The “*UML Profile for Schedulability, Performance and Time*” in the Schedulability Analysis and Modeling of Real-Time Distributed Systems[1]

Julio L. Medina, Michael González Harbour, and José M. Drake

Departamento de Electrónica y Computadores, Universidad de Cantabria
Avda. de los Castros s/n, 39005 - Santander, SPAIN
{medinajl, mgh, drakej}@unican.es

ABSTRACT

In this paper we address some aspects of the *“UML Profile for Schedulability, Performance and Time”* (SPT), that have been found to be improvable. In particular, we describe problems and possible solutions related to the usage of the profile in the representation of schedulability analysis models for real-time distributed systems. The issues appear mainly from a comparison between the SPT and the concepts used in MAST (*Modeling and Analysis Suite for Real-Time Applications*).

1. INTRODUCTION

Even though the *“UML Profile for Schedulability, Performance and Time”* (SPT) [14][13] has been adopted as a specification of the Object Management Group (OMG) there still are some pending issues, most of which are already posted [1] and are in the course of discussions to be possibly incorporated in the next revised version of the profile.

This brief position paper provides arguments to supports some of the pending issues, presents some additional ones, and comments on the intended purpose of the profile. All these issues arise from the experience in the development of MAST (*Modeling and Analysis Suite for Real-Time Applications*) [3][18] and UML-MAST [10][2]. MAST is a conceptual modeling framework and a set of tools for schedulability analysis of real-time distributed systems. It supports UML-MAST, a set of mappings, concepts, and a tool for the representation of real-time analysis models in UML and the generation of MAST models from UML-CASE tools.

Considering that MAST tools and concepts are in the scope of the SPT profile, in the development of the UML-MAST meta model we have endeavoured to keep semantic compatibility as far as possible. Unfortunately this has not been strictly achieved, mainly because the SPT profile limits some important aspects of the MAST conceptual model. Adapting MAST to these limitations would have implied either loosing capacity to model certain types of systems or internally redefining concepts of a higher abstraction level, concepts that clearly deserve its own entity in the SPT profile.

The next section describes very briefly UML-MAST. Section 3 shows and explains the limitations found in the SPT profile as well as the respective proposals. Section 4 comments on other more general aspects of the profile that could be improved, though no concrete proposals are provided. Finally Section 5 makes some short concluding remarks.

2. UML-MAST

UML-MAST is a UML-based set of modeling concepts and a tool for representing the real-time behaviour of distributed systems. At present, single and multiple processor systems with fixed priority scheduling are supported by the MAST tools. There are different flavour of scheduling such as preemptive, non-preemptive, sporadic server and polling scheduling policies. The MAST toolset can be used for modeling and schedulability analysis of real-time applications.

One of the main features of the UML-MAST modeling approach is that it separates the definition of the models of the platform (processors, networks, timers, network drivers, schedulers), the application software (operations, messages, shared resources, threads), and the real-time situations (external events, transactions, timing requirements).

The real-time models are formulated in UML and they are managed like a “Real-time View” of the system, which is complementary to those used for the software development process. The tool that generates and extracts the MAST model from a UML CASE tool has been implemented and integrated to Rational Rose’2000e like a pseudo Add-in.

UML-MAST is a tool and a modeling environment representative of those which are the target of the SPT profile. In particular it can be considered as an implementation/extension of the *SAprofile*.

3. LIMITATIONS AND PROPOSALS

Most of the limitations we have found seem to be based in the conceptualization of schedulability analysis techniques as they were in their beginnings. These techniques have evolved significantly along the last decade, particularly those oriented to fixed priority scheduling schemes, being these the most widely used in commercial and standard operating systems and concurrent languages. Although the first widely known techniques (RMA) were suited for single-processor systems [9][7][8][6][4], at present a growing range of techniques enable analysis in distributed systems [11][12][17][5]. Particularly now that real-time communications infrastructure is more available [15][16].

It is upon these premises and by observing the evolution of the SPT profile along the successive versions up to its final adopted specification, that the limitations described next arise.

