The Real-Time UML Standard: Definition and Application

Bran Selic
Rational Software Canada
bselic@rational.com

Abstract

This very short paper describes the objectives, content, and usage of a real-time UML profile that has been standardized by the Object Management Group. This profile defines a comon framework for describing the quantitative aspects of software systems. In addition, it provides specific facilities for analysing real-time systems for schedulability or performance.

1. Introduction

In September 2001, the Object Management Group (OMG) adopted a specification for a standard way of modelling real-time systems using the Unified Modelling Language (UML) [1]. This specification is in the form of a UML "profile" – a set of interpretation rules, markings, and constraints imposed on standard UML to more accurately capture the specific phenomena of a given application domain. The benefits of such standard are clear: interoperability of tools and the emergence of highly reusable common skills and knowledge that extend beyond individual organizations.

This standard goes under the ostensibly omnibus technical title of *"UML profile for schedulability, performance, and time"* [2]. However, the common underpinning is a standard model of *metric* time, that is, time represented as a physical quantity. This model is then used to support two well-established forms of time-based model analysis: schedulability analysis based on schedulability theory and performance analysis based on queuing theory. By introducing quantitative information into a UML model, the model can be analysed to predict crucial time-related characteristics well in advance of committing to costly implementation decisions. This, of course, is the primary purpose of models in most traditional engineering disciplines.

Space limitations preclude any in-depth coverage of this standard. In the rest of this brief paper, we examine the basic objectives of the profile, describe its contents, and conclude with an overview of how it can be applied in practice.

2. Objectives and Approach

A principal requirement was to allow the construction of *predictive* UML models, that is, models that can be used to compute key system properties. A second major requirement was to automate, as much as possible, the formal analysis of such models by a process represented in Figure 1. In this conceptual view, modellers construct UML models of their systems without focusing specifically on analysis (that is, they do not define analysis-specific models). They then attach supplementary quantitative properties to the appropriate model elements to capture temporal values such as delay, frequency, execution time, etc. Such a model can then be transferred from a model-editing tool to a specialized *model processor* for analysis. The standard defines the format and semantics of these supplementary annotations so that models can be exchanged between any tools that comply with the standard. Of course, the same model can be analysed from a number of different perspectives.

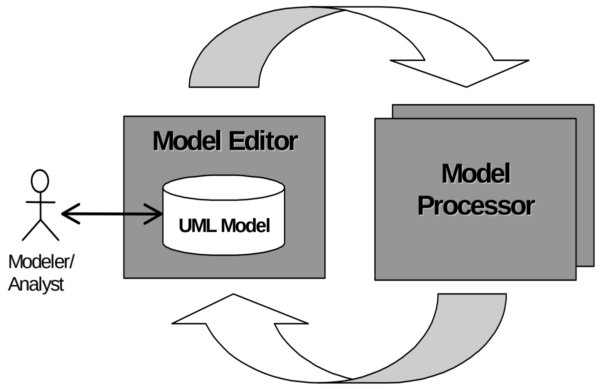

Figure 1. Usage of the profile

Since each model processor *automatically* transforms the UML model into the internal form required by its analysis method, the technical details of the analysis algorithms are hidden from the modeller. This eliminates the need for the rare and complex expertise required to use these techniques (this has been one of the biggest impediments to their more widespread use in practice).

3. Elements of the Profile

The profile itself is defined in layered fashion as described by the UML model in Figure 2.

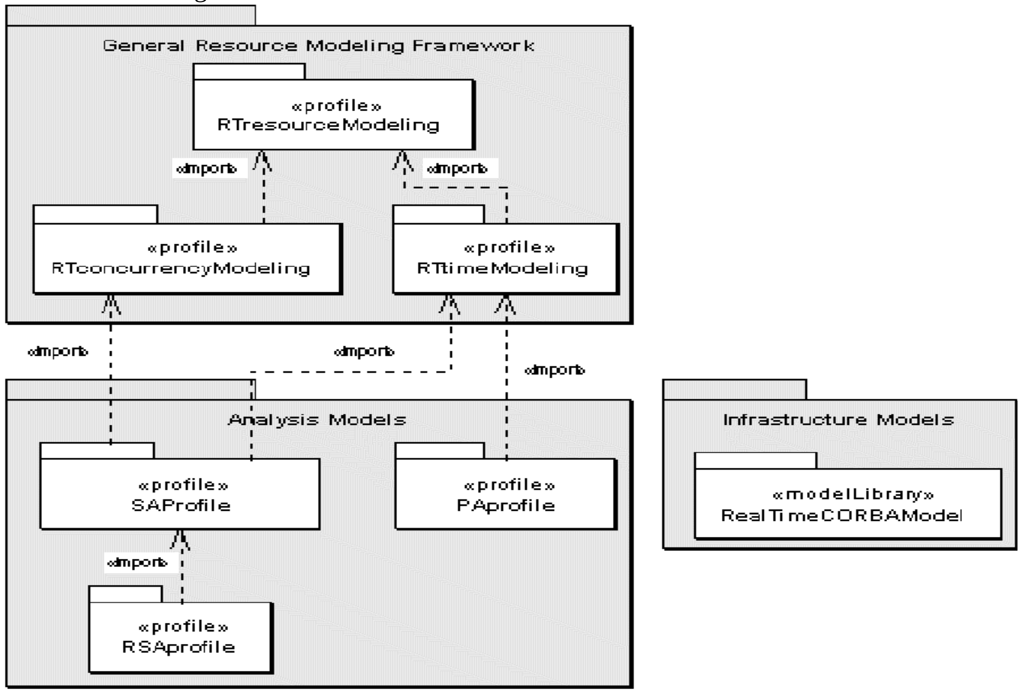

Figure 2: The structure of the real-time profile

The General Resource Modeling Framework package defines the conceptual underpinning for all other parts of the profile. At the core of this framework is a generic model of resource usage relationships that incorporates the notion of *quality of service (QoS)* characteristics (the RTresourceModeling sub-profile). In this model the physical foundation that underlies all software systems is captured through the generic notion of a *resource*. Resources represent the inherent limitations of the physical world; i.e., elements that have finite capacities and limited speeds. This model is a very general and can serve as a basis for all kinds of quantitative analyses, not just those defined in this standard.

