Annotating UML Models with Non-Functional Properties for Quantitative Analysis

Huáscar Espinoza[a], Hubert Dubois[a], Sébastien Gérard[a], Julio Medina[b], Dorina C. Petriu[c], Murray Woodside[c]

[a] CEA Saclay, DRT/LIST/DTSI/SOL/L-LSP,
F-91191, Gif sur Yvette Cedex, France
{huascar.espinoza, hubert.dubois, sebastien.gerard}@cea.fr

[b] Universidad de Cantabria, Departamento de Electrónica y Computadores,
Av. Los Castros s/n, 39005 Santander, Spain
medinajl@unican.es

[c] Carleton University, Department of Systems and Computer Engineering
1125 Colonel By Drive, Ottawa, ON, Canada, K1S 5B6
{petriu, cmw}@sce.carleton.ca

Abstract. This work is motivated by the recent Request For Proposals issued by OMG for a new UML Profile named "Modeling and Analysis of Real-Time and Embedded systems". The paper describes first some domain concepts for annotating Non-Functional Properties (NFPs), whose focus is on supporting temporal verification of UML-based models. Particular emphasis is given to schedulability and performance analysis for real-time systems. We discuss next some general requirements for NFP annotations and evaluate how the UML profiles for "Schedulability, Performance, and Time Specification" and for "Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms", address these requirements. Last but not least, the paper proposes a preliminary framework for describing NFPs by considering the major requirements previously stated and by analyzing some UML mechanisms to attach NFPs to model elements.

1 Introduction

The change of focus from code to models promoted by OMG's Model Driven Architecture (MDA) raises the need to integrate the analysis of non-functional requirements of UML models (such as performance, schedulability, reliability, scalability, etc.) in the development process of Real-Time and Embedded Systems (RTES). Different kinds of analysis techniques require different annotations in the UML models to express quantitative and qualitative non-functional requirements and properties.

The focus of this paper is on annotations for quantitative analysis techniques used for the verification and validation of temporal characteristics of RTES. Such annotations are required to bridge the gap between the domains of software development and analysis, because they should be usable by software designers but, at the same time, they also must support the analysis model concepts. This paper discusses the problem

of adding non-functional properties to an UML model, but does not address other related problems, such as transforming an annotated UML model into an analysis one, evaluating the analysis model, or reporting the results back to UML models [8].

The solution proposed by OMG to the problem of extending the power of expression of UML for different application domains is to define standard UML profiles. Two examples of profiles able to add annotations for non-functional characteristics are the "UML Profile for Schedulability, Performance, and Time Specification" (SPT) [11] and the "UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms" (QoS&FT) [13]. In order to upgrade SPT to UML 2.0 and extend its scope with RTES modeling capabilities, a Request For Proposals (RFP) was issued for a new UML Profile named MARTE ("Modeling and Analysis of Real-Time and Embedded systems") [12]. The goals of this paper are to give a first reflection on the analysis concerns for the upcoming UML MARTE profile and to promote its standardization at the OMG.

Some of this underlying work is supported by the Accord$|_{UML}$ project [3, 7] and by the MAST (Modeling and Analysis Suite for real-Time applications) project [10] to connect UML models of real-time embedded systems with schedulability analysis tools. A first experimental approach was defined in [15], where the authors have presented a schedulability analysis model which is semi-automatically derived from a conception model, and is then analyzed [9]. MAST defines and builds UML conceptual models, which align quite well with the SPT profile concepts, for the consideration of timing properties in object-oriented distributed systems.

A rich body of work on performance analysis from UML models has been surveyed in [1]. Examples of UML model transformations to different performance modeling formalisms are: from UML to Layered Queueing Networks in [8, 16], to Stochastic Petri Nets in [2], and to multiple performance models in [19].

Other UML profiles for different quantitative analyses have already been proposed in the literature, such as a reliability profile in [4], a profile with formal semantics dedicated to real-time modeling named OMEGA in [6], and a profile for real-time constraints with OCL in [5].

The paper is organized as follows. Section 2 describes the domain model for non-functional property annotations for quantitative analyses. Section 3 presents a list of requirements for NFP annotations. Section 4 compares briefly the advantages and disadvantages of the SPT and QoS&FT Profiles. Section 5 presents our proposal for a framework for NFP annotations for MARTE, which realizes the domain model introduced in Section 2. Finally, the conclusions are presented in Section 6.