1. This work has been partially funded by the Comisión Interministerial de Ciencia y Tecnología of the Spanish Government under grant TIC2002-04123-C03-02

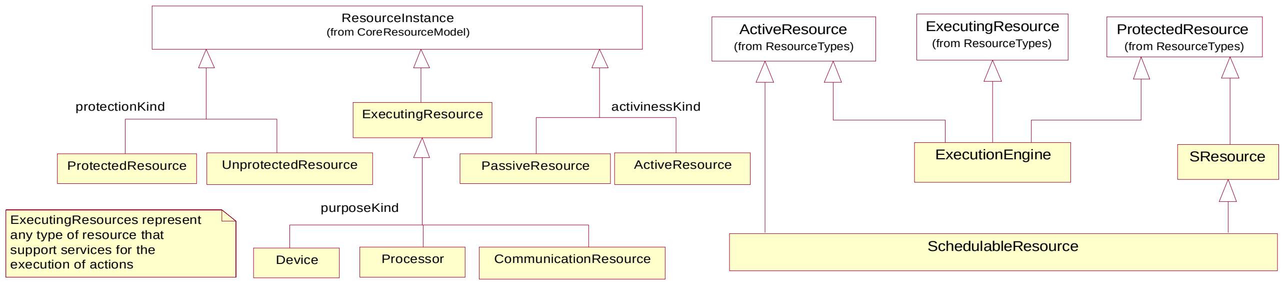

Fig. 1. Extension of ExecutionEngine to enable the holistic analysis of distributed systems.

3.1. Definition of *ExecutionEngine*

3.1.1. Limitation:

ExecutionEngine was formerly required to be a *Processor*, excluding it from being a *CommunicationResource* or *Device*, something that in a holistic analysis would enable the actions in a *SchedulingJob* to represent code execution as well as messages on networks. After some discussion a text relaxing the definition was introduced (see 6.1.3.3), and explanatory phrases in 6.1.1 make the concept more general than *Processor*, but the diagram in Figure 6-2 still shows the inheritance relationship.

3.1.2. Proposal:

This seems to be a “typo”, so just removing the inheritance relationship in Figure 6-2 would make the text consistent. If that solution appears not specific enough, something like the proposal in [5706] shown in Fig. 1 should be considered.

3.2. Relationship between *SchedulingJob* and *Trigger*

3.2.1. Limitation:

The association between a *SchedulingJob* and its *Trigger* was formerly one-to-one. It meant that a *SchedulingJob* could be stimulated by one and only one *Trigger*. There are cases in which several *Triggers* can be involved in a *SchedulingJob* and viceversa: One *Trigger* launching several *SchedulingJobs*, a *SchedulingJob* invoked alternatively by different *Triggers*, or a complex triggering pattern resulting from the combination of events that occur at different stages along the execution of the *SchedulingJob*. The adopted specification allows several *Triggers* for a *SchedulingJob*, each with its own particular *Response*, considering different *Responses* as part of the *SchedulingJob*. These multiplicities do not reflect the underlying semantics.

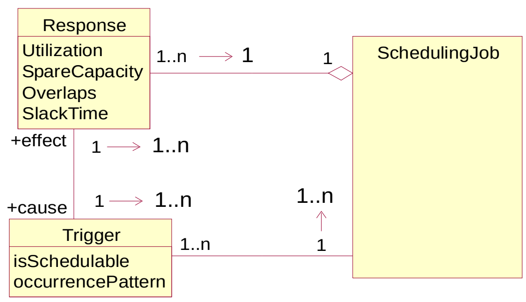

Fig. 2. Relation between SchedulingJob and Trigger

3.2.2. Proposal:

The association between the *SchedulingJobs* and its *Triggers* should be multiple in both directions, see Fig. 2.

3.3. Sequences of *SActions*

3.3.1. Limitation:

SActions inherit from *TimedAction* and transitively from *Scenario*, though they do not inherit *predecessor*, neither *successor* nor *requiredQoS* properties from *ActionExecution*, so they can not be put in sequence or demand concrete *QoSvalue* from their resources.According to the dynamic resource usage model shown in Figure 3-8 (reproduced here in Fig. 3-right) a *Scenario* may contain an ordered sequence of *ActionExecutions*, and these in turn may be connected to others by means of the *predecessor* and *successor* roles or contain others since they are a specialization of *Scenario*. The desired model for a *SAction* is such that it allows for connection to others in sequence as well as encapsulate the behaviour of others, which in turn can be allocated to different *SchedulableResource* and/or have their own scheduling parameters.

