A UML Profile for Dependability Analysis of Real-Time Embedded Systems

Simona Bernardi[*]
Dipartimento di Informatica
Università di Torino, Italy
bernardi@di.unito.it

José Merseguer[†]
Dpto. de Informática e Ingeniería de Sistemas
Universidad de Zaragoza, Spain
jmerse@unizar.es

ABSTRACT

In this paper, we aim at giving a contribution toward the definition of a UML profile supporting the dependability analysis of real-time and embedded systems (RTES) that conforms to the upcoming profile named "Modeling and Analysis of Real-Time and Embedded Systems" (MARTE), for which a Request For Proposal has been issued by the Object Management Group (OMG).

A set of basic dependability and fault-tolerance concepts need to be included in the profile to support the dependability analysis of RTES. We have exploited the best practices, proposed in the literature, on extending UML with dependability modeling capabilities in order to draw up a check list of requirements to be used as guideline for the definition of a dependability analysis profile. The proposed profile is then applied to the UML design of a case study: a gas turbine control system.

Categories and Subject Descriptors

D.2.1 [**Software Engineering**]: Requirement/Specification; D.2.8 [**Software Engineering**]: Metrics

General Terms

Design, Reliability, Standardization

Keywords

UML profiles, dependability, real-time embedded systems

1. INTRODUCTION

Within the Unified Modeling Language (UML), two profiles are currently available to support the assessment of non-functional properties of software systems: the Schedulability, Performance and Time (SPT) profile [19] and the Quality of Service and Fault Tolerance Characteristics and Mechanisms (QoS&FT) [21] profile.

The SPT profile has been the first attempt to extend UML with basic timing and concurrency concepts and to express requirements and properties for conducting schedulability and performance analysis. The more recent QoS&FT profile has a broader scope, its objective is to allow the user to define a wider variety of QoS requirements and properties. Nevertheless, as emerges from the comparative analysis carried out in [5], the QoS&FT requires too much effort for the final users (software analyst, designer) to be applied and, being more recent, only few application examples are given, mainly addressed to the schedulability analysis.

Dependability assessment of software systems, as performance and schedulability, should be carried out early in the software lifecycle to avoid unacceptable costs in terms of loss of life and resources due to system failures. Although the sub-profile proposed in the QoS&FT can be used for the specification of the fault tolerant software architectures, nor the QoS&FT profile neither the SPT profile specifically support the dependability analysis of UML based system models.

The OMG has recently issued a request for proposal (RFP) for a new UML profile for "Modeling and Analysis of Real-Time and Embedded Systems" (MARTE) in order to upgrade the SPT profile to UML2.0 [20] and to extend its scope with real-time embedded system (RTES) modeling capabilities. The work [14] provides a flexible and straightforward framework for MARTE by adopting the best modeling practice of the SPT and QoS&FT, and proposes a domain model for annotating non functional properties to support temporal verification of UML based models.

In this article, we aim at giving a contribution toward the definition of a UML profile for the dependability assessment of RTES which is compliant with the MARTE RFP and uses the general framework proposed in [14]. We have exploited the best practices, proposed in the literature, on extending UML with dependability modeling capabilities in order to draw up a check list of requirements to be fulfilled by a profile supporting the different facets of dependability analysis.

We then propose several complex non functional properties that cover most of the dependability aspects dealt in the considered literature and that characterize a set of stereotypes: the new UML extensions introduced are structured according to [14], then providing a skeleton of the Dependability Analysis (DA) profile. The profile can be used to enrich the UML models with dependability requirements and properties and aims at supporting the transformation of the UML-DA annotated models into suitable dependability models (such as fault trees, Bayesian networks, stochastic Petri Nets) for the quantitative assessment of the system.

[*]Simona Bernardi has been supported by the European IST project CRUTIAL-027513 (CRitical UTility InfrastructurAL resilience)

[†]José Merseguer has been supported by the project IBE2005-TEC-10 of the University of Zaragoza

The contribution of this work is the definition of the skeleton of the DA profile and its application to the case study of a gas turbine control system. The issues related to the derivation of dependability models from UML-DA annotated models as well as their analysis are not discussed here and they will be subject of future research.

The paper is organized as follows: in Section 2 we recall the basic concepts of dependability and the commonly used approaches to dependability analysis of software systems. In Section 3 we analyze the different proposals made in the literature for the annotation of dependability requirements and characteristics in UML based models. A check list of basic concepts that should be included in a dependability analysis profile is then derived. Section 4 describes the proposed Dependability Analysis (DA) profile that conforms to the MARTE request for proposal. In Section 5 we use the DA profile to specify the dependability requirements and properties in the UML design of a gas turbine control system for safety and availability analysis purpose. Conclusions and open issues are given in Section 6.

2. DEPENDABILITY BASIC CONCEPTS

We refer to the work [2] for the definition of basic concepts and taxonomy of dependability. We briefly summarize in the following those key concepts that should be considered in the definition of a dependability analysis profile.

The concepts defined in [2] are based on a *component-based system* view, that is a system is an entity that interacts with other entities (i.e., other systems, including hardware and software, and the physical world with its natural phenomena) and it is composed of a set of components bound together in order to interact. Each component can in turn be another system and, at the lowest nesting level, is atomic that is any further internal structure cannot be discerned or is not of interest and can be ignored.

The service delivered by the system is its behavior as it is perceived by its users, and a service *failure* represents an event that occurs when the delivered service deviates from the fulfillment of the intended system functionalities. Service failures manifest in different modes that are classified according to several viewpoint; considering, for example, the failure domain, we can distinguish content failure, when the information delivered by the system is not correct, and timing failure, when the information is delivered too late or too early.

A failure may occur when at least a state of the system deviates from the correct service state, such deviation is called *error*. Errors can be caused by *faults* that, as for failure concept, can be classified according to different viewpoints, such as the software development phase in which their occur (i.e., development/ operational faults), the timing persistence (i.e., permanent, transient and intermittent faults), the system components they affect (i.e., hardware/ software faults). Faults, errors and failures are called *impairments* of dependability.

The *dependability* of a system is the ability to avoid failures that are more frequent and more severe than is acceptable. The dependability encompasses a set of system properties, called *attributes* of dependability, listed in the following:

- *Availability*, the readiness for correct service;
- *Reliability*, the continuity of correct service;
- *Safety*, the absence of catastrophic consequences on the users and environment;
- *Integrity*, the absence of improper system alterations;
- *Maintainability*, the ability to undergo modifications and repair.

