

SoC/SoPC development using MDD and MARTE profile

Denis Aulagnier, Ali Koudri, Stéphane Lecomte, Philippe Soulard, Joël Champeau, Jorgiano Vidal, Gilles Perrouin, Pierre Leray

▶ **To cite this version:**

Denis Aulagnier, Ali Koudri, Stéphane Lecomte, Philippe Soulard, Joël Champeau, et al.. SoC/SoPC development using MDD and MARTE profile. Babau, Jean-Philippe and Blay-Fornarino, Mireille and Champeau, Joël and Gèrard, Sèbastien and Robert, Sylvain and Sabetta, Antonino. Model Driven Engineering for Distributed Real-time Embedded Systems, ISTE, 2009. inria-00468650

HAL Id: inria-00468650
https://inria.hal.science/inria-00468650

Submitted on 31 Mar 2010

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

SoC/SoPC development using MDD and MARTE profile

Denis Aulagnier[1] — **Ali Koudri**[1] — **Stéphane Lecomte**[2] — **Philippe Soulard**[3] — **Joël Champeau**[4] — **Jorgiano Vidal**[5] — **Gilles Perrouin**[6] — **Pierre Leray**[7]

[1]*Thales Aerospace Division*
10, avenue de la 1ère DFL
29283 Brest - France
{denis.aulagnier, ali.koudri}@fr.thalesgroup.com

[2]*Thomson R&D France*
Corporate Research - Networking Lab.
1, Avenue de Belle Fontaine
35576 Cesson-Sévigné Cedex - France
stephane.lecomte@thomson.net

[3]*Sodius*
6, rue de Cornouaille
44300 Nantes - France
psoulard@sodius.com

[4]*ENSIETA*
2, rue François Verny
29806 Brest Cedex 9 - France
joel.champeau@ensieta.fr

[5]*Université de Bretagne Sud*
Lab-STICC
56321 Lorient Cedex - France
jorgiano.vidal@univ-ubs.fr

[6]*INRIA*
Campus de Beaulieu
F-35042 Rennes - France
gilles.perrouin@irisa.fr

[7]*Supélec*
Avenue de la Boulaie B.P. 81127
35511 Cesson-Sévigné Cedex - France
pierre.leray@rennes.supelec.fr

ABSTRACT. *This paper presents a new methodology to develop SoC/SoPC applications. This methodology is based on UML and MDD and capitalizes the achievements of "Electronic System Level" community by taking into account the new MARTE profile dedicated to real-time embedded systems. In the MOPCOM SoC/SoPC research project, a tooling has been developed to support this SoC/SoPC methodology, the MARTE profile, HDL code generation and documentation generation. A Cognitive Radio demonstrator is presented to illustrate the methodology and the tooling.*

KEYWORDS: *UML, MDD, MDA, MARTE, ESL, SoC/SoPC, FPGA*

1. Introduction

Thanks to the ever-increasing performance of digital electronics, an entire embedded System can now be integrated on a single Chip: i.e. a SoC – *System on Chip* – or a SoPC – *System on Programmable Components* – for FPGA – *Field Programmable Gate Array* – reconfigurable components.

In parallel, to catch up with this components complexity, a dramatic enhancement of hardware design productivity is required to avoid a "productivity gap" [ITR 07]. ESL – *Electronic System Level* – tools have emerged in order to tackle this issue by improving the level of abstraction of hardware developments. For example, some ESL tooling, enable to simulate a design at TLM – *Transaction Level Modeling* – with SystemC language or to synthesize hardware architecture directly from C functional code rather than from a RTL – *Register Transfer Level* – description.

Besides ESL modeling approaches, UML – *Unified Modeling Language* – [OMG 06b] originally dedicated to Software development has extended its scope to System or real time embedded application development through SysML – *System Modeling Language* – [OMG 08a] and MARTE – *Modeling and Analysis of Real-Time Embedded systems* – [OMG 07] profiles.