3.3.2. Proposal:

Make *TimedAction* a specialization of *ActionExecution* [5720], instead of inheriting directly from *Scenario*, this allows *SAction* to inherit both behaviors. See in the left side of Fig. 3.

3.4. Relationship between SAction and SchedulableResource

3.4.1. Limitation:

An *SAction* is deployed to one and only one *SchedulableResource* (host). This limits the definition of distributed *SActions*, whose internal *SActions* could not be effectively deployed by different *ExecutionEngines*. This became already relaxed in the *host* attribute definition of *SAction* (6.1.3.2), but the “just one” multiplicity shown in Figure 6-1 misleads.

3.4.2. Proposal:

Have a *0..1* multiplicity in the *host* role of the association between *SAction* and *SchedulableResource* [5721]. Fig. 4(a).

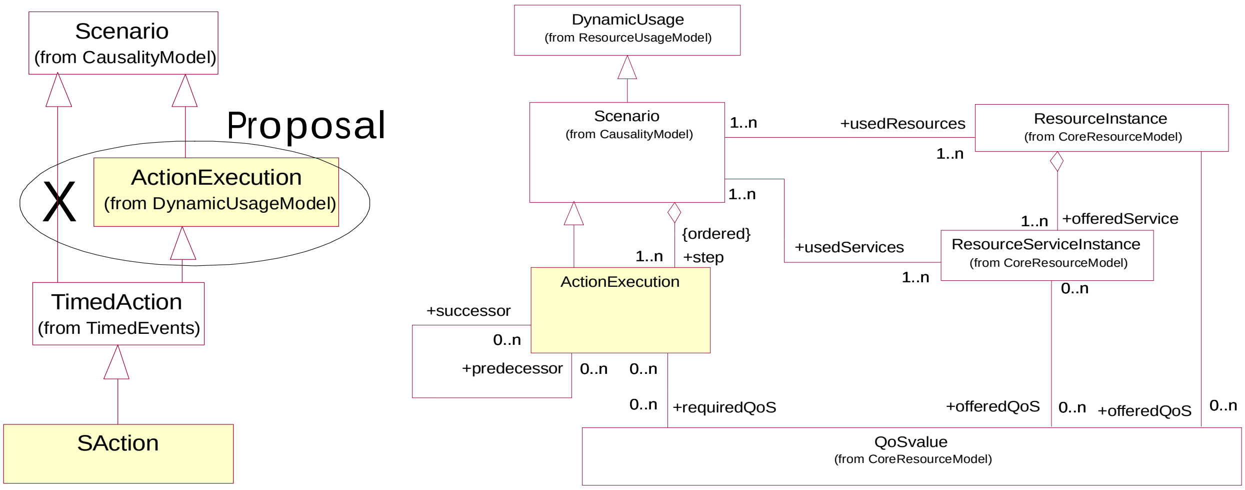

Fig. 3. Enabling sequences of SAction and the usage of QoSvalues

3.5. Duration of a *SAction*

3.5.1. Limitation:

Inherited from *TimedAction*, *SActions* have a unique timing attribute, whether expressed by means of the *duration* or the *start* and *end* attributes. This characterization is not sufficient to feed advanced schedulability analysis techniques like the offset based ones [12][11], which take advantage of the minimum known duration of the actions.

3.5.2. Proposal:

The *duration* attribute of *TimedAction* can be considered a list [5711] of generic *durationCharacteristics*, in which the meaning of the values can be tailored by the implementation, see Fig. 4(b). Another more specific possibility is to define at least three concrete values: WCET, BCET and ACET, worst best and average case execution time, respectively. The *start* and *end* lists may keep their present semantics or adapt consistently to the new duration definition.