2 Domain Model for Non-Functional Properties Annotations

2.1 Domain Model

The *model* of a computing *system* describes its architecture and behavior by means of *model elements* (e.g., resources, resources services, behavior features, logical operations, configurations modes, modeling views), and the externally visible properties of those model elements. When we refer to *properties*, this includes the

functional and also the *non-functional properties*. Functional properties describe what a system model does, and non-functional properties how it does it.

In the context of model-driven development approaches for real-time and embedded systems, modeling of Non-Functional Properties (*NFPs*) is essential for the quantitative analysis of the system (see Figure 1). NFPs provides information about different characteristics, as for example throughput, delays, overheads, scheduling policies, correctness.

Quantitative analysis techniques are used to verify early NFPs of interest (e.g., response times, utilization, queue sizes) based on other available NFPs (e.g., worst case execution times -WCET-, deadlines). The analysis techniques considered in this paper belong to the two following analysis domains: *Schedulability* and *Performance.* Further work will also cover *WCET Analysis*. Schedulability analysis uses mathematical means (e.g., RMA-based techniques) to predict whether a set of software tasks meets its timing constraints and to verify its temporal correctness. Performance analysis uses statistical techniques (e.g., queuing theory, Petri Nets, etc.) to determine whether a system will meet its performance requirements (such as response time or throughput).

Due to the abstraction involved in the construction of a model, only some NFPs are relevant to a certain Quantitative Analysis. In other words, a given Quantitative Analysis uses a set of NFPs which establish the ontology of the analysis domain.

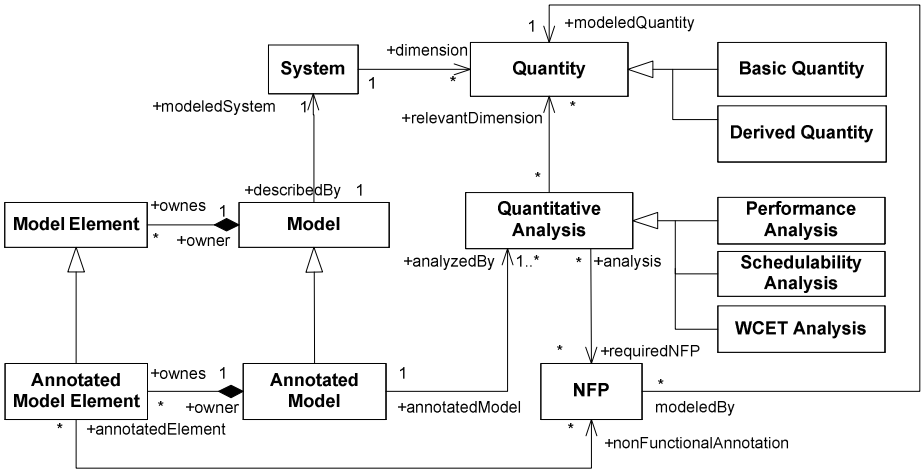

Figure 1. Domain model for Non-Functional Property annotations

According to measurement theory, physical *Systems* (see Figure 1) are characterized along different dimensions that correspond to a set of measurement *Quantities*, which can be *Basic* or *Derived*. The most used Basic Quantities are length, mass, time, current, temperature and luminous intensity. The units of measure for the basic quantities are organized in systems of measures, such as the universally accepted *Système International* (SI) or International System of Units. Values are expressed in the same unit and can be compared. Derived Quantities (e.g., area, volume, force, frequency, etc.) are obtained from the Basic Quantities by known formulas.

A *Model* of a System (which is considered here to be expressed in UML) can be extended by standard UML mechanisms with additional semantic expressing concepts from a given analysis domain. An *Annotated Model* contains *Annotated Model Elements*, which are UML model elements extended by standard UML mechanisms. For example, some typical performance-related *Annotated Model Elements* are: *Step* (a unit of execution as defined in the SPT profile), *Scenario* (a sequence of Steps), *Resource* (as defined in the General Resource Model of SPT), *Service* (an operation offered by a *Resource* or by a component of some kind, which may be further defined by a *Scenario*).