Moreover, MDA – *Model Driven Architecture* – [OMG 03], promotes a development methodology based on models transformations at several levels of abstraction and that follows the well known Y-Chart co-design approach: at each level, a PIM – *Platform Independent Model* – representing the application is mapped onto a PM – *Platform Model* – representing the target architecture to obtain a PSM – *Platform Specific Model* – representing the implementation.

As the development of SoC/SoPC components covers system, software and hardware engineering activities, from the system requirement capture, up to the fine analysis of the hardware logic timing, a SoC/SoPC development methodology should take advantage of these new UML profiles and MDA methodology in capitalizing, as well, the achievements of the ESL community.

An experimentation of this approach has been carried out in the frame of the MOPCOM SoC/SoPC project [MOP 07, KOU 08] and is presented in this paper. Section 2 is a state-of-art overview. The following sections present the different types of models identified in MOPCOM SoC/SoPC development methodology and the MARTE profile elements used at each level. An example of Cognitive Radio application is used to illustrate this process in section 4. Finally, sections 9 and 10 describe MOPCOM tooling developed to support the design process and the generation of – HDL – *Hardware Design Language* – as well as the documentation automatic generation.

2. Related Works

As explained above, developments of SoC/SoPC and RTES – *Real Time Embedded Systems* – in general is related to the ESL community. Only a reliable methodology,

based on appropriate languages and tools can help to handle market pressure (time-to-market, competitiveness), increasing evolution of the technology or standards (DO-178B, DO-254, etc.) [GER 03] related to such developments.

In order to address the market constraints and obsolescence issues, separation of concerns is needed to allow the concurrent development of applications and execution platforms. This kind of approach have been first proposed in the Gajski and Kuhn Y-Chart model [GAJ 83], generalized by the Model Driven Development approach and standardized by the Model Driven Architecture OMG's standard . Moreover, in order to allow faster design space exploration, system under study must be modeled and validated at several levels of abstraction [SAN 04]. There is no real consensus about the number of required abstraction levels even if there are some efforts to define and standardize abstraction levels and their related services and interfaces [KAS 07].

Several languages enable the description of behavioral or structural parts and the allocation of the system under development. The most important factors influencing the choice of a language in a modeling or analysis activity are its expressiveness power and its tooling. For instance, SystemC [OSC 05] is a language allowing functional and platform modeling at several levels of abstraction, and is supported by several free or commercial tools dedicated to analysis or compilation / synthesis.

In addition to separation of concerns and definition of levels of abstraction, there is a need to favor reusability in order to improve the productivity. Indeed, large systems development has to rely on libraries of proved and well-documented IPs at each level of abstraction. Interconnection and exchange between IPs is based on the use of standard interfaces and protocols. In some cases, Ad-doc IPs can be wrapped to conform to standards [KOU 05].

Developments of RTES include modeling activities, using languages based on either grammars or metamodels, as well as analysis activities such as formal validation or simulation. The main issues when modeling RTES are the description of concurrency / communication [GER 02], execution platform, possibly represented at several levels of abstraction, and QoS – *Quality of Service*. Modeling and Analysis activities must be replaced in the context of a well-defined methodology. For that, there are two different approaches:

– use several DSL – *Domain Specific Language* – fitting for each modeling or analysis activity,

– use a general purpose modeling language, such as UML, with dedicated profiles to support the required concepts.

Additional mechanisms such as annotations are also required in order to add relevant information needed by analysis tools (example: resource usage for schedulability analysis).

Based on the use of selected formalisms, several methodologies and tools have been developed to support RTES development. Few examples are given below.

The methodology MCSE – *Méthodologie de Conception des Systèmes Electroniques* – [CAL 08] proposed by the University of Nantes, enables design space exploration through the use of the SystemC TLM language. The French company Cofluent Design [COF], in partnership with MCSE Team, has developed the ESL design tool "Cofluent Studio", based on Eclipse environment, which supports MCSE methodology.

The Ptolemy environment from UC Berkeley [BUC 02] allows description of systems mixing several MoC – *Models of Computation* – through the notions of actors and directors. A director defines a domain of execution for its actors enabling the mixing of several models of computation in the same model. This is an important issue because real-time systems usually mix analog and digital devices and possibly several time domains.

