

Describing and Analysing Quality of Service using UML

The UML Profile for Scheduling, Performance and Time

As I mentioned in my previous column, the Real-time Analysis and Design Group (RTAD) of the OMG exists specifically to recommend UML extensions in the area of real-time systems. In this column I intend to describe in more detail the Request for Proposal (RfP) for a UML Profile for Scheduling, Performance, and Time [2], issued by the RTAD in 1999.

INTRODUCTION

The request for a UML profile on Scheduling, Performance and Time was the first (and so far only) RfP issued by the RTAD; it aims to plug a significant hole experienced by real-time UML users. An initial submission of a proposed profile was made in September of 2000, and the submitters, of whom I'm one, have since been working on significant revisions. As the due-date for the revised submission, June 18th, 2001 approaches, the submission is becoming a lot clearer, and so now seems a good time to describe it in detail. This column covers the core of the submission and one specific extensions that deals with schedulability analysis.

THE PROFILE

The submitted response is in the form of a profile, a UML concept that is used to package up extensions to UML so that they can be applied selectively to users' models. Figure 1 shows the overall structure of the profile, including its various sub-profiles.

The Core sub-profile is the basis for everything else; it consists of:

- The Resources sub-profile - which enables the addition of Quality of Service (QoS) to any model element, both descriptors (e.g. class) and instances (e.g. object). It also includes a model of resources as servers with associated services, used by clients.
- The Time sub-profile - which describes how to represent time. It defines time characteristics and adds them to core UML concepts, such as action and event.
- The Concurrency sub-profile - defines concurrency, activeness and communication in sufficient depth to represent causality within and between concurrently executing entities.

The two infrastructure sub-profiles that are mandated in the RfP are RT CORBA and Enhanced View of Time because these are both existing OMG infrastructure standards. However, we expect infrastructure providers (e.g. ORB and O/S vendors) to add their own sub-profiles.

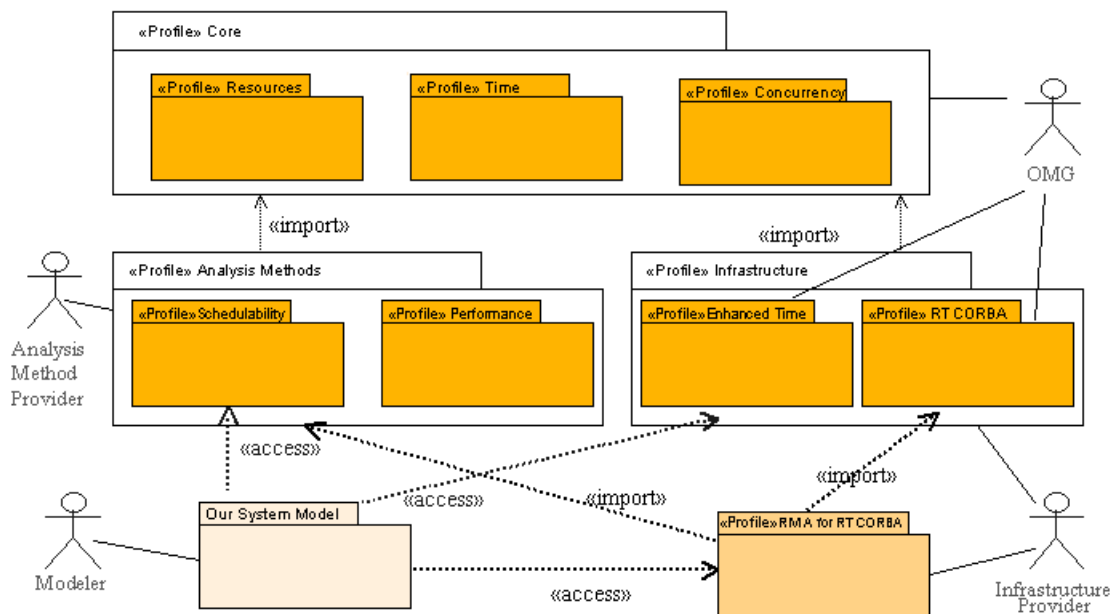


Figure 1. Structure of the submission.

files subsequent to the adoption of this submission.