There are different *means* to attain dependability. The most interesting toward the definition of a dependability profile, from our point of view, are *fault tolerance* and *fault forecasting*.

Fault tolerance aims at avoiding system failures despite the presence of faults and it is carried out via *error detection* and *system recovery*. Error detection techniques aim at identifying the presence of errors in the system and different types of mechanisms can be implemented for this purpose, such as run-time checks (e.g., hardware overflow and division by zero mechanisms, software checks that raise exception), timing checks (e.g., watchdog timers) and coding checks based on information redundancy. System recovery techniques are instead used to bring the system state from erroneous, or faulty, to a functioning one, without detected errors or latent faults. Among such techniques we can mention the ones used for masking errors through the systematic usage of spatial and/or temporal redundancy (e.g., replicated computation and voting) and reconfiguration. The measure of the effectiveness of a fault tolerance solution is called *coverage*, in particular the coverage factor is the probability of system recovery given that a fault has occurred in the system [23].

On the other hand, fault forecasting is carried out through the qualitative and quantitative evaluation of the system behavior with respect to faults occurrences. Qualitative evaluation consists in identifying, classifying and rating the system failure modes. Quantitative evaluation, instead, aims at computing dependability metrics, through modeling and testing, using probabilistic assumptions. The methods used for fault forecasting [8] can be either specific (e.g., Failure Mode and Effect Analysis for the qualitative assessment of the system reliability and safety, and Stochastic Petri Nets for the quantitative assessment of the system reliability and availability) or can be used to carry out both qualitative and quantitative evaluation (e.g., fault-trees and Bayesian networks).

Considering the quantitative evaluation, the attributes of dependability are characterized by metrics [24]. The *reliability* is then defined as:

$$R(t) = Prob\{\tau > t\}$$

that is, the probability that the time to failure (τ) is greater than instant t or, the probability that the system is functioning correctly during time interval $(0, t]$. Considering that $F(t) = 1 - R(t)$ (i.e., unreliability) is a probability distribution function, we can calculate the expectation of the random variable τ as:

$$\int_0^\infty t\, dF(t) = \int_0^\infty R(t)dt$$

that is called *MTTF* (Mean Time To Failure). One of the most used metrics in the reliability analysis domain is the *failure rate* (called also hazard rate) defined as a function of $R(t)$:

$$h(t) = -\frac{1}{R(t)}\frac{dR(t)}{dt}$$

The failure rate represents the probability that a component fails between (t, dt), assuming that it has survived until the instant t. Often, this metric is used by the manufacturers of hardware components to specify their failure rates (e.g., number of failures per year).

For repairable systems, that is systems that can be recovered after failure, *maintainability* and *availability* are more significant properties than reliability. As for the reliability, the maintainability is measured by a probability:

$$M(t) = Prob\{\theta \leq t\}$$

that is, the probability that the time to repair (θ) falls into the interval $(0, t]$. Similarly, we can calculate the expectation of the random variable θ as:

$$\int_0^\infty t \, dM(t),$$

that is called *MTTR* (Mean Time To Repair), and the *repair rate* as:

$$\frac{dM(t)}{dt} \frac{1}{1 - M(t)}$$

The availability is instead defined as the probability that the system is functioning correctly at a given instant:

$$A(t) = Prob\{\text{state} = UP, \text{time} = t\}.$$

In particular, the *steady state* availability can be expressed as function of *MTTF* and *MTTR*:

$$A_\infty = \frac{MTTF}{MTTF + MTTR}$$

The metrics used for safety assessment are similar to the reliability metrics, in particular, a significant metric in safety analysis context is the *safe mission time*, that is the time interval in which the system unreliability is lower than a preassigned threshold.

Finally, it is worthwhile to notice that, often, to qualify the dependability properties of a system the concept of level is used, such as the *Safety-Integrity* levels [9] that correspond to intervals in which the probability of a (dangerous) failure falls.

3. PROPOSALS FROM LITERATURE

This Section is devoted to analyze several significant works in the literature on extending UML to support dependability analysis of software systems. The study has been carried out from a critical perspective, with the purpose of build on such works our proposal of a common dependability profile for real-time embedded systems.

Bondavalli et al. [7, 18] use UML standard extension mechanisms, that is stereotypes and tags for annotating dependability properties of software systems on UML design models. From the annotated models, the authors are able to derive analyzable probabilistic models (Petri Net models) to use in the quantitative system evaluation. In Table 1 we have summarized the stereotypes and related tags used by Bondavalli et al. The proposed approach is compliant with the taxonomy and basic concepts defined in [2].

The non functional properties considered are the reliability and the availability, under both transient and steady state assumptions. Dependability parameters, that can be both input parameters or measures to be derived, are associated to hardware and software components. They allow the analyst to characterize the timing occurrence of faults, the possible error latency for components with an internal state, and the timing of the repair process.

The solution adopted by Bondavalli et al. is not the best choice, since the specification of some tags requires the joint use of two stereotypes and some stereotypes introduce unnecessary redundant information in the system model. For example, a node, that models a hardware component in UML, must be stereotyped as *hardware* and *stateful* to specify the error latency.

Error propagation between components is specified by assigning a probability to the model elements representing either relationships or interactions between such components (such as association between software components, communication path between nodes and exchanged messages).

While dependability parameters can be used to specify component failures due to independent fault occurrences, common failure modes can be specified only for redundant components belonging to complex fault tolerance (FT) structures. Extensions for states and events of state machines representing the behavior of *redundancy manager* components are introduced in order to discriminate normal and failure states and events. Such extensions are used to analyze different failure modes of the FT structures.

Pataricza [22] proposes to extend 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. Although no explicit extension is given, the work emphasizes the importance of including two phenomena in the system model: 1) permanent and transient faults in the resources, and 2) error propagation across the system to estimate which fault may lead to a failure.

Explicit fault injection behavioral models are also proposed to represent faults as special *virtual clients* that request service to components and that have higher priority than the other actual clients. The effect of their request causes a change of state of the server (that is the hardware component affected by the fault occurrence) which moves from normal states (state in which the system is well-functioning) to faulty ones, and to normal states again in case of transient faults. Fault injectors can be used also to model constraints on fault occurrence (e.g., single fault assumption).

