From composable design models to schedulability analysis with UML and the UML profile for MARTE

Julio L. Medina and Álvaro García Cuesta

Departamento de Electrónica y Computadores, Universidad de Cantabria, 39005-Santander, SPAIN
{ julio.medina , alvaro.garciacuesta }@unican.es

Abstract

Consider the design of hard real-time distributed systems using a model-based and composable approach, in which their specification is made using a high level modelling language like UML. This demo abstract, presents a tool-aided methodology to enable the composition and transformation of such design intended models into also composable schedulability analysis models usable in the verification of the timing properties of the fully composed and loaded systems. In order to annotate the required non-functional properties and state other real-time enabling features, the UML profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE), a recent modeling standard of the OMG, has been used. The methodology comprises several methodological guidelines, specific model transformations, and finally the generation as an output of the concrete schedulability analysis models used by the MAST set of tools, whose results are back annotated into the high level design UML models.

1. Introduction[1]

Model-based software development is one of the most promising software engineering approaches, since using reusable, configurable, and composable models may help significantly in the separation of concerns, increasing the efficiency, but also the quality of software.

In the case of applications with real-time requirements, a model-based methodology can help simplifying the process of building their temporal behaviour analysis models. These models constitute the basis of the real-time design and the schedulability analysis validation processes. With that purpose, the designer of a component must generate, in synchrony with the models used to generate the component's code, an additional parameterizable model, suitable for the timing validation of the system resulting out of the composition. The analysis model for each component abstracts the timing behaviour of all the actions it performs, and includes all the scheduling, synchronization and resources information that is necessary to predict the real-time qualities of the applications in which it might be integrated. In the approach that we present here, these analysis models are to be automatically derived from high level design models annotated with a minimum set of real-time features taken from the requirements of the application in which they are to be used. In analogy to the generation of the application's code as a composition of the code of its constituent components, the analyst, or application designer, can also compose the set of real-time sub-models, and build the complete real-time analysis model of the application. This strategy helps the designer to get rid of the tedious and error prone task of building in one piece the complete reactive model of the application.

A discussion of the process followed for the design of the real-time characteristics in an strict component-based development methodology may by found in [1]. This brief demonstration abstract concentrates on (1) the rational for the approach, (2) the generation of the output MAST analysis models [2] and the technologies used for it, and (3) guidelines for the construction of high level application design UML models that can be transformed in an automated way in the respective schedulability analysis models. Finally some conclusions and future work.

2. The approach

The UML Profile for MARTE [3] brings a large number of modeling constructs and concepts that may be used for realizing schedulability analysis in a variety of ways. This effort presents a comprehensive abstract of the way this approach intends to use those modeling constructs in order to enable (1) early V&V and afterwards (2) the iterative use of the models created.

In order to cope with complexity, to manage the risks associated to the research and the development of tools efforts, and also to make better use of the modeling resources offered by MARTE, the complete problem is divided in two challenging but achievable steps.

1. This work has been funded by the European Union under contracts, FP7/NoE/214373 (ArtistDesign), and FP7/CSA/224330 (ADAMS); and by the Spanish Government under grant TSI-020400-2009-108 (ITEA2-EVOLVE). This work reflects only the author's views; the EU is not liable for any use that may be made of the information contained herein.