Schedulability is the most advanced analysis sub-profile so far. It builds on the Core, but adds specific capabilities to resources, actions and events in order to represent the QoS required for calculating worst-case scenarios. A sub-profile for more general performance analysis is also underway.

Figure 1 also shows two examples of how the profiles might be used: the modeler might either choose to incorporate elements from both the infrastructure they're using and the analysis technique they're using and combine them in their model. If a particular combination of analysis technique and infrastructure was particularly popular then a special profile might be created that combined both; any modeler that used both could then use this new profile.

The detailed content of the profiles is described in the following sections.

PROFILES IN UML

From the UML [1] - "The UML provides a rich set of modeling concepts and notations that ... meet the needs of typical software modeling projects. However, users may sometimes require additional features beyond those defined in the UML standard. These needs are met in UML by its built-in extension mechanisms that enable new kinds of modeling elements to be added to the modeler's repertoire as well as to attach free-form information to modeling elements."

The principal extension mechanisms are:

- Tag Definition - which is used to specify new kinds of properties that may be attached to model elements. The actual properties of individual model elements are specified using Tagged Values. These may either be simple data type values or references to other model elements.
- Stereotype - which is a model element that defines additional values (based on tag definitions). Any model element may be "branded" by one or more particular stereotypes, in which case it receives the tag values of the stereotypes in addition to the attributes, associations and super classes that it has in the standard UML.
- Profile - which is a stereotyped package that contains model elements that have been customized for a specific domain or purpose by extending the UML metamodel using, primarily, stereotypes and tag definitions.

RESOURCES

This sub-profile describes an essential basis for modeling real-time systems using UML. At the core of this sub-profile is the notion of Quality of Service (QoS), which provides a uniform basis for attaching quantitative information to UML specifications. QoS may come in many forms, such as capacity, availability, performance, but this submission is primarily focused on QoS to do with time.

The underlying model of resources is a client/server model; resources can be thought of as servers with offered QoS, such as the length of time required to

acquire and release the resource. At a more detailed level, resources offer services, often represented as operations in UML; services too offer QoS, such as response time. Users of resources are called "resource clients" and have required QoS, such as deadlines for service completion.

A good example of a resource is a semaphore, which offers services, 'get' and 'put', which acquire and release the resource; a counting semaphore has a 'capacity' quality of service, which denotes the maximum number of concurrent users.

TIME

Time is of course an ever-present aspect of real-time systems, and it has impact in several different areas. The submission creates stereotypes of certain key UML concepts, such as Action and Event, which can be used to add timing patterns at a very detailed level in UML models. An Event will have a time stamp, and an Action will have duration. Relative times are dealt with by relating Events and Actions to a common time-base, for example a system clock.

The submission also adds fundamental timing patterns of different kinds that are essential in supporting schedulability and performance analysis; they include modeling whether an event recurrence is periodic or not, and in the former case also modeling period, distribution functions, and jitter.

CONCURRENCY

The Concurrency sub-profile enables modelers to describe a rich model of concurrently executing and communicating objects, which includes explicit detail of causality. There are two principal contributors to causality in the model: sequencing of actions and interaction between objects.

A procedure is the root of an action sequence, an ordered a set of actions, some of which contain other action sequences. In this case, causality can be defined using these ordered collections of actions at ever-deeper levels of nesting. An "active unit" is an entity (e.g. an object) with one main procedure (a.k.a. thread), which executes concurrently with those of all other active units.

Causality between procedures is defined by the use of communication primitives. If a class offers a service this indicates that the class will provide at least one procedure that handles a request for that service (use of mechanisms such as inheritance may result in more than one). A message action is a specific type of action that requests a service by sending a message to an object of the class offering that service. The message is hooked into the appropriate handler once received. Although there may be multiple potential handlers for a service request at the descriptor (class) level, there will only be one at the instance (object) level, providing a causal link between the procedure issuing the message action and the handling procedure.

There are two choices at either end of the communication, which affect the detailed causality between concurrent threads of control.