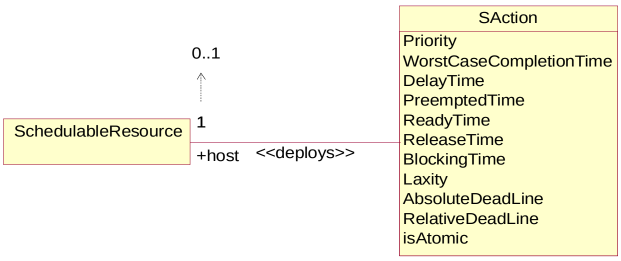

Fig. 4. (a) *SchedulableResource* and *SAction* relationship

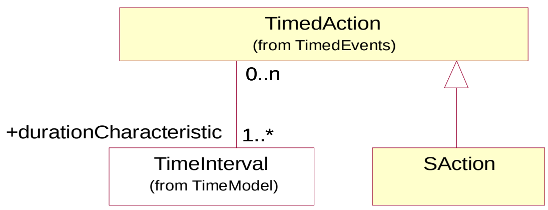

Fig. 4. (b) Multiple association that allows for several kinds of timing parameters in actions

3.6. Modeling branching and merging sequences

In the reasonable presumption that the proposal in section 3.3.2. above is accepted, the next limitation arises on the connectivity possibilities given to the *predecessor* and *successor* roles of an *ActionExecution*.

3.6.1. Limitation:

Successor-predecessor relations between *SActions*, inherited from *ActionExecution*, allow only for *join* (input AND) and *fork* (output AND) relationships, excluding others like *merge* (input OR) or *branch* (output OR). Although this is consistent with the general instance-based intended usage context to which the profile seems to be oriented, it is an unnecessary restriction of its modeling power. It would be very useful to have in just one model all the alternative control flow lines that a scenario could display. There are schedulability analysis tools [5] that can manage this kind of models. Besides it would allow the same model to be used in simulation or even to feed worst case execution time calculation tools.

3.6.2. Proposal:

Extend the *successor* and *predecessor* roles semantics, to allow also the *merge* and *branch* possibilities.

4. OTHER COMMENTS TO THE SPT PROFILE ORIENTATION AND USAGE STYLE

4.1. Modeling software components instead of only software execution

Exploring the possibilities of the profile in the generation of higher abstraction level extensions, we consider interesting to reduce the emphasis on the instance-based nature of real-time analyses, clearly expressed in the profile, in favour of the descriptors that support the respective concepts and are more suitable for reusability and parameterization. We found this particularly important in the context of real-time component-based software engineering models and also in the modeling duties of the infrastructure provider.

4.2. Timer effects and other operating system services

The processing capacity of an *ExecutionEngine* is reduced and affected by the operating system services that run on it. These can be modeled as part of the application, but they are usually known in detail only by the infrastructure provider, so it would be useful to have at least the most frequent interaction characterized: the effect of the timer resolution in the execution of activities (alarm clock or ticker), the overload of the network drivers, the interruption mechanism, etc. So far, all these seem to be specializations to be included in more specific profiles but it is interesting to see how general they are. In particular the possibility of having more complex scheduling mechanisms (possibly hierarchical) should be considered in the multiplicities shown in Figure 6-3, as well as the fact that in distributed systems the schedulable entity for a scheduler is not the *SchedulingJob* but the *SchedulableResource* instead.

4.3. The usefulness of a real-time view

Using stereotypes and tag values spread all over any entity of the UML model, has several drawbacks. The real-time aspects of the model become quickly unreadable and unexpressive for a human observer, particularly in big applications with a relatively small but not negligible part having real-time requirements. In addition, stereotypes and tag values are not fully supported by all the tools.

In our opinion a better representation would be the packaging of all real-time related aspects in a separate view, complementary to the others considered. This would make maintenance and management easier. As for the UML mapping of concepts, the usage of classes/objects to represent descriptors/instances, attributes for QoScharacteristics, and ActionStates for SActions, has the advantage that it uses notational resources widely available in many UML-CASE tools.

5. CONCLUDING REMARKS

The UML SPT profile supports most of the necessary concepts for building real-time analysis models for applications and software infrastructure. Some limitations have been detected on its usage in applications that use communication resources and have end-to-end timing requirements. The construction of specific UML real-time analysis models instead of annotating other user-defined models is proposed, as well as the mapping of concepts to conveniently stereotyped classes, attributes, actionstates or other semantically closer UML conceptual and notational entities.