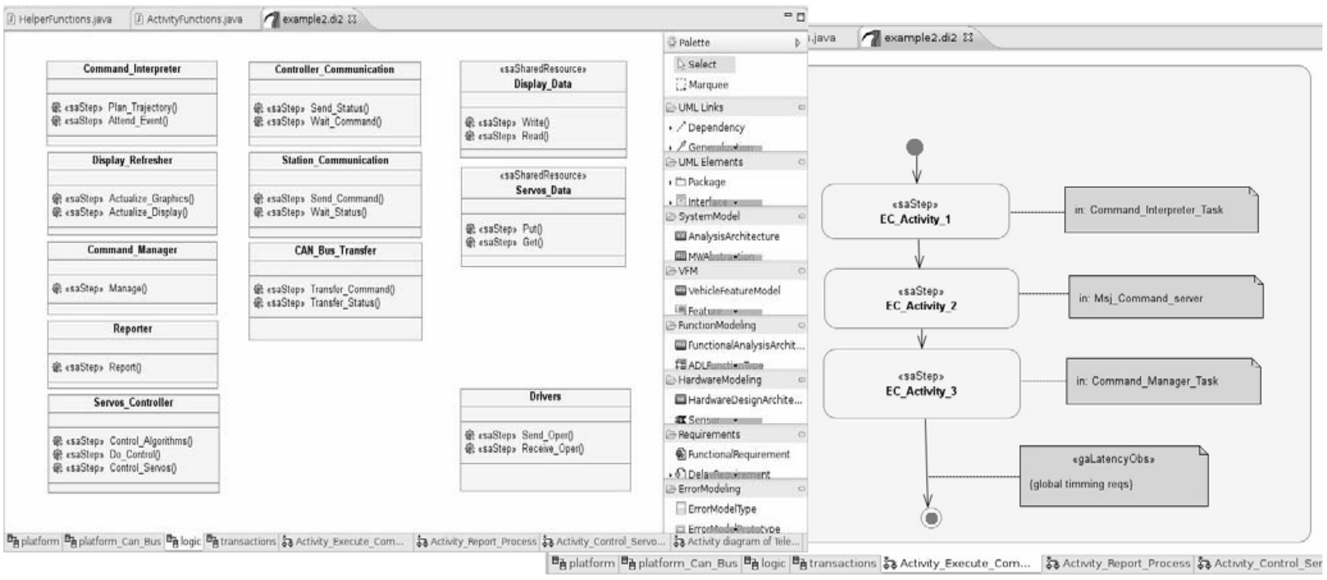

Fig. 1. Views of the tool dealing with structural and behavioral modeling for schedulability analysis

The first step (a) comprises the definition and manipulation of what we will denominate the "analysis models". The second one (b) is the specification and automation of those modeling constructs related to what we will call the "design models". This method helps to support the design of applications in terms of composable parts, which are closer in granularity to the concept of real-time objects than to the fully CBSE interpretation of components. In a fully component-based approach, the creation of the analysis models would have to be made as a combination of both, structural elements plus their deployment. In a model-driven approach, this later strong form of composability is in a higer level of abstraction, but still may benefit of the approach here described in order to assess a variety of non-functional properties, in our case of course its timing properties by neans of schedulability analysis.

3. Analysis models

The first and more important problem has been the definition/selection of which and how elements in MARTE are to be used in the creation of Schedulability analysis models. These elements are the basis for the tool that has been developed for the generation of MAST analysis models taken from UML+MARTE annotated analysis models. Following previous research efforts [4], MARTE provides concepts to structure the analysis models using three main categories: The platform resources (a), the elements describing the logical behaviour of the system constituent parts (b), and finally the real-time situations to be analysed (c). Though the precise mapping from MARTE to MAST elements may be seen in the demo session, here we summarize a condensed view of the MARTE elements pro-

Table 1

Platform Resources	Behavioral Models	Real-Time Situations
GaResourcesPlatform SaExecHost * SaCommHost * SaSharedResource * SchedulableResource *	GaWorkloadBehavior GaScenario SaStep * SaCommStep	SaAnalysisContext * GaWorkloadEvent * Allocate Allocated Assign SaEndToEndFlow * SaSchedObs GaLatencyObs *
* Elements used in the extraction tool in current version.		