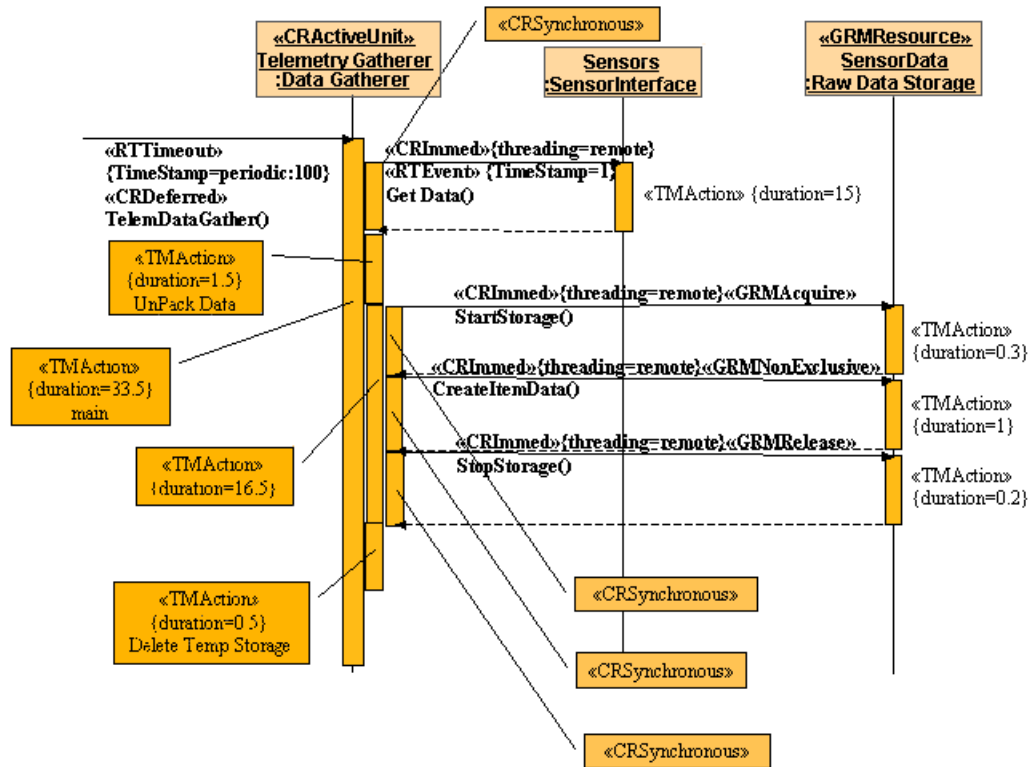


Figure 2. Example of the application of the Core.

At the server end, the service request may either be handled immediately, or deferred. In the immediate case, a further property describes whether the object creates its own thread to execute the handling procedure, or assumes that there is an existing thread available.

At the client end, the message action may either represent an asynchronous or synchronous invocation of the service. If the request is asynchronous then the client proceeds immediately; if the request is synchronous then the client waits for a response before continuing.

Figure 2 shows a sequence diagram, which has been annotated with several stereotypes from the core profiles. Stereotypes are indicated using guillemets (""); tag values are listed in braces after the stereotype indication. Each profile provides a different stereotype prefix: "GRM" for Resources, "RT" for Time and "CR" for concurrency.

- 'SensorData' is a "GRMResource", which supports a number of services, 'StartStorage', which is stereotyped "GRMAcquire" because it acquires the SensorData resource, 'StopStorage', which is stereotyped "GRMRelease" because it causes the resource to be released, and 'CreateItemData', which can afford to be "GRMNonExclusive" if bracketed by resource acquisition and release.
- 'TelemetryGatherer' is a "CRActiveUnit" whose main thread processes the 'TelemDataGatherer' timeout; the 'main' procedure simply waits for the timeout, which occurs periodically at 100ms intervals.
- Many of the communication actions that access resources are synchronous, hence the stereotype

"CRSynchronous", with the resource handling the requests immediately on a remote thread, hence "CRImmed"["threading=remote"]. This corresponds to the semantics of a standard procedure call or RPC.

