UML-MAST: Modeling and Analysis Methodology for Real-Time Systems Developed with UML CASE Tools[1]

By J.L. Medina (medinajl@unican.es)
J.M. Drake (drakej@unican.es)
M. González Harbour (mgh@unican.es)
Group of Computers and Real-Time. University of Cantabria. Spain.

Keywords: Real-Time, UML-RT, Schedulability Analysis, Object-Oriented Real-Time Systems.

Workshop: Real-Time Specification for Embedded Systems.

I Introduction.

This paper describes a methodology for building an analyzable real-time model of an object-oriented system that is developed using a UML CASE tool. By using it, the designer is able to build models that can be used to analyze and predict crucial real-time properties of the system.

UML-MAST is based on concepts defined in the Modeling and Analysis Suite for Real-Time Applications (MAST) [1] and [2]. This suite is still under development and its main goal is to provide an open source set of tools that enables engineers developing real-time applications to check the timing behavior of their application, including worst-case schedulability analysis for hard timing requirements and discrete-event simulation for soft timing requirements.

Currently, a group of OMG member companies is elaborating the "UML Profile for Schedulability, Performance and Time" [3] that enables the construction of real-time models for making quantitative predictions about the timing and scheduling properties of an application designed with UML tools. Its adoption will be useful to facilitate communication of design artifacts between developers in a standard way and to enable interoperability among different analysis and design tools. UML_MAST is within the scope of the OMG profile, and specially within the Schedulability and Performance Analysis sub-profiles. At present, the UML-MAST methodology does not follow the nomenclature and the UML representation rules suggested of those profiles, but as both approaches share the same modeling philosophy and the same domain viewpoint, adapting the UML MAST format to the OMG proposed rules would be quite simple. Nevertheless we have not adopted it yet, because the OMG profile is at its elaboration phase and thus not yet stable, and because it does not support generic models based on parameters which would restrict some relevant modularity features of UML-MAST.

At present, MAST handles single-processor, multiprocessor, and distributed systems based on different fixed-priority scheduling strategies including preemptive and non-preemptive scheduling, interrupt service routines, sporadic server scheduling, and periodic polling servers. Many of the MAST tools are already available and others are under development. They include offset-based schedulability analysis [4], calculation of blocking times, optimal priority assignment, transaction and system slacks, monoprocessor and distributed simulation, etc. They implement the main schedulability analysis techniques that have been developed in the last decade [5] [6] for fixed-priority systems.

[1] This work has been funded by the *Comisión Interministerial de Ciencia y Tecnología* of the Spanish Government under grants TIC'99-1043-C03-03 and 1FD 1997-1799(TAP)

The UML-MAST models are formulated as proposed by UML revision 1.3 and they are defined by the metamodel [7] that is informally described in this paper. The UML-MAST tools can be used in combination with most of the commercial UML CASE tools. A framework has been implemented for Rational ROSE'2000.

II UML Model of a Real-Time System.

The UML-MAST real-time model describes three complementary aspects of the system:

- The computational capacity of the hardware and software resources that constitute the platform that executes the application.
- The processing requirements and synchronism artifacts that are relevant for evaluating the timing behavior of the execution of the functional operations, and also their blocking sources.
- The workload of activities and the timing requirements they must meet, for every situation in which the system may be executed.

MAST_RT_View is a new proposed view in the UML description of a system that models its real-time behavior according to the UML-MAST methodology. It is a set of UML entities, such as packages, objects, instance classes, links, states, and transitions that are declared in several class and activity diagrams, which collect all the necessary information to perform the schedulability and other analyses using the MAST automatic tools. The set of those components and the concrete values of their attributes constitute the real-time model of the system. The UML-MAST metamodel [6] documents the UML component types and the relations that are valid in the view.

As is shown in Figure 1, three complementary sections constitute the MAST_RT_View:

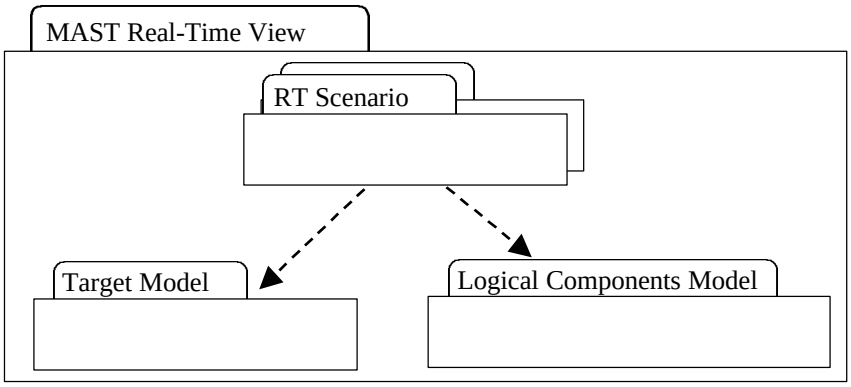

Figure 1: Modules of the MAST_RT_View.