Cortellessa and Pompei [10] propose a UML annotation for the reliability analysis of component-based systems. Their work is aimed at including the annotation approach presented in [11], where Bayesian models are derived from UML annotated models to compute the system failure probability, within the frameworks of the SPT and QoS&FT profiles.

Table 2 summarizes the UML extensions introduced in [10]. Although no specific extension is used for the annotation of the reliability requirements and metrics, a set of stereotypes and related tags are instead proposed for the specification of reliability input parameters. Such stereotypes are specialization of stereotypes defined in the Resource Model of the SPT profile.

The most interesting input parameters considered are the *atomic* failure probabilities of software components (*REcomponent*) or (logical/physical) links (*RE connector*), that is the probability that a component, or connector, fails in a single invocation of it. There are no explicit annotations for the failure probability of hardware components, however such trivial extension is proposed in [15] where nodes stereotyped as *REhost* are annotated with *REfailprob* tags.

D'Ambrogio et al. [13] propose a transformation of UML models (Sequence and Deployment diagrams) into fault tree models to predict the reliability of component-based software. Although no UML extension is proposed, several UML model elements whose failure (basic events in fault tree models) can lead to the system failure (top-event in fault tree models) are identified. In particular, the basic events considered are the failure of nodes and communication paths and the failure of call actions, operations and return actions.

DalCin [12] proposes a UML profile for specifying dependability mechanisms, that is hardware/software components to be implemented or integrated in the real-time system to ensure fault tolerance. The proposed profile provides facilities for capturing dependability requirements of such mechanisms to support the evaluation of the effectiveness of the fault tolerance strategy adopted. However, a support to the modeling of the interactions among dependability mechanisms and the system components is missing in the profile.

Jürjens et al. [17, 16] propose a check list, based on UML stereotypes and related tags and constraints to support the analyst in the identification of failure-prone components and assessment of safety/ reliability requirements during the software design.

Stereotype	*Base Class*	*Tag*	*Description*
-	UseCase	measure of interest	$R(t)$,MTTF,$A(t)$,A_∞
hardware (stateless or stateful)	Node	FO	fault occurrence
		PP	percentage of permanent faults
		RD	repair delay
hardware stateful	” ”	EL	error latency
software (stateless or stateful)	UseCase, Class, Object, Component, Package	FO	fault occurrence
software stateful	” ”	EL	error latency
		RD	repair delay
propagation	Extend, Include, Association, Usage (use,call), Dependency, Link, Message Action(send, call), CommunicationPath, Deployment	PP	propagation probability
redundancy manager	Class, Object, Node		redundancy management
variant	Class, Object, Node		replica
adjudicator	Class, Object, Node	CF	common mode failure occurrence
		DC	detection coverage
tester	Class, Object, Node		sub-stereotype of adjudicator
comparator	Class, Object, Node		sub-stereotype of adjudicator
failure	State, Event		SC of redundancy manager
normal	Event		” ”

Table 1: UML availability and reliability annotation used by Bondavalli et al.

Stereotype	*Base Class*	*Tag*	*Description*
REcomponent	Classifier, ClassifierRole, Component, Instance	REcompfailprob	atomic failure probability
		REbp	number of invocation
REconnector	Message, Stimulus, AssociationRole	REconfailprob	atomic failure probability
		REnummsg	number of invocation
REuser	Classifier, Classifier, ClassifierRole, Interaction, Instance	REaccessprob	probability of system access
		REserviceprob	probability of a service request after system access
REservice	Classifier	REprob	probability of service request
REhost	Node, Classifier, ClassifierRole	REindexHost	list of hosts to which the host is physically connected

Table 2: UML reliability annotation used by Cortellessa and Pompei

The extensions introduced in [17, 16] are summarized in Table 3: most of them are used to specify requirements on communication, such as the stereotype *guarantees* whose tag *goal* is of complex type and can express either a maximum duration allowed for data transmission, or a probability that eventually a data is delivered or a maximum probability of message loss. Several stereotypes can be applied to specify guarantees at subsystem level. An interesting aspect considered in [16] is the possibility of specifying both requirements (*guarantees* stereotype) and failure assumptions (*risk, crash/performance, value* stereotypes) according to the failure domain, that is timing failure or content failure. Some extensions are also proposed for the specification of fault tolerant mechanisms, such as the types of voting algorithms.

Addouche et al. [1] propose a profile for dependability analysis of automated system of production. Actually, a pair of stereotypes are defined to include probabilistic aspects of functioning and malfunctioning. The static model of the system is enriched with new stereotyped classes that are associated with each class representing a resource according to the SPT profile. The *indicator* classes are characterized by attributes related to the dynamic aspects of the resource classes associated with, and their values represent the degraded or failure state of the resource classes. The *cause* classes are characterized by attributes representing logical expressions of failure occurrence in the resource classes associated with.

This approach allows the analyst to specify non trivial non functional properties; nevertheless it is not the best choice since new classes need to be defined and introduced in the system model, beside the classes representing the actual system components, for dependability analysis purposes.

Bernardi et al. [3] propose a set of UML Class Diagrams structured in packages, to support the collection of dependability and real-time requirements of automated embedded systems with the use of COTS FT mechanisms. Although no extension is proposed, several class attributes are introduced that support the specification of quantitative dependability properties. The most interesting class diagrams are the ones that define the causal relationships among the impairments of dependability, according to [2], and the FT strategies.

In [4] we propose a method to assess Quality of Service (QoS) of FT distributed systems via derivation of performance models from SPT annotated UML behavioral and deployment diagrams. Some UML extensions have been explicitly introduced since the SPT profile does not support the specification of fault occurrences and latencies.

Stereotype	*Base Class*	*Tag*	*Description*
guarantee	Link, Node	goal (type: immediate($t), eventual($p), correct($q))	requirements for communication data
safe/reliable links	Subsystem		dependency safety/reliability to be matched by links
safe/reliable dependency	Subsystem		"call","send" have to respect data safety/reliability
safe/reliable behavior	Subsystem		safe/reliable behavior
critical	Class, Object	level	safety/reliability level
containment	Subsystem		no unsafe/unreliable interference between components on different safety/reliability levels
risk	Link, Node	failure (type: delay($t), loss($p), corruption($q))	timing and content failures
crash/ performance	" "	failure (type: delay($t), loss($p))	"risk" to address timing failures
value	" "	failure (type: corruption($q))	"risk" to address content failures
redundancy	Dependency, Component	model (type: none, majority, fastest)	type of redundancy to be implemented
error handling	Subsystem	error object	handles errors

Table 3: UML safety and reliability annotation used by Jürjens et al.