- The actions and events (as appropriate) have been given time annotations - durations in the case of actions, e.g. for "UnPackData" and "main" itself, and times for the events, such as [TimeStamp=periodic:100] for the "TelemDataGatherer" event (often event timestamps are not simple values but may be periodic or statistical distributions).

SCHEDULABILITY SUB-PROFILE

Real-time systems are systems where the question of when a response to an event occurs is as important for the correct behavior of the system as its algorithmic behavior. A common distinction in this regard is between soft and hard real-time, the difference being that in the former a late reply is a good reply as long it is within some acceptable range, while in the latter case a late reply is useless, and sometimes totally unacceptable (fatal). In the former category, statistical prediction is a way to model the relevant characteristics, which can usually be quantified as average performance and standard deviation. In the latter category, the problem is usually in making sure that there is an upper bound for response to a stimulus, i.e., there has to be a guaranteed response time. Actual systems often combine traits of both categories.

The Schedulability sub-profile particularly focuses on systems having hard timeliness requirements, and how to annotate the model in ways that allow a wide variety of schedulability techniques to be applied. The

goal is of course to make it possible to determine whether or not a model is schedulable, i.e., if it will be able to meet its guaranteed response times. This is the most developed of the analysis sub-profiles. Although, it is particularly intended to support modelers who wish to analyze hard real-time constraints, it will also handle soft real-time constraints.

The key concepts in schedulability analysis are:

- A Schedulable Entity, which represents a load on the system. It consists of an initiator, or trigger and a sequence of actions, or response. A trigger specifies its occurrence pattern and deadline; an action specifies its utilization of an Execution Engine (work) and its use of resources.
- An Execution Engine, which represents the capability of the system to do work. It specifies its scheduling Policy, e.g. FIFO, Fixed Priority and such performance attributes as context switch time and processing rate.
- A Resource, which represents any resource, other than the Execution Engine, needed by Schedulable Entities while responding to their triggers. It specifies its arbitration policy, e.g. Priority inheritance, FIFO, Highest Lockers, whether it is preemptable or not, and the setup and access overheads associated with its use.

Figure 3 shows a collaboration diagram, which has been annotated with stereotypes from the schedulability sub-profile, hence the "SA" prefix.

A typical series of analysis steps is:

1. Create a behavioral model including concurrency, contention, and behavior
2. Include quantifiable characterization of contention, blocking, and execution times
3. Analytically predict worst case response time using

Rate Monotonic Analysis (or similar technique)

Typically, the analyst will iterate through the above steps until the analysis predicts an acceptable outcome (typically that all of the deadlines can be met).

The stereotypes in the Core sub-profiles are used during step 1 to describe the underlying behaviour of the system. Then the "SA..." stereotypes are used to add the appropriate quantities for step 2.

The three main objects are "SASchedulableResource"es, which correspond to Schedulable Entities, all using an "SAUsedResource", 'SensorData'. There are three events of interest, designated "SATrigger", each causing specific responses, which run on the 'main' thread of their respective schedulable resources (schedulable resources must be active).

The worst-case response time for a response, the key measure of system load, can be calculated from the raw execution time (as if the response was being executed on its own exclusive execution engine) and the time spent either being blocked waiting for a resource (in this case 'Sensor Data'), or preempted by another higher priority "SASchedulableResource" (note the priority attached to the "SAResponse" stereotype). In this case an analysis has already been performed assuming a single Execution Engine (not shown) and all three triggers have been found to be Schedulable, the worst-case response time of their respective responses having been calculated at less than the deadline of their triggers.