An *Annotated Model Element* has certain non-functional characteristics represented by NFPs. The annotations are specified by the designer in the UML model and attached to different model elements. Examples are: the total delay of a *Step* when executed (including queueing delays), the utilization of a *Resource*, the response time and throughput of a *Service*, etc.

2.2 Quantitative and Qualitative NFPs

In general, a NFP can be either *qualitative* or *quantitative*, as shown in Figure 2. Most of the NFPs used for quantitative analysis (such as performance or schedulability) are quantitative, but some properties may be qualitative.

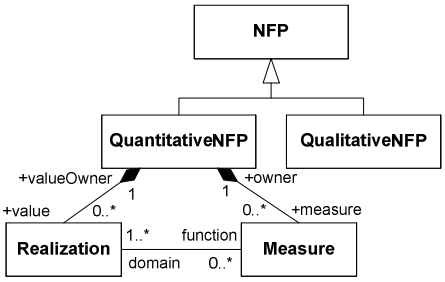

Figure 2. Domain model for Quantitative and Qualitative NFPs

A ***Quantitative NFP*** is measurable, countable, or comparable, and can be represented by an amount which is a numerical value. When the system is simulated or executed, a given *Quantitative NFP* may be characterized by a set of *Realizations* and *Measures* (see Figure 2). *Realizations* (also called *Sample Functions*) represent a set of values that occur for the *Quantitative NFP* under consideration at run-time (for instance, measurements collected from a real system or a simulation experiment). A *Quantitative NFP* may be realized once or repeated times over an extended run. In a cyclic deterministic system, in which each cycle has the same values, a single Realization is sufficient to characterize completely the *Quantitative NFP*. In performance analysis with random traffic, a long run producing long sequences of values may be necessary in order to obtain accurate evaluation results.

A *Measure* is a (statistical) function (e.g, mean, max, min, median, variance, standard deviation, histogram, etc.) characterizing the set of Realizations. Measures may be computed either directly by applying the desired function to the set of Realizations

values, or by using theoretical functions of the probability distribution given for the respective *Quantitative NFP*.

In any case, even Realization sets are not annotated directly on the UML model (too much information!) They are represented instead in an abstract way through the corresponding Measures, which should be annotated on the UML model.

On the other hand, a ***Qualitative NFP*** refers to inherent or distinctive characteristics that are not easy to measure directly. In general, a *Qualitative NFP* is denoted by a label (e.g., "bronze", "silver" and "gold" level of service) representing a high-level of abstraction characterization that is meaningful to the analyst and the analysis tools. More specifically, a *Qualitative NFP* takes a value from a list of allowed values (e.g., an enumeration data type), where each value identifies a possible alternative. When looking in more detail at a *Qualitative NFP*, it may be possible to define it in function of a set of criteria, which may be in turn qualitative or quantitative. Some *Qualitative NFPs* have precisely-known meanings that can be interpreted by the analysis domain, for example the choice of a scheduler type for a processor, or the choice of a statistical distribution for the latency of a network. In both of these examples, the full specification of the property requires not only a qualitative value, but also some quantitative parameters, as for instance:

```
Scheduler-type = roundRobin(quantumSize)
Latency-value = gamma(mean, variance)
```

3. Requirements for NFP annotations

In our context, "annotation" is a process of attaching information to selected UML model elements. We must be able to annotate NFPs to structural elements such as objects and nodes, as well as behavioral elements such as lifelines, execution-occurrences, messages, activities and transitions. We identified different requirements for attaching NFPs to model elements which are described in the rest of the section.

3.1 Variables and Expressions

In most quantitative evaluations, some of the expressed quantities are derived from other quantities. This particularity is so fundamental to quantitative studies that it must be provided in the annotations discussed in this paper. As a motivating example, let us suppose that there is a characteristic size (call it *$dataSize*, in bytes) of a data structure that is stored, retrieved, processed and passed in messages. The CPU cost of operations, the delay for transmitting messages, the memory space required for storage, are all functions of *$dataSize*. It is much easier as well as more informative, to define these quantitative properties by expressions; also, the evaluation is more robust to changes in the design or the usage of the system, that could change the value of *$dataSize*. We can call *$dataSize* an independent parameter of the evaluation.