Although the work is focused on QoS assessment, it can provide some tips for the quantitative characterization of the "goodness" of a FT strategy. Indeed, the objective of the analysis is to evaluate the QoS of the FT strategy implemented in the system under late-timing failure assumption. The QoS is defined as a function of two non functional properties: one is strictly related to the FT effectiveness (i.e., the time to detect a failure) and the other is related to the cost of the FT (i.e., communication overhead).

	(1)	**(2)**	**(3)**	**(4)**	**(5)**	**(6)**	**(7)**
Bondavalli et al.	R,A	✓	✓	✓	✓	✓	✓
Cortellessa-Pompei	R	✓					
Grassi et al.	R	✓					
D'Ambrogio et al.	R	✓			✓		
Pataricza	R,A		✓	✓		✓	
DalCin	R,A						✓
Jürjens et al.	S,R	✓			✓		✓
Addouche et al.	R,M	✓	✓		✓	✓	
Bernardi et al.	R,A	✓	✓	✓	✓		✓
Bernardi-Merseguer	QoS				✓	✓	✓

Legend **(1)**: A=availability, R=reliability, S=safety, M=maintainability, QoS=quality of service.

Table 4: Dependability aspects dealt by the mentioned works

Table 4 summarizes the dependability aspects considered in the mentioned works according to the following check-list that will guide us toward the definition of a UML profile for dependability analysis of RTES:

(1) Non functional requirements to be assessed and measures to be computed during the dependability analysis, with particular focus on the attributes of dependability [2]

(2) Dependability input parameters that characterize the fault-failure processes of system components (both hardware and software) and/or the repair processes

(3) Faults behavior with respect to their timing persistence

(4) Error propagation process among system components

(5) System failure modes and system service degradation

(6) Behavior of system components affected by impairments

(7) FT mechanisms, representation of hardware/ software redundancies, effectiveness of FT and its cost.

4. THE DEPENDABILITY PROFILE

In this Section we propose a set of UML extensions to support the dependability analysis of real-time and embedded systems. Since our goal is to be compliant with the MARTE RFP, our main guideline is the proposal [14] in which a general framework to support the definition of *complex non functional properties (NFP)* is provided.

Then, the approach followed here is to apply such framework considering the check-list drawn up at the end of Section 3 to include the necessary dependability concepts. Each choice made in the definition of the dependability profile refers to a specific item **(n)** of the check-list, that will be indicated, near the proposed UML extension and between brackets, as **guideline-n**.

Figure 1 shows the package overview representing the dependency of the Dependability Analysis (DA) profile with the non-functional properties (NFP) profile proposed in [14], where the concept of *complex NFP* has been introduced. Indeed, the new stereotypes defined in the DA profile are characterized by *complex NFP* attributes. Following the approach presented in [14], the DA profile is structured in two sub-packages:

- The *Dependability Analysis UML Extensions* package, includes the new stereotypes that will be used to annotate the system model built by the user, and
- The *NFP Library For Dependability Analysis* package, includes the concepts of dependability represented as *complex NFP*.

4.1 The NFP library

Impairments of dependability are defined as complex NFP (Figure 2) and they are characterized by a set of *basic NFP* attributes that can represent either quantitative properties or qualitative properties.

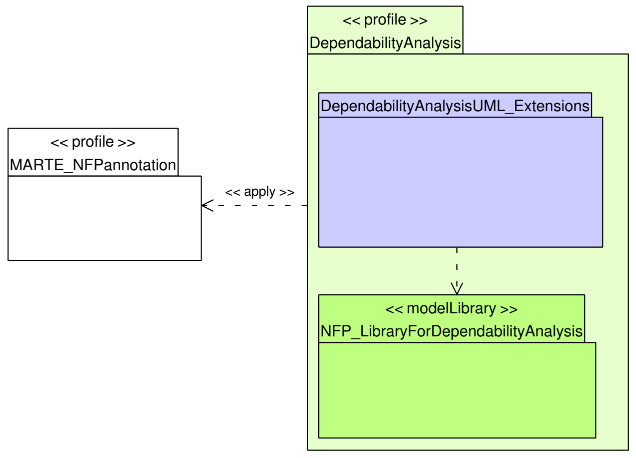
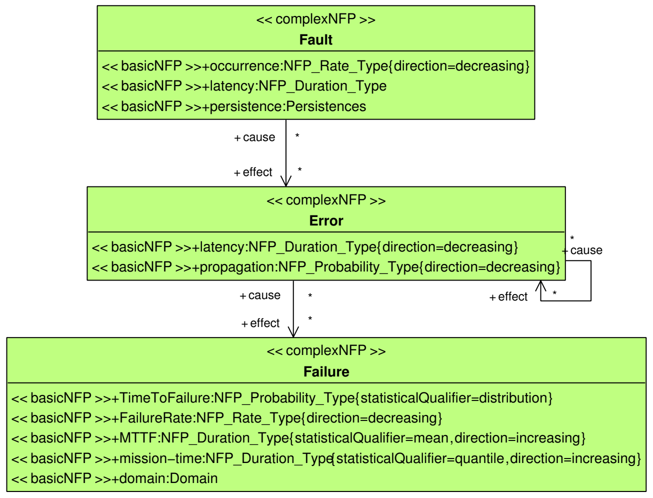


Figure 1: Overview of the DA profile

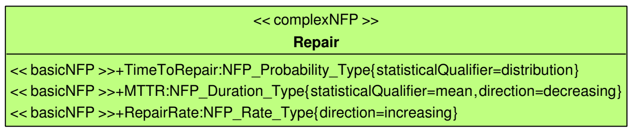
In the DA profile, quantitative *basic NFP* can represent: *rates*, such as the fault occurrence and the failure rate (**guideline-2**), *durations*, such fault and error latencies (**guideline-2**), or *probabilities*, such as the probability of error propagation between two system components (**guideline-4**) and the time to failure (**guideline-1**). Qualitative *basic NFP* are instead enumeration types, such as the fault persistence (**guideline-3**) and the failure domain (**guideline-5**).