ISSUES

Although the profile mechanism in UML has been sufficient for most of our needs, there are two major issues that require revision to the UML itself:

- Modeling actual situations using instances - UML has a complete descriptor model (classes, opera-

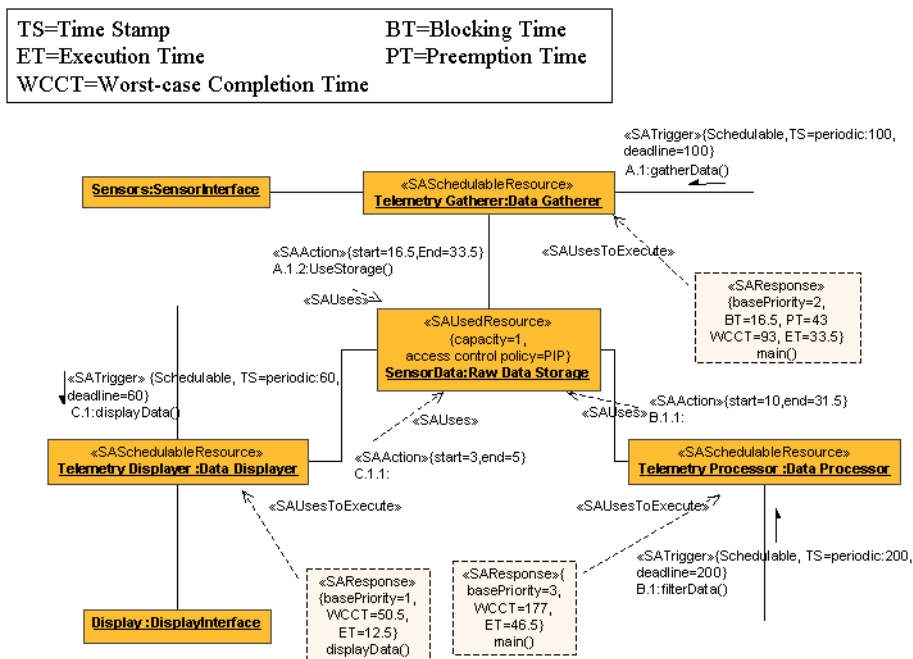


Figure 3. Schedulability Example.

tions etc.), but not a complete instance model (instances, event occurrences, etc.). The submission suggests that the concept of action execution, i.e. the specific execution of a defined action, be added, but this is just a 'patch'; a much more comprehensive solution is needed.

- Parameterized models - a given analysis may require many executions of the same scenario, with slightly varying parameters, in order to find an optimal solution. The submission suggests a textual convention to identify parameters and a model post-processor to plug in actual parameter values. However, what is needed is a general capability to parameterize composite UML objects, such as package and collaboration, so that a modeler can specify and view parameters within a UML context.

I hope that both of these issues will be addressed in UML 2.0.

CONCLUSION

I've presented in this column a very brief run through of the submission to add timeliness and schedulability information to UML. ■

Alan Moore has 15 years of experience in the development of real-time and object-oriented methodologies, and their applications in a variety of problem domains. He has been actively involved in product development, training and consulting related to OOAD and structured development tools during that time. Alan has co-authored a book on GUI design and published several papers, and has lectured on a wide variety of analysis and design issues.

Alan is responsible for the specification, planning and management of the ARTiSAN product strategy. He is the author of ARTiSAN Real-time Perspective, a pragmatic approach to the development of real-time systems and is an active participant in the Real-time Analysis and Design Group (RTAD) of the Object Management Group (OMG). (website: <http://www.artisansw.com>)

REFERENCES

1. OMG, Unified Modeling Language Specification (draft), version 1.4
2. UML Profile for Scheduling, Performance, and Time RFP (OMG document ref: ad/99-03-13)

ADVERTISEMENT INDEX

QNX SOFTWARE SYSTEMS	2
ENEA OSE SYSTEMS AB	23
NATIONAL INSTRUMENTS	51
TEXAS INSTRUMENTS	91
VMIC	115
VMETRO	116

For more information on advertisement opportunities, call +32-2-520.55.77 or send an e-mail to info@dedicated-systems.com

EDITORIAL CALENDAR

- 4Q 01** System Architectures
(SoC, Software Intensive Systems, Complexity issues, Networking..)
- 1Q 02** Aerospace & Automotive
(Fieldbus, PLC's, SSD, Embedded Platforms, Automation Control, Hard Real-Time Control, ...)
- 2Q 02** Manufacturing