In the tool that has been provided for the generation of MAST models from UML+MARTE models, the platform elements are modelled as a set of structural elements with stereotypes annotated on them. Figure 1 shows an example of the usage of these elements in the modeling of a teleoperated robot distributed platform. The end-to-end flows that described scenarios are modeled using sequence charts or activity diagrams. In summary, in the current version of this tool the elements taken from MARTE to generate the MAST analysis models are:

Processing_Resource <= SaExecHost, SaCommHost

Scheduler <= SaExecHost, SaCommHost

Scheduling Servers <= SchedulableResource

Shared_Resource <= SaSharedResource

Operations <= SaStep <= Sequence/activity Diagram plus subUsages (ordered list of called operations)

Transactions <= Sequence/Activity Diagram + GaWorkloadEvent + GaLatencyObs

This effort has been realized using the technologies provided by PapyrusUML as graphical tool, the UML2 plugin as model repository, and the Acceleo plugin for the extraction of text from the UML2 models plus a significant number of Java functions. The code used as well as the scripts created will be shared as open source. An initial version with support for activity diagrams and composition of independently characterized timed behaviours will be demonstrated.

4. Design models

The second challenge is the definition of the elements in UML+MARTE to be used in early V&V design models in such a way that they can be used for the double purpose of constructing development (implementation) oriented models or even code directly while at the same time their respective analysis models may be generated through simple and as much as possible automated model transformation mechanisms.

For this purpose the natural candidates in MARTE are the fundamental modeling constructs described in the HLAM (High Level Application Modeling) chapter: RtUnit and PpUnit.

The RtUnit modeling element is the basic building block for handling concurrency in the design and analysis of real-time applications. The PpUnit is the modeling element used for specifying mutual exclusion between concurrent units and the adequate protection protocols in the access to passive shared resources, for avoiding unbounded priority inversion.

The key for the usage of these elements is the enabling of simple mechanisms to keep in synch the two specialized views that are elaborated as transformations from the design models built with them: the code generation+implementation, and the corresponding schedulability or even performance analysis models. In order to get this we propose a methodology founded in a small number of modeling rules for the usage of RtUnits and PpUnits, and directions for the generation of the subsequent implementation and analysis models.

In order to accomplish the objective of setting up the basis for an iterative development process, the driving forces for the definition of the methodology have been the easiness to iterate over modeling intents and a design space exploration strategy to introduce analysis results back in the design constraints.

For the purpose of this methodology we will consider all the requirements as applicable to a generic unit of design called module. A module in this sense represents a fraction of the system that is to be mapped to the equivalent abstraction/encapsulation element used on the concrete target design methodology applied by the industrial practitioners for coping with complexity in the field. This results natural when considering them as independent subsystems, but it is applicable also to other composition mechanisms like loosely coupled software/hardware components, or physical concurrent units

The modeling rules to be applied are the basis for the combined purpose of a design & analysis methodology and are later complemented with guidelines for specific phases and concerns.

The description of the RtUnits and PpUnits and their precise semantics are made in the domain view of HLAM chapter and the appendix F of MARTE [3] respectively. The set of rules is enounced considering the semantics there described but using the nomenclature of the attributes available in the corresponding stereotypes.

Early V&V assumes that at the time of analysis there are still a number of decisions not taken about aspects like the platforms or specific interface technologies. To be able to asses the viability of the system without this information, some by default values will be filled in the analysis & design models.

The set of rules for the use of UML with the HLAM modeling elements of MARTE needs to restrict the design space to get models that may be analysed by schedulability analysis with the available techniques. This way it formulates the basis for modeling at any stage of the development process. This initial set of rules is enounced as follows:

1. Real concurrency is handled by RtUnits at processing resource level, each node by them represented may in turn handle several schedulable resources by means of a regular scheduler.
2. Each RtUnit may have up to one schedulable resource on it, and all its behaviours, which may be called from other RtUnits, run under the scheduling parameters associated to that schedulable resource. In case the RtUnit has no schedulable resource, its behaviours run under the scheduling parameters of the calling RtUnit.
3. All the RtUnits deployed in a processing resource are handled by the same scheduler and use the same (or fully compatible) scheduling policy.
4. Each RtUnit whose isMain attribute is set to true, implies the presence of an execution host where the main service of the RtUnit is deployed.
5. The attribute srPoolPolicy holds the value infiniteWait
6. ExecKind of PpUnits is ImmediatRemote
7. All services use the same priority scheme: ImmediateCeiling or PriorityInheritance
8. The ConcurrencyPolicy of PpUnit is Guarded. [The concurrency policy of the kind Concurrent might be enabled in order to have the writer/reader ConcurrencyKind available but this behaviour requires additional capa-

bilities from the analysis techniques so in principle it is discouraged].