The second sub-profile in this core framework contains a very general model of physical and measured time (RTtimeModeling). Both continuous and discrete time models are supported. Also included in this part of the profile are general models of various timing mechanisms such as clocks and timers, which are common in most real-time operating systems.

The third foundational element is a general model of concurrency and concurrency mechanisms. This includes the notion of *active objects* (threads), the carriers of all concurrency in concurrent software systems.

Each of these foundational sub-profiles can be specialized further for specific needs. For instance, many performance analysis techniques have an idiosyncratic model of concurrency mechanisms, which can be derived by constraining the general concurrency model in a suitable way. The same foundation can be specialized differently for the purposes of schedulability analysis.

The various types of analysis-specific models are defined in the Analysis Models package of the overall profile. This package contains three specific sub-profiles:

- The SAProfile package, specializes the concepts of the concurrency modelling and time modelling sub-profiles into a set of generalized concepts commonly used in different schedulability analyses, such as rate-monotonic analysis [1] and others.
- The PAprofile package incorporates a set of general concepts used in performance analyses.
- The RSAprofile package further refines the concepts of the schedulability profile for the very specific case of the real-time CORBA middleware standard. This is useful for analysing the schedulability of applications based on this technology.

Both the schedulability and the peformance sub-profiles can be specialized further to meet specific needs although they are concrete enough that they can also be used directly without further specialization (but, for relatively general analyses). The RSAprofile provides a convenient example of how either individual organizations or other more refined standards could customize these general analysis profiles to support specific analyses.

The final element of the profile is a pre-packaged library of modelling elements (RealTimeCORBAModel), which capture the essential features of real-time CORBA technology. The purpose of this package is to simplify the construction of standard models of applications that are based on this technology. Individual vendors of real-time CORBA middleware can provide their own versions of this library that are annotated with the appropriate QoS information corresponding to their particular realizations. These elements can then be imported directly into user models, saving time and allowing modellers to focus on modelling application-specific parts.

It is anticipated that in the future there will be many other such pre-built models of specific supporting technologies. The intent is to emulate the situation that currently exists in the hardware world, in which chip vendors typically publish VHDL models of their products, so that designers can incorporate them directly into their CAD models.

4. Application of the Standard

In the process of drafting the standard a number of prototypes were constructed to validate the approach. The results proved quite successful. Modellers would develop UML models of their applications using a standard UML editing tool. This was done without specialized knowledge of the analyses to be performed – on the assumption that modellers are not experts in the analysis methods. Using the various concepts defined in the standard (stereotypes and tagged values), such qualitative models were then given quantitative characteristics (arrival rates, delays, capacities, etc.). These models were then transferred to an analysis tool that complied with the standard where they were analysed and the results returned back to the model editing tool.

The standard provides for a fairly sophisticated usage model. For example, it includes the possibility of defining parameterised QoS annotations, that is, annotations whose values are variables. This allows the same model to be analysed for a variety of different values – something that is fundamental in exploring design alternatives. Thus, the modeller does not have to create separate copies of a given model for each different set of QoS values. There is even a facility, based on the standard shareware language Perl, which allows modellers to complex program expressions for computing these values.

The results are generally intended to be fed back to the model-editing tool in the form of UML models in which the computed analysis values are embedded directly. This ensures that the modellers can understand the results of the analysis without having to be experts in the analysis method. There is also a capability to include tool-specific results (e.g., expressed in the form of HTML pages that can be displayed using standard web browsers) for cases where specific tools have additional useful information that is not defined in the standard.

5. Conclusions

The difficulty of software design and in particular the design of real-time and embedded software is exacerbated by a lack of quantitative analysis. It is not unusual for software systems to be designed and implemented that fully meet their functional requirements but that still have to be discarded or redesigned at great expense because they fail to meet their quantitative requirements. The purpose of the real-time UML profile is to provide a standard means whereby developers can take advantage of established quantitative analysis techniques and tools automatically – without necessarily being experts in these techniques. This can help transform the notoriously unreliable discipline of software design into a more mature and more predictable engineering discipline.

6. References

1. Klein, M., Ralya, T., Pollak, B., Obenza, R., and Gonzalez Harbour, M., *A Practitioner's Handbook for Real-Time Analysis: Guide to Rate Monotonic Analysis for Real-Time Systems*, Kluwer Academic Publishers, 1993.
2. Object Management Group, *Response to the RFP for Schedulability, Performance, and Time* (Real-time vendors consortium -- initial submission), OMG document number ad/2001-06-14, June 2001.
3. Object Management Group, *RFP for Scheduling, Performance, and Time*, OMG document number ad/99-03-13, March 1999.
4. Object Management Group (OMG), *The Unified Modeling Language* Specification *(version 1.4)*, OMG document number formal/2001-09-67, September 2001.