From this example, it is easy to see that an important requirement is to be able to annotate NFPs not only with concrete values, but also with variable names and expressions. However, defining variable names in the annotation space raises the question of scope. For instance, it should be possible to combine views and diagrams created separately into a single analysis, where the same name may have been used more

than once. Some way to disambiguate these names is necessary. The scoping mechanisms should also handle the problem of UML models that are simultaneously annotated for multiple kinds of analysis.

Another requirement drawn from the above example is that there is a need for *independent evaluation parameters* that may affect many other NFPs through dependencies, which in turn can be expressed through functional relationships. These evaluation parameters need to be attached to the analysis as a whole, either at the level of a UML diagram or at the level of a collection of diagrams.

3.2 Sources of NFPs

It is a peculiarity of the NFPs that the same property may be defined separately from different sources. An obvious example is *required* values, versus *achieved* values, but additional subdivisions may arise. For example, the achieved value may be measured in a certain test (there may be more than one of these for the same NFP), or be estimated by an analytic model. Values may be stated for different execution environments. Input attributes may take *assumed* values based on the expertise of the designer/analyst, and there may be more than one of these (e.g., for worst-case and best-case, or representing the expertise of different parties). The ability to designate different sources and to compare the values given by different sources is fundamental to the full exploitation of the evaluation methodology.

It would be desirable to support user-definable sources, apart from the strings described to convey details. However, for tool support it seems desirable to define a list of standard codes for required and achieved values. It should clearly be possible to define as many versions of a single NFP, from different sources, as necessary. The capability for defining details could be used to list the results of a series of tests or model analyses representing different platforms, or different imposed load levels.

The purpose of expressing different sources is to gather the maximum information from the designer side. Automated analysis tools will have to filter the values according to the kind of data needed for the current analysis.

3.3 Usability of NFPs

Other requirement for NFP annotations is a tradeoff between usability and flexibility. Usability suggests the merit of defining a set of standard NFPs for a given analysis domain, so they can be easily referred to and, consequently, every user of the annotations means the same thing. For NFPs with well-known variants, a set of definitions can be standardized, which cover the important cases with differently-named measures; these can be translated if necessary by domain specialists for the use of an analysis tool with different names. However there are some NFPs whose meaning is model-dependent. This requires a capability for users to define their own NFPs. Thus flexibility and expressive power requires that the users have the capability to define their own quantitative measures, but usability requires a set of standard measures that can be used in straightforward way.

4. Comparing the SPT and QoS&FT Profiles

As mentioned, the background for MARTE comes from two existing profiles: SPT and QoS&FT. While SPT is specifically customized for the real-time systems domain, QoS&FT profile has a broader scope that includes all kinds of QoS properties. The MARTE RFP asks for a full compliance with the UML profile for QoS&FT. It is true that the QoS&FT profile already defines a framework to express NFPs. However, it exist some strong reasons to define a different framework in the context of MARTE:

- In general, the term "QoS" is associated to the aptitude of a service for providing a suitable quality level to the different demands of its clients. The NFPs considered here have a larger extent, and may describe the internals and externals of the system, some of them directly related to the users of resource services and their QoS perception and others not.
- The QoS&FT profile supports modeling of NFPs, with statistical qualifiers and measurement units. However, it ignores some necessary attributes such as measurement sources, property versions, variables, and values defined by mathematical expressions.
- The QoS&FT profile provides a flexible mechanism to store pre-defined QoS Characteristics. However, it requires too much effort for the users due to its three-step annotation process: a) define a *QoS Catalog* with the most common *QoS Characteristics* for each analysis domain, b) derive a *Quality Model* for each application by instantiating template classes from the catalog and c) annotate UML models with *QoS Constraints* and *QoSValues* (which imply the creation of extra objects required just for annotation purposes).

On the other hand, the SPT profile provides a straightforward annotation mechanism through predefined stereotypes end tagged values, and supports already some of the requirements for NFP annotations, such as symbolic variables and expressions through its specialized *Tag Value Language* (TVL). Table 1 compares different features of the two profiles.