9. Behaviours of RtUnits stereotyped as RtServices are those that may be called from others.

Additional rules that apply in specific phases of the development process are:

11. The platform models of the execution hosts are derived from the RtFeatures of RtUnits with the attribute isMain set to true. The basic assumption is that a main is the starting of the full piece of software running on a concrete node. The scheduling policy of the scheduler derives from the one used for this main. Consistently the range of priorities (in the case in which this is the policy chosen) will be set to be greater than the number of RtUnits (with their isMain attribute to false) with which the main RtUnit has any sort of interaction.

12. The rules for analysis platform models will be refined after practising with the MAST default values. (using initially no context switch time for example).

13. The links between services will be used to define the steps in the end-to-end-flows.

14. The parameters of the Analysis Context modeling element will be used to define the variations in the analysis due to refinements in the design.

15. Results will be place back in design models by means of RtFeatures Specifications and the parameters of AnalysisContexts.

16. The iterative nature of the models used for design space exploration will be handled by specializing/using the configuration stereotype, described in the Modal Behaviour section of HLAM chapter of MARTE.

The tooling support for enforcing and helping to asses the usage of these rules is not already embedded in the version that will be demonstrated, but it is on its way to be realized

5. Conclusions and future work.

Considering the prospects of the OMG´s UML Profile for MARTE as a modelling standard for analysis tools interoperability, it seems reasonable to look for model based strategies that link it with modeling intensive activities. And a clear semantics for the High level application modeling is the basis for automating the process of having timing analysis results quickly in the development life cycle.

The extraction of MAST analysis models from the UML+MARTE schedulability analysis specific models is a first demonstrable step in the direction pointed out by this effort and comprises the construction of analysis models from separated composable modeling descriptions using the specific constructs brought by the SAM chapter of MARTE, which is consistent with MAST and previous efforts in this direction [4]

From the real-time and embedded systems research community perspective, this effort constitutes a step to get the effective exploitation of the capabilities of the available analysis and verification techniques, which despite the efforts in dissemination, have not yet reached an audience large enough to reward the many years of work in the field.

The modelling strategy and tools proposed in this work are just a first step in this direction, a significant work remains to be done in order to have a fully automated process. The validation of the rules and their automation by means of a model validator and the necessary transformations are part of our ongoing work and will be addressed in the near future.

References

[1] López P., Drake J.M., and Medina J.L., Enabling Model-Driven Schedulability Analysis in the Development of Distributed Component-Based Real-Time Applications. In Proceedings of 35th Euromicro Conference on Software Engineering and Advanced Applications, Component-based Software Engineering Track, Patras, Greece, August 2009, IEEE, ISBN 978-0-7695-3784-9, pp. 109-112.

[2] M. González Harbour, J.J. Gutiérrez, J.C.Palencia and J.M.Drake, MAST: Modeling and Analysis Suite for Real-Time Applications, in Proc. of the *Euromicro Conference on Real-Time Systems*, June 2001.

[3] Object Management Group, UML Profile for Modeling and Analysis of Real-Time and Embedded systems (MARTE) version 1.0, OMG doc. formal/2009-11-02, 2009

[4] J.L.Medina, M.González Harbour and J.M. Drake, Mast Real-Time: A Graphic UML Tool for Modeling Object-Oriented Real-Time Systems, in Proc of the *22nd IEEE Real-Time System Symposium (RTSS 2001)*,pp 245-256, 2001.