<< complexNFP >>
Fault
<< basicNFP >>+occurrence:NFP_Rate_Type{direction=decreasing}
<< basicNFP >>+latency:NFP_Duration_Type
<< basicNFP >>+persistence:Persistences
+cause
*
+effect
*
<< complexNFP >>
Error
<< basicNFP >>+latency:NFP_Duration_Type{direction=decreasing}
<< basicNFP >>+propagation:NFP_Probability_Type{direction=decreasing}
*
+cause
+effect
*
+cause
*
+effect
*
<< complexNFP >>
Failure
<< basicNFP >>+TimeToFailure:NFP_Probability_Type{statisticalQualifier=distribution}
<< basicNFP >>+FailureRate:NFP_Rate_Type{direction=decreasing}
<< basicNFP >>+MTTF:NFP_Duration_Type{statisticalQualifier=mean,direction=increasing}
<< basicNFP >>+mission−time:NFP_Duration_Type{statisticalQualifier=quantile,direction=increasing}
<< basicNFP >>+domain:Domain

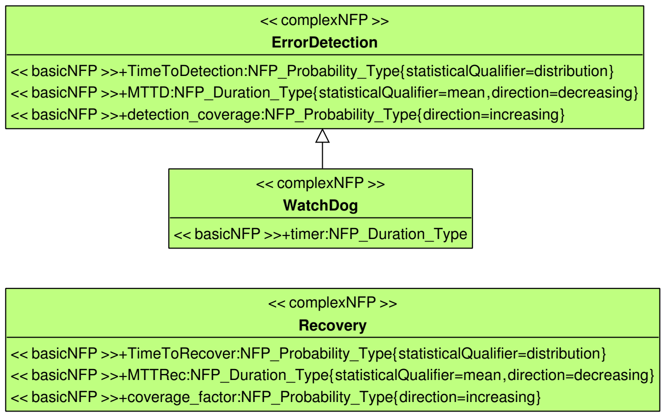
Figure 2: Impairments

Basic NFP attributes may have associated two kinds of properties: the statistical qualifier property specifies the type of statistical function represented by the attribute, while the direction property specifies the order of relation for QoS purposes. For example, the time to failure is a probability distribution function, characterizing the system unreliability, and the MTTF is the mean of such distribution (**guideline-1**). Moreover, if we compare two system scenarios with different MTTF, the better scenario is the one with the greater MTTF (increasing order of relation). Faults, errors and failures are related with the cause-effect relationships, and errors may propagate between system components before causing a system failure.



<< complexNFP >>
Repair
<< basicNFP >>+TimeToRepair:NFP_Probability_Type{statisticalQualifier=distribution}
<< basicNFP >>+MTTR:NFP_Duration_Type{statisticalQualifier=mean,direction=decreasing}
<< basicNFP >>+RepairRate:NFP_Rate_Type{direction=increasing}

Figure 3: Repair



<< complexNFP >>
ErrorDetection
<< basicNFP >>+TimeToDetection:NFP_Probability_Type{statisticalQualifier=distribution}
<< basicNFP >>+MTTD:NFP_Duration_Type{statisticalQualifier=mean,direction=decreasing}
<< basicNFP >>+detection_coverage:NFP_Probability_Type{direction=increasing}
<< complexNFP >>
WatchDog
<< basicNFP >>+timer:NFP_Duration_Type
<< complexNFP >>
Recovery
<< basicNFP >>+TimeToRecover:NFP_Probability_Type{statisticalQualifier=distribution}
<< basicNFP >>+MTTRec:NFP_Duration_Type{statisticalQualifier=mean,direction=decreasing}
<< basicNFP >>+coverage_factor:NFP_Probability_Type{direction=increasing}

Figure 4: Error detection and recovery (FT)

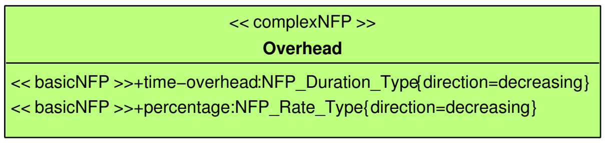
In order to characterize the repair process of a system component after failure, a *complex NFP* has been introduced (Figure 3). The quantitative characterization of the repair process is given in terms of a probability distribution - time to repair, its mean - MTTR (**guideline-1**) and the repair rate (**guideline-2**). The specification of the repair process in the dependability model is important from the analysis point of view, when the goal is, for example, the assessment of the system availability.

The QoS&FT profile includes a sub-profile devoted to support the design of FT software architectures, and hence the specification of the functional properties of the mechanisms implementing the FT, in which most of the FT basic concepts have been introduced (such as the software redundancy and the replication styles).

Here, we propose further extensions (*complex NFP*) that support the analyst in the quantitative assessment of FT techniques implemented in RTES. Figure 4 shows the representation of the two main steps of FT [2], that is error detection and system recovery. Error detection encompasses those techniques and mechanisms used to identify the presence of an error in the system.

The *complex NFP Error Detection* is characterized by attributes that quantify the time of error detection once an error has occurred in the system. In particular, they represent the probability distribution, its mean and the detection coverage (**guideline-7**). Typical mechanisms used to detect errors during the normal service delivery are watchdog timers that raise error exceptions if they do not receive from the controlled application signal of life within a prefixed time interval. The *complex NFP Watchdog* has been added as an example of how to specialize the error detection concept with further mechanism-specific NFP, such as the timer duration of the watchdog (**guideline-7**).

System recovery includes those techniques that bring the system from a faulty or erroneous state into a state without errors or latent faults. A *complex NFP* has also been defined for this FT step in order to quantify the time required to recover the system from error



<< complexNFP >>
Overhead
<< basicNFP >>+time–overhead:NFP_Duration_Type{direction=decreasing}
<< basicNFP >>+percentage:NFP_Rate_Type{direction=decreasing}

Figure 5: Overhead (FT)

to avoid a system failure and the coverage factor (**guideline-7**). As for error detection, also the recovery stereotype can be specialized to characterize NFP of specific mechanisms implemented in this step, such as voter mechanisms used to mask faults.

