (connector) risk factors together with a "control flow graph" obtained from the functional software requirements, the methodology is enabled to construct a Markov model for each scenario, where the scenario risk factor can be computed. Finally, the scenario risk factors are used to get the UCs risk factors as well as the overall system risk factor.

In conclusion, we observe that the DAM profile covers all the safety input parameters and metrics proposed in [27]. Since the DAM profile is compliant with the UML standard, it is possible now to enter all the annotations required for the Goseva methodology by using any standard UML tool. The only additional effort is the implementation of the model transformation which will take as input UML models with DAM annotations and produce as output the Markov chain model, following the approach proposed in [27].

7 Conclusion

In this paper we have proposed a profile to support dependability modeling and analysis of UML designs. The proposed profile is compliant with the standard MARTE profile, and has been built considering current dependability standards. We have defined the profile by following the approach proposed by Selic [6] and applying the patterns from Lagarde [7]. To the best of our knowledge, this is the first attempt to provide a common conceptual UML model for different dependability communities. We consider the profile as an open proposal, subject to future refinements and extensions to address particular issues in the different dependability domains (e.g., FT systems). Several reasons let us envision a great potential for our proposal. First, the existing UML-based approaches for analyzing dependability aspects of software systems surveyed in Sect. 3, can get direct benefit from the profile. Indeed, our proposal offers a common modeling support intended to covers all the information required by the existing methodologies. As an example, we have illustrated how one of these proposals [27] can take advantage of the profile. Future work includes the implementation of the model transformations that take as input UML+DAM models and produce different dependability models, by following the same approach as in the existing proposals. On the other hand, the works that focus exclusively on modeling dependability as opposed to analyzing it, can also be benefit from the profile by using its annotations or even by extending it. Among such works, there are methodologies for collecting dependability requirements, e.g. safety domain requirements in [29]; others target certification according to standard software requirements [30]; and finally, there are traditional works in dependability analysis of software systems outside the UML umbrella. In the last case, the challenge is not how to integrate the DAM profile, but rather how to integrate the UML within the respective methodology.

References

1. 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)
2. Object Management Group: UML Profile for Schedulability, Performance and Time Specification. (January 2005) V1.1, f/05-01-02
3. Object Management Group: A UML profile for Modeling and Analysis of Real Time Embedded Systems, Beta 1. (August 2007) Adopted Spec., ptc/07-08-04
4. Object Management Group: UML Profile for Modeling Quality of Service and Fault Tolerant Characteristics and Mechanisms. (April 2008) V1.1, f/08-04-05
5. Bernardi, S., Merseguer, J.: A UML profile for dependability analysis of real-time embedded systems. In: Proc. of WOSP, February 2007, pp. 115–124. ACM, New York (2007)
6. Selic, B.: A systematic approach to domain-specific language design using UML. In: 10th IEEE Int.l Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC 2007), pp. 2–9 (2007)
7. Lagarde, F., et al.: Improving UML profile design practices by leveraging conceptual domain models. In: 22nd Int.l Conf. on Automated Software Engineering, Atlanta (USA), November 2007, pp. 445–448. ACM, New York (2007)
8. Leveson, N.: Safeware. Addison-Wesley, Reading (1995)
9. Lyu, M.R. (ed.): Handbook of Software Reliability Engineering. IEEE Computer Society Press, Los Alamitos (1996)
10. Lyu, M.: Software Fault Tolerance. John Wiley & Sons, Ltd., Chichester (1995)
11. Commission, I.E.: IEC-60300-3-1 standard: Dependability management
12. Commission, I.E.: IEC-61508 standard: Functional Safety of Electrical/ Electronic/ Programmable Electronic safety related problems
13. Bernardi, S., Merseguer, J., Petriu, D.: An UML profile for Dependability Analysis and Modeling of Software Systems. Technical Report RR-08-05, Universidad de Zaragoza, Spain (2008), `http://www.di.unito.it/~bernardi/DAMreport08.pdf`
14. Pataricza, A.: From the General Resource Model to a General Fault Modelling Paradigm? In: Workshop on Critical Systems, held within UML 2000 (2000)
15. Addouche, N., Antoine, C., Montmain, J.: UML models for dependability analysis of real-time systems. In: Proc. International Conference on Systems, Man and Cybernetics, October 2004, vol. 6, pp. 5209–5214. IEEE Computer Society, Los Alamitos (2004)
16. Bernardi, S., Donatelli, S., Dondossola, G.: A class diagram framework for collecting dependability requirements in automation systems. In: Proc. of 1^{st} Int.l Symposium on Leveraging Applications of Formal Methods, Cyprus (October 2004)
17. Bernardi, S., Merseguer, J.: QoS Assessment via Stochastic Analysis. IEEE Internet Computing, 32–42 (May-June 2006)
18. Majzik, I., Pataricza, A., Bondavalli, A.: Stochastic Dependability Analysis of System Architecture Based on UML Models. In: Architecting Dependable Systems. LNCS, vol. 2677, pp. 219–244. Springer, Heidelberg (2003)
19. Dal Cin, M.: Extending UML towards a Useful OO-Language for Modeling Dependability Features. In: Proc. of 9th Int.l Workshop on Object-Oriented Real-Time Dependable Systems, Capri Island, Italy, October 2003, pp. 325–330. IEEE Computer Society, Los Alamitos (2003)
20. Pai, G., Dugan, J.: Automatic Synthesis of Dynamic Fault Trees from UML system models. In: Proc. of 13th Int. Symposium on Software Reliability Engineering, Annapolis, MD, USA, November 2002, pp. 243–256. IEEE Computer Society, Los Alamitos (2002)