Requirement	SPT Profile	QoS Profile
Annotation process	Light-weight	Heavy-weight
Allows for user-defined measures	No (measures are predefined)	Yes (targeted for user-defined measures)
Type for time values	RTtimeValue	No
User-defined delay measure between an arbitrary pair of events	No	No
Expressions for defining quantitative NFPs	Yes Part of the TVL language	No
Quantitative variables and independent evaluation parameters	Yes Part of the TVL language	No
Expressions for defining constraints	Limited	Yes Full power of OCL

Table 1: Comparison of SPT and QoS&FT profiles

In summary, we can say that SPT's modeling method and annotation style are really simple for users (namely light-weight), but its structure is not flexible enough to allow for new user-defined QoS properties or for different analysis techniques. Conversely, the QoS&FT profile's annotation style is more complicated for users (namely light-weight), but its structure is more flexible because of the library style for defining QoS properties, OCL constraints to describe complex QoS functions, and useful qualifiers for QoS properties. In our work, we intend to provide a flexible and straightforward framework for MARTE while adopting the best modeling practices from both profiles.

5. Proposed Framework for MARTE NFP annotations

In this section, we describe our proposal of a NFP modeling framework intended to meet the major requirements stated in Sections 2-4. Figure 3 shows the core UML metamodel to support major NFPs descriptions.

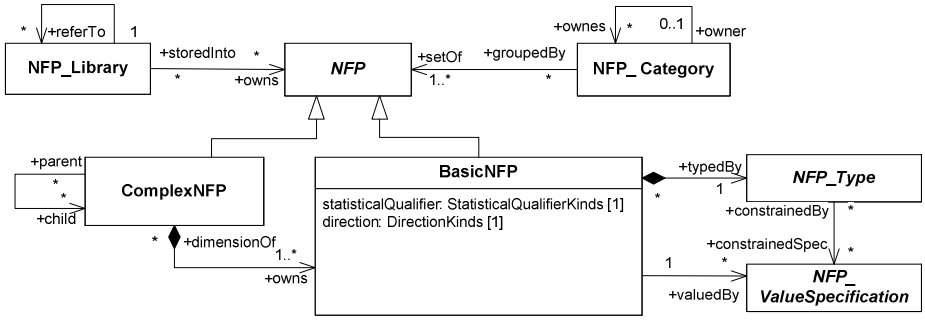

Figure 3. Core NFP: Abstract Syntax

A given Quantitative Analysis domain uses a set of NFPs which are organized in a *NFP Library*. For instance, in the case of software performance analysis, the NFPs are throughput, response time, utilization, CPU execution demand, etc. Likewise, NFPs can be grouped into *NFP Categories*, similarly to the way in which the *QoS Characteristics* are grouped into *QoS Categories* in the QoS&FT profile.

The Core NFP package provides the capability of annotating model elements by *Complex NFP* or directly by *Basic NFP*. The first one is just a constructor, and the second one the concrete holder of NFPs. For instance, we could represent the *Arrival Pattern* property as a data structure (i.e., Complex NFP) that has a number of attributes: *Pattern*, *Period*, *Minimum Arrival Time*, etc. (i.e., Basic NFPs) which will be associated to a concrete value. A Basic NFP can represent either a quantitative property (ultimately a *value* and a *unit*) or a qualitative property (e.g, enumeration type or string). Also, a Basic NFP can be a realization (e.g., a set of values) or a statistical function (mean, variance, etc.).

Thus, Complex NFPs (e.g., response latency, processor throughput, correctness) are a generalization of QoS Characteristics described in the QoS&FT Profile. Basic NFPs (e.g. event period, minimum arrival time, WCET, deadline, scheduling opti-

mally criterion) corresponds to the QoS Dimensions of the QoS Profile. We adopt the attributes *Statistical Qualifier* (e.g., max, min, mean, variance) and *Direction* (e.g. increasing, decreasing) from the QoS profile, but we remove the *Unit* attribute because we are interested on defining the units at the *user model* level.

Each Basic NFP has a *NFP Type* that constrains the specification of their values. At level of user models, we can apply different versions of *NFP Value Specifications* for each Basic NFP.