Finally, the last *complex NFP* proposed is the overhead introduced in the system by the implemented FT strategy (Figure 5) that allows the analyst to specify the cost of the FT. We have defined the overhead in terms of time spent by the system in carrying out FT actions and of percentage (**guideline-7**). The time required to execute a voting algorithm on outcomes produced by independent computation replicas is an example of time overhead. The augmented traffic in the network due to the execution of a self-checking algorithm, consisting in exchanging control messages among components of a distributed system [4], can be expressed as a percentage overhead.

4.2 The dependability stereotypes

The *complex NFP* defined so far are not used directly in the annotation of the system model, they represent instead complex types of some attributes of the stereotypes defined in the dependability analysis package. This package contains the extensions that will support the annotation of dependability requirements and properties in the system UML models.

The attributes of dependability described in Section 2 are defined as stereotypes (Figure 6), then suggesting the type of dependability analysis to be carried out, e.g., reliability, availability and safety analysis (**guideline-1**). The three stereotypes have the same base classes; the base classes have been selected from the UML2 meta-model considering the UML model elements extended with non-functional annotations in the different proposals mentioned in Section 3. So that, hardware and software components, communication path between nodes and interactions between software components, as well as the whole subsystem included in package and use cases representing the top-level services offered by the system can have assigned the same stereotype, e.g., reliability, which allows the analysts to characterize their reliability requirements and properties.

The reliability stereotype is characterized by attributes of *complex NFP* type, such as fault, error and failure, and by an enumeration type attribute (*basic NFP*) that can be used to specify the critical level of system components (**guideline-1**). Similarly, the safety stereotype has a *complex NFP* type attribute, to be used to specify the hazards that may bring the system to catastrophic failures (**guideline-2**), and an enumeration type attribute (*basic NFP*) for the assignment of safety levels to system components (**guideline-1**). We have considered the availability property as a particular case of reliability: it is meaningful only for repairable components and systems, and to compute it a quantitative characterization of the repair process should be specified. To specify the availability two *basic NFP* attributes can be used: steady state and instantaneous (**guideline-1**). The former is a derived attribute, since its value can be computed from the values of attributes defined in the *complex NFP* failure (MTTF) and repair (MTTR), the latter represents instead a probability value.

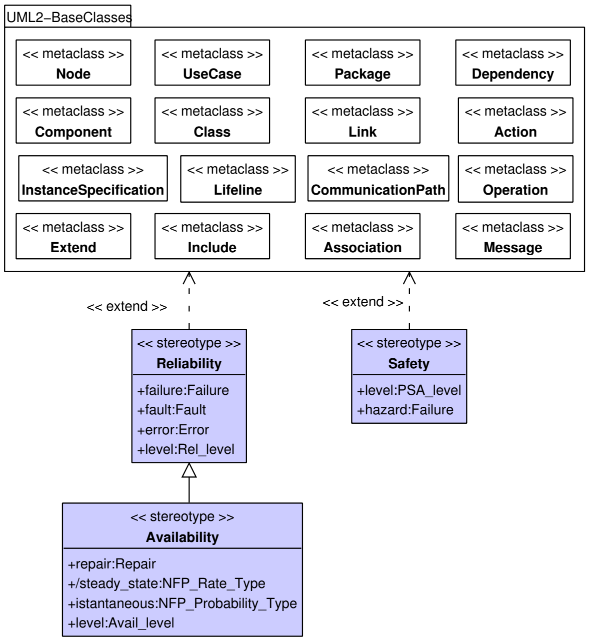


Figure 6: Dependability stereotypes

As alternative to the steady state attribute, the availability level allows the analyst to specify the so called "nines of availability", used by the hardware manufacturers to characterize their products (**guideline-1**). For example, a high availability level may be associated to 3 nines (i.e., $99,9\%$ of availability) that corresponds to $8,5$ hours per year of downtime. The availability stereotype inherits from reliability one also the base classes, so it can be applied to the same model elements as the reliability stereotype.

The integrity and maintainability stereotypes, although not explicitly shown in Figure 6, are characterized by similar complex NFP attributes. In particular, the integrity stereotype has an attribute of type *Error* that supports the specification of communication errors (**guideline-4**) and the maintainability stereotype has an attribute of type *Repair* to be used for the characterization of the repair process (**guideline-2**).

Finally, a FT stereotype has been defined to support the assessment of the FT solution adopted for a given real-time embedded system (Figure 7). The attributes of FT are used to characterize, from a quantitative point of view, the error detection and recovery techniques implemented in the system as well as the overhead paid by using such FT solutions (**guideline-7**). The attribute of type integer can be used instead to specify the maximum number of faults supported.

4.3 Discussion

Most of the shortcomings pointed out in some of the reviewed literature have been addressed by the proposed dependability profile. For example, the specification of the error latency on a faulty node is simplified in the profile. Instead of using the pair of stereotypes *hardware* and *stateful*, as suggested in [7, 18], to set a value to the tag *EL*, the node is stereotyped with a single stereotype, e.g., *reliability*, and a value is set to the *latency* attribute of the *complex NFP error*.

The DA profile supports the specification of the two phenomena emphasized in [22], that is the fault persistence (**guideline-**

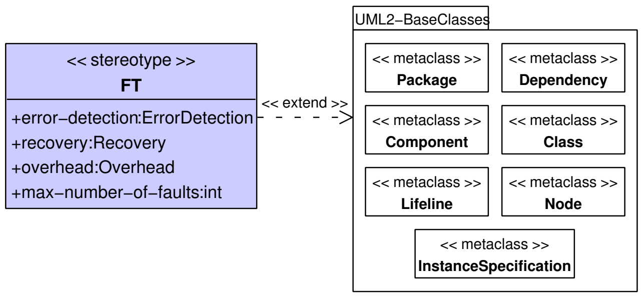


Figure 7: FT stereotype

3) and the probability of error propagation between components (**guideline-4**), through the definition of proper attributes (*persistence* and *propagation*) of *complex NFP* (*fault* and *error*, respectively).

In general, each dependability property, annotated by the analyst on the UML system model, will correspond to a value assigned to a proper *basic NFP* attribute. The value of a *basic NFP* attribute includes a textual specification of the property and the *source* which indicates whether the property value is a requirement to be assessed - *req* - or a measure to be predicted with the dependability analysis - *pred* (**guideline-1**), or an input parameter to be assumed - *assm* (**guideline-2**). The specification of the *source* addresses a shortcoming of the work [10], where no specific extensions are given to specify requirements and measures.