II.1 RT_Target Models.

The **RT target model** describes the processing capacity and the operative constrains of the hardware and software resources that constitute the platform of the system. The processing capacity available to the application is determined by taking into account the processing capacity of the hardware, and all the overheads associated with the different software resources (threads, scheduler, drivers, timers, etc.) running in the background, which must be adequately modeled. Figure 2 shows the abstract classes of the UML-MAST metamodel that are used to model the platform.

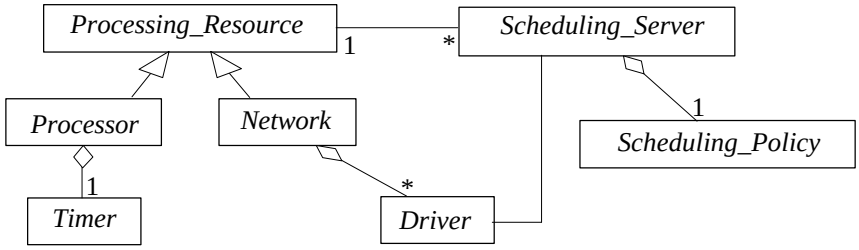

Figure 2: Abstract classes of the platform metamodel.

- ***Processing Resource***: It models the processing capacity of a hardware component that executes some of the modeled system activities. At present, there are two specialized classes defined: the ***Processor*** and the ***Network***. Some of their main attributes are the speed factor, the context switch overheads, the valid priority ranges, etc.
- ***Timer*****:** It describes the specific overhead model of a timing hardware device associated to a processor.
- ***Driver:*** It models the background operations executed in a processor as a consequence of the transmission or reception of a message or part of a message through a network.
- ***Scheduling_Server****:* It describes a schedulable entity in a processing resource, such as a single-threaded process. It has a link to the processing resource on which its activities are scheduled, as well as an entity that describes the scheduling policy.
- ***Scheduling_Policy:*** It defines the scheduling policy of a scheduling server. At present, several fixed-priority scheduling policies are defined, which are compatible within the same processing resource.

II.2 RT Logical Components Model.

The Real-Time Model of the Logical Components describes the timing behavior of the functional operations whose execution times are relevant to satisfy the timing requirements defined in the real-time scenarios of the system. The operations that are modeled in this section can be classified into three types:

- The logical components (methods, procedures and functions) that are defined in the classes' specification of the Logical View of the system. This operation type corresponds to a piece of code that can be executed by some processing resource.
- The physical transference of a message across a communication network, whose timing is relevant to the temporal behavior of the system.
- Predefined tasks which, when invoked by the application, are executed by a co-processor, a hardware device, or a peripheral component, and whose timing is relevant to the temporal behavior of the system.

The model of each logical operation describes the two features that determine its temporal behavior:

- The normalized execution time, which is the amount of processing capacity that is required for executing the operation.
- The list of passive shared resources that must be accessed in a mutually exclusive way.

Figure 3 shows the three abstract classes in the UML-MAST metamodel from which we derive the concrete classes used to model the timing behavior of the logical operations. A **job** models the set

of concurrent activities that are carried out when a logical component (such as a method) is called. A job can be composed of several activities that are scheduled on several threads, and may survive beyond the end of its launching method. The jobs and the composite operations are described by means of activity diagrams.

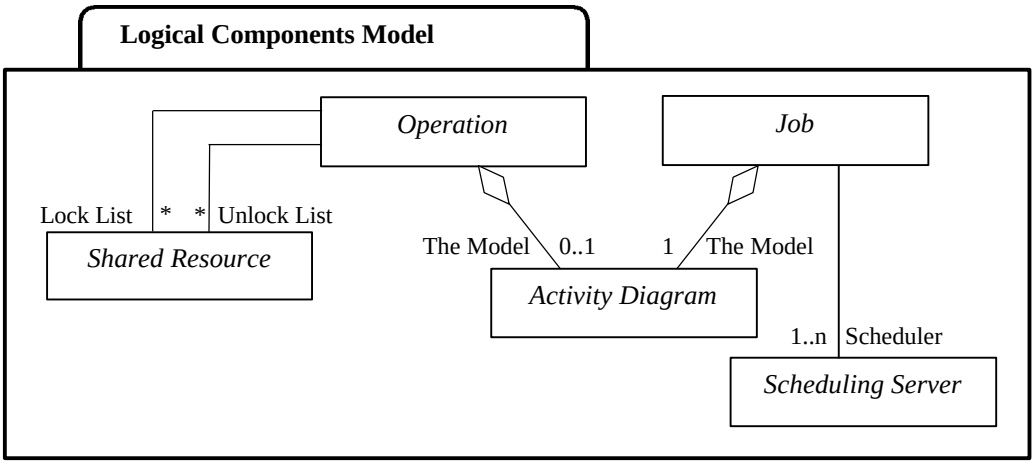

Figure 3: Main abstract classes of the logical component metamodel.