In Figure 4, we show the domain model for different Basic NFP types. *NFP_type* includes the general attributes *source* (e.g., required, estimated, calculated) and *Language* used for specifying the textual notations of the *Value Specification*. In the same way, specific NFP types use a set of pre-defined *units* (e.g., ms, s, kB/s). *Units* are attributes of most Quantitative NFP and it is important that standard forms are used. For space reasons, we do not show here the predefined units (e.g., duration units, size units). In order to complete the description of different types, the *values* of each particular Basic NFP will be specified according to its NFP Type.

The NFP Type concept proposed here allows for the definition of types for annotating NFP values similar to the *RTtimeValue* type in SPT. However, we propose to use a different taxonomy (Figure 4).

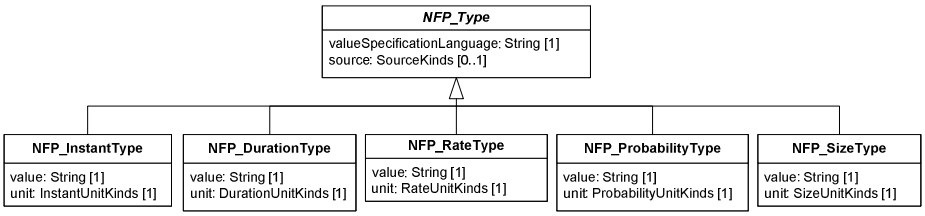

Figure 4. NFP Types: Abstract Syntax

In order to define the legal lexical atoms to specify NFP values, we use the model presented in Figure 5. A *value* can be specified as a constant value (*NFP Constant*), as a variable (*NFP Variable*) or as an expression (*NFP Expression*).

NFP Constant is a literal expression that represents a constant. In addition to the Literal constants supported by UML, we include *List* and *Real* constants. List constants are literals of heterogeneous types that can be combined into a list of items between a set of parentheses, with individual items separated by commas. Notice that, here, we do not define the grammar for the syntax of textual annotations.

NFP Variable can be used as placeholders for results from analysis tools in the UML annotations, or to support relationships between different NFPs. We adopt the SPT's syntax "*$string*" for variable names in the annotation domain, to distinguish them from names used in the UML model itself.

NFP Expressions are used to derive NFPs from other NFPs. An expression can be a simple constant or variable, or it can be a compound expression formed by combining expressions through operators. From an analysis point of view, allowing for NFP Expressions makes the analysis more flexible and more robust to change.

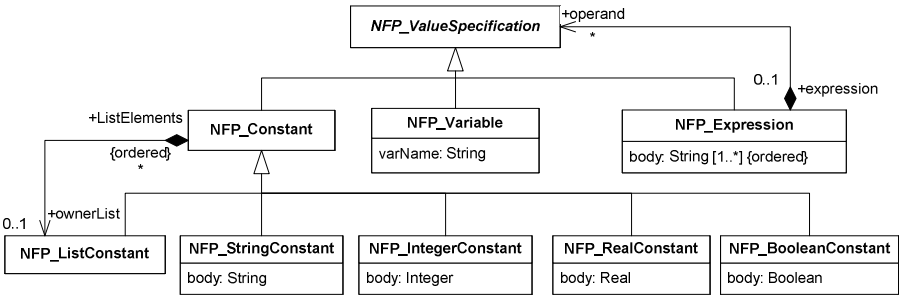

Figure 5. NFP Value Specification Abstract Syntax

Next, we have to define the mechanism for attaching the MARTE annotations to UML model elements while providing flexibility and usability as discussed in Section 3.3. We consider two potential mechanisms: *Tagged Values* and *Constraints*. Tagged values are value slots associated to attributes of specific UML stereotypes, hence, one tagged value characterizes just one model element. On the other hand, a constraint is a condition expressed in natural language text or in a machine readable language (e.g., OCL) for declaring some semantics of one or various model elements. This is useful if we define NFPs that involve more than one element (for instance, a delay between two different events). Thus, we are interested in supporting both mechanisms.

Figure 6 illustrates the alternative in which tagged values are used for annotating NFPs (only a simplified version is shown).