Failure occurrence in nodes, communication paths, actions and operations, as indicated in [13], can be specified by using the same *basic NFP FailureRate* and the same stereotype, which suggests the type of analysis to be carried out. This choice avoids the introduction of a different stereotype, and related tag, for each type of system component affected by a failure, as proposed in [10], to specify the same dependability property.

It is worthwhile to note that no new classes are added in the UML models of the system for the specification of dependability properties, as instead proposed in [1]. Indeed, the extensions defined in the profile support the specification of dependability properties through the introduction of annotations attached to the model elements.

As suggested by the works [17, 16], failures are discriminated with respect to the domain - timing and content failures - through the usage of the attribute *domain* of the *complex NFP failure*. However, the **guideline-5** has been partially supported by the dependability profile and further investigations are needed to support the specification of the system service degradation.

Finally, the **guideline-6** has not been specifically addressed in the current proposal and will be part of our future work. Provide UML extensions for the specification of the behavior of the system components affected by impairments will possibly solve the lack of support to the modeling of interactions between system components and fault tolerance mechanisms; a shortcoming we pointed out in the work [12].

5. A GAS TURBINE CONTROL SYSTEM

The case study presented in [6] is the digital embedded control system supplied for the gas turbine in a co-generative plant in Casaccia, Italy. This system will be useful to illustrate how to apply some of the most relevant NFPs proposed for dependability analysis.

The work in [6] proposes different dependability models (fault-trees, Bayesian networks and stochastic Petri nets) for the safety and availability assessment of the control and protection functions of the gas turbine control system. In this Section we recall a small part of the system, specified by the UML models shown in Figure 8, in order to demonstrate the applicability of the DA profile to a real case study. The UML models of Figure 8 don't allow us to get the dependability models proposed in [6], since not all the system components together with their dependability parameters have been explicitly modeled here. A fault tree model could be obtained following the approach [13]; however, we think that this example will be enough for our illustrative purposes.

The system architecture, deployment diagram in Figure 8, consists of a main controller (MC) and a backup unit (BU), which interact with a speed probe and two thermocouple sensors. The actuators provide input signals (overspeed, overtemp) to the MC and the BU to protect the turbine. Each device fails under different assumptions that have been specified in the model as number of failures per hour (*FailureRate* basic NFP).

Watchdog software components are associated to the MC and BU for shutdown requests, see the sequence diagram in Figure 8: the timer duration of the watchdog associated to the main controller is an input parameter ($duration) whose value has to be set during the analysis. The performance annotations for the duration of actions in the sequence diagram could complement the dependability system view.

For the system safety assessment some measures need to be predicted via dependability analysis, such as the mean time to failure (MTTF) and the mission time. Moreover, as specified by the *Time To Failure* basic NFP, the system unreliability should be lower than 10^{-3}. In [6], the safety assessment is followed by the availability analysis to evaluate the availability of the gas turbine system under different repair rate assumptions.

In Figure 8 some annotations have been included as examples of measures for this type of analysis: the repair rate is specified as an input parameter ($rr), the steady state availability represents a measure to be computed in the analysis, then it is specified as an output parameter ($avail), and the availability level of the system is required to be 4 nines, i.e., $99,99\%$.

Finally, assume that the MC can be found in three conditions (*working, degraded, failed*). When the MC is *degraded* induces an anomalous behavior in the BU, quantified as a failure with probability 0.9. The proposed profile does not support the modelling of non trivial relations among NFP of different model elements. Apart from this problem, the profile has been powerful enough to model the NFPs of the case study.

6. CONCLUSION

In this paper we have proposed a UML Dependability Analysis (DA) profile to support the assessment of the dependability of real-time embedded systems. The profile conforms to the upcoming MARTE RFP, issued by the OMG and, in particular, the basic concepts of dependability defined in the DA profile are expressed in terms of complex non-functional properties (NFP).

The DA profile specifically addresses the *quantitative* evaluation of dependability and the notions introduced in the profile should complement the ones defined in the QoS&FT sub-profile, which supports instead the specification of FT software architectures. Although the DA profile has been conceived to be compliant with the MARTE RFP, it is general enough to support the dependability assessment of any software/hardware system.

We have shown the applicability of the DA profile by enriching the UML design model of a gas turbine control system with the in-

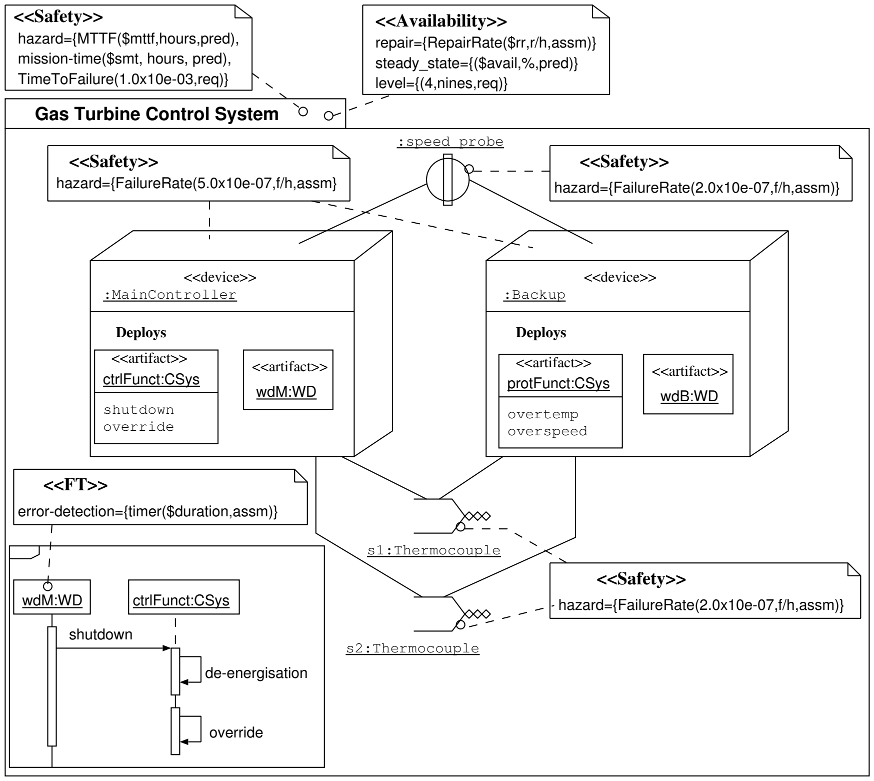


Figure 8: System architecture

troduced NFP, for safety and availability assessment purposes. We have not discussed, in this work, the issues related to the derivation of dependability analysis models (e.g., fault-trees, Baye-sian networks, Stochastic Petri Nets) from the UML-DA annotated models. However, we feel confident that the derivation approaches proposed in the literature can be applied on UML-DA annotated models since the DA profile has been built on the best practices on extending UML with modeling capabilities for dependability analysis purposes.