BIBLIOGRAPHY

[1] Pending Issues sent to the OMG Finalization Task Force corresponding to the "Schedulability Performance and Time", profile `http://www.omg.org/issues/uml-scheduling-ftf.html.`

[2] J.M. Drake and J.L. Medina: "UML-MAST Metamodel, specification, example of application and source code". `http://mast.unican.es/umlmast`

[3] M. González Harbour, J.J. Gutiérrez, J.C.Palencia and J.M. Drake: "MAST: Modeling and Analysis Suite for Real-Time Applications". Proceedings of 13th Euromicro Conference on Real-Time Systems, Delft, The Netherlands, IEEE Computer Society Press, pp. 125-134, June 2001

[4] M. González Harbour, M.H. Klein, and J.P. Lehoczky. "Fixed Priority Scheduling of Periodic Tasks with Varying Execution Priority". Proceedings of the IEEE Real-Time Systems Symposium, December 1991, pp. 116-128

[5] J.J. Gutiérrez García, J.C. Palencia Gutiérrez, and M. González Harbour, "Schedulability Analysis of Distributed Hard Real-Time Systems with Multiple-Event Synchronization". Euromicro Conf. on Real-Time Systems, Stockholm, Sweden, 2000.

[6] M. Joseph and P. Pandya, "Finding Response Times in a Real-Time System," *BCS Computer Journal*, Vol. 29, no. 5, October 1986, pp 390-395.

[7] M. Klein, T. Ralya, B. Pollak, R. Obenza, and M. González Harbour, "A Practitioner's Handbook for Real-Time Systems Analysis". Kluwer Academic Pub., 1993.

[8] J.P. Lehoczky. "Fixed Priority Scheduling of Periodic Tasks Sets with Arbitrary Deadlines". *IEEE Real-Time Systems Symposium*, 1990.

[9] C.L. Liu, and J.W. Layland, "Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment". Journal of the ACM, 20 (1), pp 46-61, 1973.

[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" RTSS'01, London, December, 2001.

[11] J.C. Palencia, and M. González Harbour, "Schedulability Analysis for Tasks with Static and Dynamic Offsets". Proc. of the 19th IEEE Real-Time Systems Symposium, 1998.

[12] J.C.Palencia, and M. González Harbour, "Exploiting Precedence Relations in the Schedulability Analysis of Distributed Real-Time Systems". Proceedings of the 20th IEEE Real-Time Systems Symposium, 1999.

[13] B. Selic, "A Generic Framework for Modeling Resources with UML". IEEE Computer, Vol.33, N. 6, pp. 64-69. June, 2000.

[14] Bran Selic, Alan Moore, Murray Woodside, Ben Watson, Morgan Bjorkander, Mark Gerhardt and Dorina Petriu: "UML Profile for Schedulability, Performance and Time". OMG document formal/03-09-01 September 2003. `http://www.omg.org/technology/documents/formal/schedulability.htm`

[15] J.K. Strosnider, T. Marchok, J.P. Lehoczky, "Advanced Real-Time Scheduling Using the IEEE 802.5 Token Ring". Proceedings of the IEEE Real-Time Systems Symposium, Huntsville, Alabama, USA, pp. 42-52, 1988.

[16] K. Tindell, A. Burns, and A.J. Wellings, "Calculating Controller Area Network (CAN) Message Response Times". Proceedings of the 1994 IFAC Workshop on Distributed Computer Control Systems (DCCS), Toledo, Spain, 1994.

[17] K. Tindell, and J. Clark, "Holistic Schedulability Analysis for Distributed Hard Real-Time Systems". Microprocessing & Microprogramming, Vol. 50, Nos.2-3, pp. 117-134, 1994.

[18] MAST project web page, documentation, download and related tools. Real-Time and Computers Group. University of Cantabria - Spain. `http://mast.unican.es/`

[XXXX] Pending issue XXXX: `http://www.omg.org/issues/uml-scheduling-ftf.html#IssueXXXX`