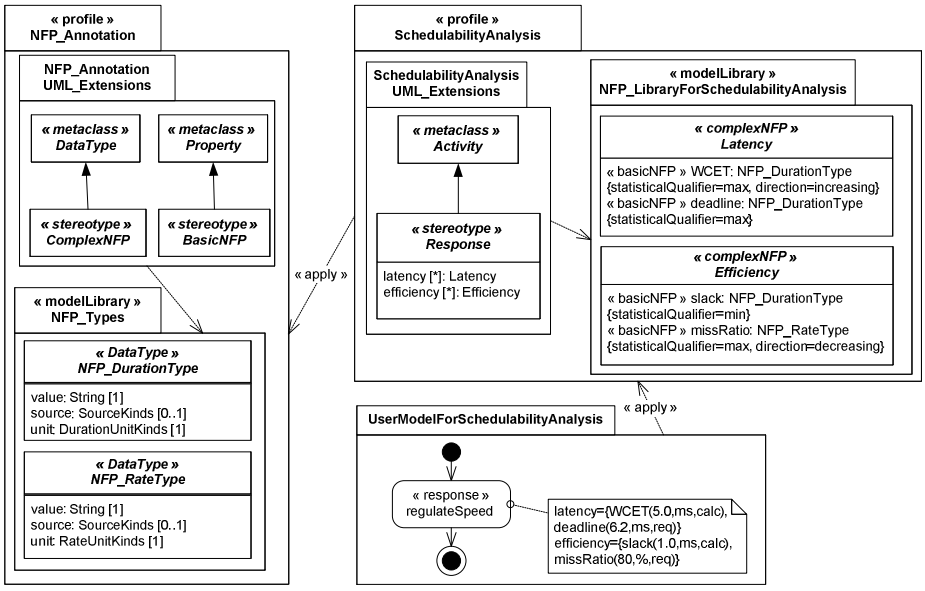

Figure 6: Applying Tagged Values for annotating NFPs

Here, we include a partial view of the *NFP Annotation* profile, including the NFP types used in our example. The Complex NFP concept is extended to *UML DataType*, and the Basic NFP one to *UML Properties*. Thus, Complex NFPs become structured types which are compounded of Basic NFP.

In the example from Figure 6, the NFP annotation profile is applied to a NFP library for Schedulability Analysis. Most features of Basic NFPs are declared in this library, as well as their assigned NFP types. Furthermore, the Complex NFPs defined here are used, in turn, as *types* of generic attributes associated to stereotypes for Schedulability Analysis. For instance, the *response* stereotype has the generic attributes *efficiency* and *latency,* which are typed with the corresponding Complex NFPs.

Finally, we are able to apply the Schedulability Analysis profile, and consequently the underlying NFP library to user models. This structure allows users to attach complex structures of NFPs to UML model elements in a standardized way. Moreover, user-defined NFPs can be added by modifying the existing libraries.

6 Conclusions

This paper defines a framework for annotating NFPs that are necessary for different kinds of quantitative analyses. The relationships between NFPs annotations and UML model elements are discussed. Based on the domain concepts, a list of requirements for attaching NFPs annotations to UML model elements is established. A summary of how the existing SPT and QoS&FT profiles meet these requirements is also presented. The goal is to understand and clarify the premises for some of the requirements in the MARTE RFP, in order to refine them and to make sure that they are consistent, complete and capture all the expressive power needed for a future MARTE solution.

The proposed approach for NFPs annotations involves the adoption of some useful structural concepts (e.g., libraries, categories) and qualifiers (e.g., statistical qualifiers, units) from the UML profile for QoS&FT, as well as its library style (i.e., catalogs) for defining domain-specific NFPs. However, some considerations to reduce its inherent complexity and to facilitate the modeling process are taken. Additionally, some key features provided by the SPT profile are adopted. For instance, we formalize, by means of MOF metamodels, some concepts supported by the TVL syntax to annotate constant, variable and expression values. In this manner, we intended to provide a flexible and straightforward framework for supporting a wide variety of NFPs annotations while adopting best modeling practices of both UML profiles.

Acknowledgements

The research of the Carleton team is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC). The CEA team is partially supported by the PROTES project of the CARROLL French Research program. Huascar Espinoza is supported by the Programme AlBan of the European Union, scholarship No. E04D028544BO.