Syndex from INRIA is a tool implementing the Architecture Adequation Algorithm [KAO 04] addressing the allocation issue.

In the context of the UML, several profiles have been proposed to extend UML capabilities in order to handle modeling and analysis of RTES or SoC. Among them, we can cite:

– the SPT – *Schedulability Performance Time* – OMG's profile [OMG 05a] which was based on the version 1.4 of UML and completed by the QoS and Fault Tolerance OMG's profile [OMG 08b] for the non-functional aspects,

– the UML4SOC OMG's profile is dedicated to describe SoC [OMG 05b],

– the UML for SystemC profile proposed in [RIC 05] gathers the capabilities of UML and SystemC,

– the UML for MARTE OMG's profile [OMG 07] can be viewed as an improvement of the SPT profile (cf. section 3),

– the Gaspard profile is specific to parallel and distributed computing applications implemented into SoC [PIE 08].

Based on the use of UML profiles, examples of RTES or SoC design environments are given below.

The *ACCORD/UML* methodology [GER 02] aims at using UML concepts to design RTES. It was originally based on the use of the SPT profile and now relies on MARTE profile. It is supported by the eclipse-based PAPYRUS tool [CEA].

The University of Milan, in collaboration with STMicroelectronics, proposes a development process for embedded systems and SoC called UPES – *Unified Process for Embedded Systems* – , based on UML for SystemC profile [RIC 07].

The Gaspard Methodology [PIE 08] is intended to provide a framework for developing parallel and distributed applications implemented on SoC. This methodology is an implementation of the MDA approach in the eclipse framework and provides a set of transformation rules allowing generation of optimized SystemC Code for repetitive structure architecture.

3. MOPCOM Process and Models

Our SoC/SoPC development process is based on a MDD approach. It relies on two elements: a conventional System Engineering methodology for the System Analysis, including Requirement Capture and Functional Analysis activities, and a dedicated process for the SoC/SoPC architecture definition. This process emphasizes:

– Separation of concerns: as explained in the previous sections, separation of the application (PIM) from the platform (PM) and description of the mapping to generate the implementation (PSM), as in a classical Y-chart co-design pattern,

– Analysis Driven Development: the previous pattern can be reproduced with several execution platform abstractions, depending on the kind and the number of analysis one needs to perform. For instance, in the MOPCOM methodology, we have define three abstraction levels for the target platform as explained below.

The set of input / output models of our process is presented in the figure 1.

– AML – *Abstract Modeling Level* – is intended to describe a virtual execution platform where expected concurrency, pipeline and communication policy are explicitly defined. The allocation model describes the mapping of the application onto the Model of Computation defined by the AML platform.

– EML – *Executable Modeling Level* – provides a coarse definition of the physical platform. At this stage, the platform is composed by generic components such as Processing or Memory Units. Allocation model describes the mapping between the previously generated model onto the platform. This results in a topology description allowing performance or schedulability analysis.

– DML – *Detailed Modeling Level* – refines the definition of the platform with the necessary information for Software and RTL HDL code generation. The Allocation consists in mapping the previous PSM AML model onto a more precise description of the platform.

Besides automatic code generation, other artifacts, such as documentation, can be produced automatically.

Each stage is validated against simulations. The main benefits of such approach are the minimization of the risks as emphasized in a classical iterative development process [BOE 88] and the optimization of the development time, through concurrent developments.

Another goal of our work is to evaluate the relevance of the UML MARTE profile in term of concepts: the MARTE profile encloses a large set of concepts to model and analyze Real Time Embedded Systems. Those concepts are organized in several hierarchical sub-profile packages, following concerns they are related to.