21. 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), Cambridge, MA, USA (November 2002)
22. Cortellessa, V., Pompei, A.: Towards a UML Profile for QoS: a contribution in the reliability domain. In: Proceedings of the Fourth International Workshop on Software and Performance (WOSP 2004), pp. 197–206 (January 2004)
23. Grassi, V., Mirandola, R., Sabetta, A.: Filling the gap between design and performance/reliability models of component-based systems: A model-driven approach. Journal of Systems and Software 80(4), 528–558 (2007)
24. Jürjens, J.: Developing safety-critical systems with UML. In: Stevens, P., Whittle, J., Booch, G. (eds.) UML 2003. LNCS, vol. 2863, pp. 360–372. Springer, Heidelberg (2003)
25. Jürjens, J., Wagner, S.: Component-based Development of Dependable Systems with UML. In: Atkinson, C., Bunse, C., Gross, H.-G., Peper, C. (eds.) Component-Based Software Development for Embedded Systems. LNCS, vol. 3778, pp. 320–344. Springer, Heidelberg (2005)
26. Pataricza, A., et al.: UML-based design and formal analysis of a safety-critical railway control software module. In: Tarnai, G., Schnieder, E. (eds.) Proc. of FORMS 2003, Budapest (Hungary), pp. 125–132 (May 2003)
27. Goseva-Popstojanova, K., et al.: Architectural-level risk analysis using UML. IEEE Transactions on Software Engineering 29(10), 946–960 (2003)
28. Hassan, A., Goseva-Popstojanova, K., Ammar, H.: UML Based Severity Analysis Methodology. In: Proc. of Annual Reliability and Maintainability Symposium (RAMS 2005), Alexandria, VA (January 2005)
29. Allenby, K., Kelly, T.: Deriving safety requirements using scenarios. In: 5th IEEE International Symposium on Requirements Engineering (RE 2001), pp. 228–235. IEEE Computer Society, Los Alamitos (2001)
30. Zoughbi, G., Briand, L., Labiche, Y.: A UML Profile for Developing Airworthiness-Compliant (RTCA DO-178B), Safety-Critical Software. In: Engels, G., Opdyke, B., Schmidt, D.C., Weil, F. (eds.) MODELS 2007. LNCS, vol. 4735, pp. 574–588. Springer, Heidelberg (2007)