As future work, we plan to address the guideline 6 of the checklist, drawn up at the end of Section 3, that has not been considered in the current proposal. Another issue that will be subject of our future investigation is how to specify relations among NFP of different UML model elements, such as the conditional failure probability of a system component that depends on the state probabilities of another component. The work [14] does not deal specifically with them, however it suggests the use of constraints, e.g., written in a machine readable language such as OCL, as solution for the specification of this kind of non trivial relations.

7. REFERENCES

[1] N. Addouche, C. Antoine, and J. Montmain. UML models for dependability analysis of real-time systems. In *In Proc. International Conference on Systems, Man and Cybernetics*, volume 6, pages 5209 – 5214. IEEE CS., Oct. 2004.

[2] A. Avizienis, J.C. Laprie, B. Randell, and C. Landwehr. Basic Concepts and Taxonomy of Dependable and Secure Computing. *IEEE Transactions on dependable and secure computing*, 1(1):11–33, January-March 2004.

[3] S. Bernardi, S. Donatelli, and G. Dondossola. A class diagram framework for collecting dependability requirements in automation systems. In *In Proc. of the* 1^{st} *International Symposium on Leveraging Applications of Formal Methods (ISOLA'04)*, Paphos, Cyprus, 2004.

[4] S. Bernardi and J. Merseguer. QoS Assessment via Stochastic Analysis. *IEEE Internet Computing*, pages 32–42, May-June 2006.

[5] S. Bernardi and D. Petriu. Comparing two UML Profiles for non-functional requirement annotations: the SPT and QoS Profiles. SVERTS - Satellite Events at the UML Conference, Lisbon (Portugal) - 2004.

[6] A. Bobbio, E. Ciancamerla, G. Franceschinis, R. Gaeta, M. Minichino, and L. Portinale. Sequential application of heterogeneous models for the safety analysis of a control system: a case study. *Reliability Enginering and System Safety*, 81:269–280, 2003.

[7] A. Bondavalli, I. Majzik, and I. Mura. Automatic Dependability Analysis for supporting design decisions in UML. In A. Williams, editor, *in Proc. of Fourth IEEE International High Assurance System Engineering Symposium (HASE'99)*. IEEE Computer Society Press, 1999.

[8] International Electrotechnical Commission. IEC-60300-3-1 standard: Dependability management.

[9] International Electrotechnical Commission. IEC-61508 standard: Functional Safety of Electrical/ Electronic/ Programmable Electronic safety related problems.

[10] V. Cortellessa and A. Pompei. Towards a UML Profile for QoS: a contribution in the reliability domain. In *Proceedings of the Fourth International Workshop on Software and Performance (WOSP'04)*, pages 197–206, January 2004.

[11] V. Cortellessa, H. Singh, and B. Cukic. Early reliability assessment of UML based software models. In *In*

Proceedings of Third International Workshop on Software and Performance, pages 302–309, Rome, Italy, July 2002.

[12] M. Dal Cin. Extending UML towards a Useful OO-Language for Modeling Dependability Features. University of Erlangen-Nürnberg, Informatik 3, Germany, 2003.

[13] A. D'Ambrogio, G. Iazeolla, and R. Mirandola. A method for the prediction of software reliability. In *Proc. of the 6-th IASTED Software Engineering and Applications Conference (SEA2002)*, Cambridge, MA, USA, November 2002.

[14] H. Espinoza, H. Dubois, S. Gérard, J. Medina, D.C. Petriu, and M. Woodside. Annotating UML models with non-functional properties for quantitative analysis. In *Proceedings of Models*, volume 3844 of *LNCS*. Springer-Verlag, 2005.

[15] V. Grassi, R. Mirandola, and A. Sabetta. From Design to Analysis Models: a Kernel Language for Performance and Reliability Analysis of Component-Based Systems. In *Proceedings of the Fifth International Workshop on Software and Performance (WOSP'05)*, pages 25–36, July 2005.

[16] J. Jürjens and S. Wagner. Component-based Development of Dependable Systems with UML. In Atkinson et al., editor, *Component-Based Software Development*, volume 3778 of *LNCS*, pages 320–344. Springer-Verlag, 2005.

[17] Jan Jürjens. Developing safety-critical systems with UML. In *UML 2003, San Francisco*, volume 2863 of *LNCS*, pages 360–372. Springer-Verlag, October 2003.

[18] I. Majzik, A. Pataricza, and A. Bondavalli. Stochastic Dependability Analysis of System Architecture Based on UML Models. In R. De Lemos, C. Gacek, and A. Romanovsky, editors, *Architecting Dependable Systems, LNCS 2677*, Lecture Notes in Computer Science, pages 219–244. Springer-Verlag, Berlin, Heidelberg, New York, 2003.

[19] Object Management Group. *UML Profile for Schedulabibity, Performance and Time Specification*, January 2005. Version 1.1, formal/05-01-02.

[20] Object Management Group. *Unified Modeling Language: Superstructure*, July 2005. Version 2.0, formal/05-07-04.

[21] Object Management Group. *UML Profile for Modeling Quality of Service and Fault Tolerant Characteristics and Mechanisms*, May 2006. Version 1.0, formal/06-05-02.

[22] A. Pataricza. From the General Resource Model to a General Fault Modelling Paradigm ? Workshop on Critical Systems, held within UML'2000, 2000.

[23] D. Powell, E. Martins, J. Arlat, and Y. Crouzet. Estimators for Fault Tolerance Coverage Evaluation. *Transaction on Computers*, 44(2), February 1995.

[24] R.A. Sahner, K.S. Trivedi, and A. Puliafito. *Performance and Reliability Analysis of Computer Systems: An Example-Based Approach Using the SHARPE Software Package*. Kluwer Academic Publishers, 1996.