For instance, the design of RTES deals with the modeling of properties that quantify offered and provided services by a resource to the application. In this purpose, there is a package dedicated to the description of NFP – *Non Functional Properties* –. The NFP constraints are applied all along the process from the requirements capture to

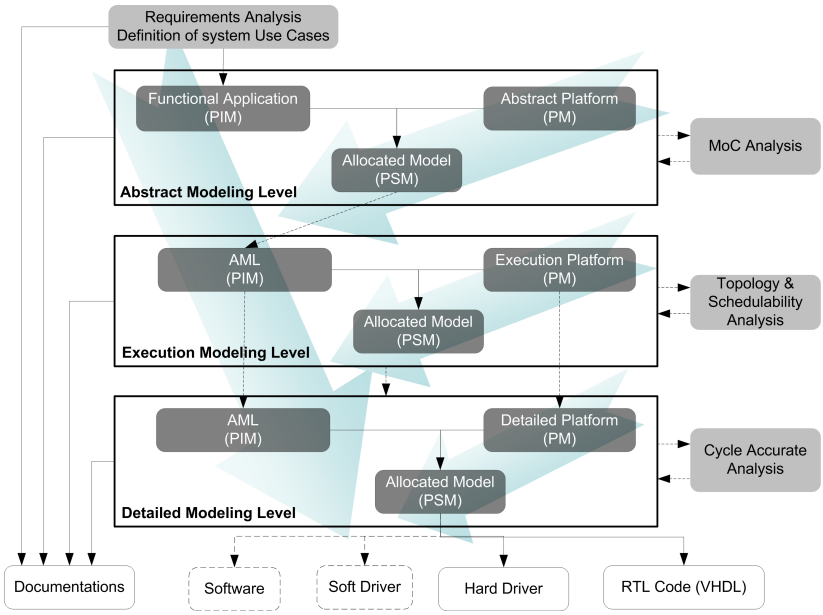


Figure 1. *MOPCOM abstraction levels and included models*

the detailed platform modeling. Properties and constraints can be expressed using the VSL – *Value Specification Language* – which can be viewed as an extension of OCL – *Object Constraint Language* – [OMG 06a] taking into account QoS characteristics.

Time models (causal, logical or continuous) and time access (logical or chronometric clocks) are also important issues when modeling RTES. Concepts related to time have been introduced into the Time Package and refine the UML time model. The clocked or synchronous time abstraction divides the time scale in a discrete succession of instants while the physical time can be viewed as dense discrete time, which is useful to model analogical devices through ODE – *Ordinary Differential Equation* –.

Analysis activities deals mainly with qualitative or quantitative features of the system, such as period, occurrence kind, duration, jitter, etc. They are part of the HLAM – *High-Level Application Modeling* – package.

Analysis of schedulability or performance of a system is allowed in MARTE using the SAM – *Schedulability Analysis Modeling* – and PAM – *Performance Analysis Modeling* – packages through the concepts of workload, resource acquisition and release and so on. MARTE allows by default two kinds of analysis: performance and schedulability but it provides generic concepts in order to allow one to define other kinds of analysis, such as WCET – *Worst Case Execution Time* –.

Platforms can be described at several levels of abstraction with more or less details, depending on the kind and the number of analysis to be performed (cf. above).

The GRM package – *General Resource Modeling* – provides concepts allowing the modeling of the platform at a high abstraction level while those concepts are refined in the SRM and HRM packages. The SRM package – *Software Resource Modeling* – provides details related to the description of software platform such as RTOS or middleware. The HRM package – *hardware resource modeling* – provides concepts related to the description of physical platform such as Buses, FPGA, Processor, etc.

The MOPCOM methodology provides the rational to use all those MARTE concepts in a consistent manner. Indeed to enforce the definition of our methodology, we select some MARTE elements for each abstraction level in our SoC/SoPC development process. We have defined the usage scope of those concepts by adding constraints to the metatype of the UML/MARTE language. Those set of constraints specialize the language (UML/MARTE) to the SoC/SoPC domain and are capitalized into the model of the process itself.

In the next sections, we present each level of our MDD process with its associated set of MARTE concepts.

4. Application

The development process presented above has been experimented through a CRS – *Cognitive Radio System* –. A CRS is a radiocommunication equipment which is able to adapt its functionality according to the Radio environment [MIT 01, HAC 07]. A CRS for instance can identify the RAT – *Radio Access Technologies* – available and determines their characteristics such as bandwidth and load as well as environment characteristics such as localization. The more suitable available RAT in the area is then selected and the CRS receiver chain is configured to communicate through the corresponding protocol.