II.3 RT Scenarios Model.

Real-time **Scenarios** are the hardware/software operating modes of the system for which real-time requirements have been defined. Each scenario models a concrete workload of the system in which it is expected to have a specified real-time behavior. They represent the system situations that are object of each real-time analysis.

A real-time scenario is modeled as a set of transactions. Each **Transaction** describes the non-iterative sequence of activities that are triggered by a specific pattern of external events. Besides, it serves as the reference frame for specifying the timing requirements of the scenario. Every transaction contains a description of all the activities that are carried out as a consequence of the arrival of an input external event, and also the ways in which they synchronize (by calling each other, signaling, rendezvous, etc.).

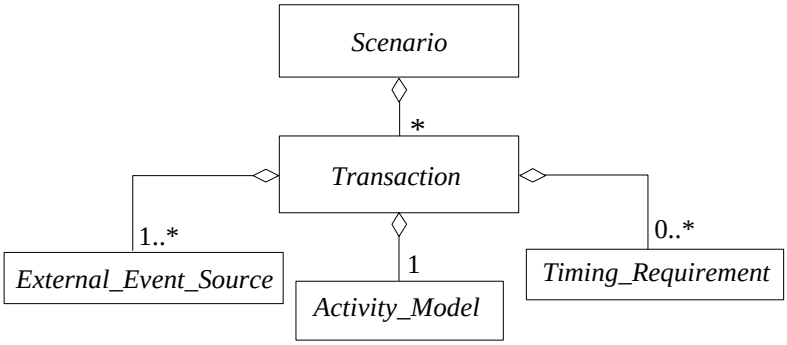

Figure 4: Metamodel of the Scenario class.

Figure 4 shows the metamodel of a scenario, described as an aggregation of the models of its transactions. The model of a transaction is composed of a declaration and a description. A transaction is declared by means of an object that is an instance of the Transaction metamodel class. Each transaction object has two aggregated lists. The first list contains the external event sources that define the transaction's input event pattern caused by the interaction with external components through interrupts, signals, etc. The second one contains all the timing requirements that are imposed on its timing behavior. The transaction description is an activity model that is formulated by one or more activity diagrams.

The timing requirements represent the temporal restrictions imposed on the time at which a particular state of the transaction must be reached. There are different kinds of requirements defined. A Deadline requirement represents a maximum time value allowed for the associated state to be reached. It is expressed as a relative time interval. A Local_Deadline is relative to the beginning of the previous activity, while a Global_Deadline is relative to the external event source that is explicitly linked with it in its declaration. In addition, deadlines may be Hard (they must be always met) or Soft (they must be met only on average). Other types of timing requirements are also defined, such as the Max_Output_Jitter_Req, Maximum_ Miss_Ratio, etc.

III Conclusions.

UML-MAST defines a model capable of describing the timing behavior of many real-time systems, including distributed systems and event-driven systems with complex synchronization schemes. The main characteristic of this methodology is that it allows modeling each relevant aspect of the system with independent real-time modeling components. Therefore it is possible to model each logical operation, the real-time scenarios, etc., independently from the hardware platform and the operating system details. Consequently, UML-MAST makes it easy to evolve and refine the system model according to the evolution of the development process.

The UML-MAST framework for the ROSE environment is available as open source software and it is distributed under the gnu license. It can be found at: `http://ctrpc17.ctr.unican.es/umlmast.`

References:

[1] M. González Harbour, J.J. Gutiérrez, J.C.Palencia and J.M. Drake Moyano: “MAST: Modeling and Analysis Suit for Real-Time Applications” Proceedings of the Euromicro Conference on Real-Time Systems, Delft, The Netherlands, June 2001.

[2] J.M.Drake, M. González Harbour, J.J.Gutiérrez y J.C. Palencia: “Description of the MAST Model”. http://ctrpc17.ctr.unican.es/mast.

[3] B. Selic and A.Moore: “Response to the OMG RFP for Scheduling, Performance and Time: Revised Submission”. OMG document number: ad/2001-06-14.

[4] 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.

[5] 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.

[6] J.W.S. Liu: “Real-Time Systems”. Prentice Hall, 2000.

[7] J.M. Drake y J.L. Medina: “UML-MAST Metamodel”. http://ctrpc17.ctr.unican.es/umlmast.