References

1. Balsamo, S., Di Marco, A., Inverardi, P., Simeoni, M., “Model-based performance prediction in software development: a survey” IEEE Transactions on Software Engineering, Vol 30, N.5, pp.295-310, May 2004.
2. S. Bernardi, S. Donatelli, and J. Merseguer, “From UML sequence diagrams and statecharts to analysable Petri net models” in Proc. of 3rd Int. Workshop on Software and Performance (WOSP02), pp. 35-45, Rome, July 2002.
3. CEA, I-Logix, Uppsala, OFFIS, PSA, MECEL, ICOM, "UML based methodology for real time embedded systems," version 1.0, April 2003, Project IST 10069 AIT-WOODDES.
4. V. Cortellessa, A. Pompei, “Towards a UML profile for QoS: a contribution in the reliability domain”, In Proc. 4th Int. Workshop on Software and Performance WOSP'2004, pp.197 - 206, Redwood Shores, California, 2004.
5. S. Flake, W. Mueller, “A UML Profile for Real-Time Constraints with the OCL” In J. M. Jezequel, H. Hussmann, S. Cook (Eds.) UML'2002, Dresden, Germany LNCS (2460), pp. 179 – 195, Springer Verlag 2002.
6. S. Graf, Ileana Ober, Iulian Ober “Timed annotations in UML”, accepted to STTT, Int. Journal on Software Tools for Technology Transfer, Springer Verlag, 2004
7. A. Lanusse, S. Gérard, F. Terrier, “Real-time Modelling with UML: The ACCORD Approach”, In Proceedings of the UML’98, Springer Verlag LNCS 1618.
8. L. Lavagno, G. Martin, and B. Selic, "UML for Real. Design of Embedded Real-Time Systems," Kluwer Academic Publishers, 2003.
9. D. Lugato, C. Bigot, Y. Valot “Validation and automatic test generation on UML models: the AGATHA approach”, In Proceedings of the Workshop FMICS, ENTCS 66 n°2, 2002.
10. J.L. Medina, M. González Harbour, and J.M. Drake, “MAST Real-Time View: A Graphic UML Tool for Modeling Object-Oriented Real-Time Systems” Proceedings of the 22nd IEEE Real-Time Systems Symposium (RTSS 2001), London, UK, IEEE Computer Society Press, pp. 245-256, December 2001.
11. Object Management Group, “UML Profile for Schedulability, Performance, and Time”, Version 1.1. 2005. OMG document: formal/05-01-02.
12. Object Management Group, “UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE)”, RFP. 2005. OMG document: realtime/05-02-06.
13. Object Management Group, “UML Profile for Modeling Quality of Service and Fault Tolerance Characteristics and Mechanisms”, 2004. OMG document ptc/04-09-01.
14. J. C. Palencia and M. G. Harbour, “Exploiting Precedence Relations in the Schedulability Analysis of Distributed Real-Time Systems”, Proceedings of the 20th Real-Time Systems Symposium, IEEE Computer Society Press, pp 328-339, December 1999.
15. T.H. Phan, S. Gérard and D. Lugato. “Schedulability Validation for UML-modeled real-time systems with symbolic execution and jitter compensation”. ERCT Workshop, 2003.
16. D.C. Petriu, "Performance Analysis with the SPT Profile", in Model-Driven Engineering for Distributed and Embedded Systems, (S. Gerard, J.P. Babeau, J. Champeau, Eds), pp. 205-224, Hermes Science Publishing Ltd., London, England, 2005.
17. B. Selic, “A Generic Framework for Modeling Resources with UML”, IEEE Computer, Vol.33, N. 6, pp. 64-69. June, 2000.
18. Sha, L., Abdelzaher, T., Arzen, K., E., Cervin, A., Baker, T., Burns, A., Buttazzo, G., Caccamo, M., Lehoczky, J., Mok, A., K., “Real Time Scheduling Theory: A Historical Perspective”, Real-Time Systems Journal, Vol. 28, No, 2-3, pp. 101-155, 2004.
19. C.M. Woodside, D.C. Petriu, D.B. Petriu, H. Shen, T. Israr, J. Merseguer, “Performance by Unified Model Analysis (PUMA)”, In Proc. of 5th Int. Workshop on Software and Performance WOSP'2005, pp.1-12, Palma, Spain, July 2005.