This type of system is quite complex and for the demonstrator only two Use Cases are analyzed:

– the "Locate RAT Source" Use Case developed by Thales with Supelec localizes the Direction Of Arrival of the available RAT. The RAT identification is carried out after a Spectrum analysis of the RF environment and a blind standard recognition of the communication protocol. The detected radiocommunication signals are recognized by analyzing some discriminating parameters that characterize the protocol such as: channel bandwidth, frequency hoping, single/multicarrier signal, etc. The localization is carried out with a beamformer and removes the eventual multipaths to only keep the Direct Line-Of-Sight direction.

– the "Wireless transmission" Use Case developed by Thomson Corporate Research [LEB 08] implements a wireless stack based on standard 802.16 and uses the TDMA – *Time Division Multiple Access* – MIMO/OFDM – *Orthogonal Frequency Division Multiplexing* – technologies. For the demonstrator, the Use Case focuses on baseband processing and more specifically on the algorithms and the architectures of MIMO – *Multiple Input Multiple Output* – decoding.

The Cognitive Radio System is made up of 4 antennas, a baseband processing, and an Ethernet connection. The targeted platform for the implementation is a reconfigurable component that can demonstrate self-reconfiguration.

5. System Analysis

The first activity of our design process is the System Analysis. This activity is not specific to SoC/SoPC development and can rely on an existing System Engineering methodology such as the Telelogic Harmony/SE^{TM} [ART 08] methodology. Harmony/SE^{TM} is a SySML based methodology that consists mainly in two steps: Requirements Capture and Functional Analysis to which we add MARTE improvements, essentially related to real-time features.

5.1. *Requirement Analysis*

A Requirements Analysis captures system functional and non-functional requirements and merges them into Use-Cases. Use-cases define services provided by the system to external entities (actors).

The operational contracts (scenarios) and the interfaces between the actors and the system are formalized at high abstraction level using for example textual descriptions or models such as Sequence Diagrams or Activity Diagrams.

At this stage, we use the stereotypes of the packages NFP, VSL, Time of MARTE. Figure 2 is an example of a Sequence Diagram related the Use Case "Locate RAT Source" that relies on ≪TimedInstantObservation≫ and ≪TimedConstraint≫ stereotypes. The TimedConstraint, specifying a duration constraint, is expressed using the VSL syntax.

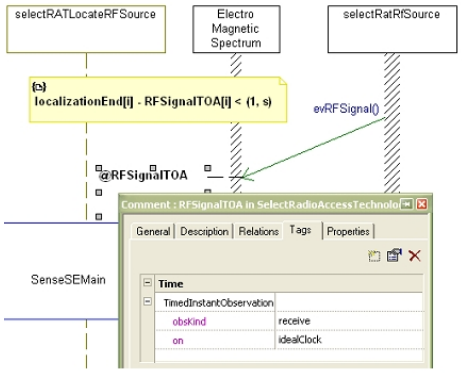


Figure 2. *Sequence Diagram of UC "LocateRAT Source"*

5.2. *Functional Analysis*

Compared to the requirements Analysis, the functional analysis focuses on the functional decomposition of the Use Cases. Use Cases are split into functions. This functional decomposition is captured through Class and Object Diagrams whereas the behavior of the Use Cases is defined with Activity, Statechart and Sequence Diagrams refining the internal interactions between the different functions.

At this stage, we use the same MARTE elements that in the previous one in order to precise for example derived constraints and internal data types. Nevertheless specific Functional Analysis rules have been applied to ease allocation onto the platform in the next steps of the process through orthogonalization of specification and implementation and the use of some Design Patterns [GAM 95]:

– "Facade": to unify configuration interfaces,

– "Decorator": to separate concerns simulation and system behavior,

– "Singleton": to share single object,

– "Strategy": to model dynamic reconfiguration.

Figure 3 shows an example of the functional decomposition for the UC "Locate RF Source".

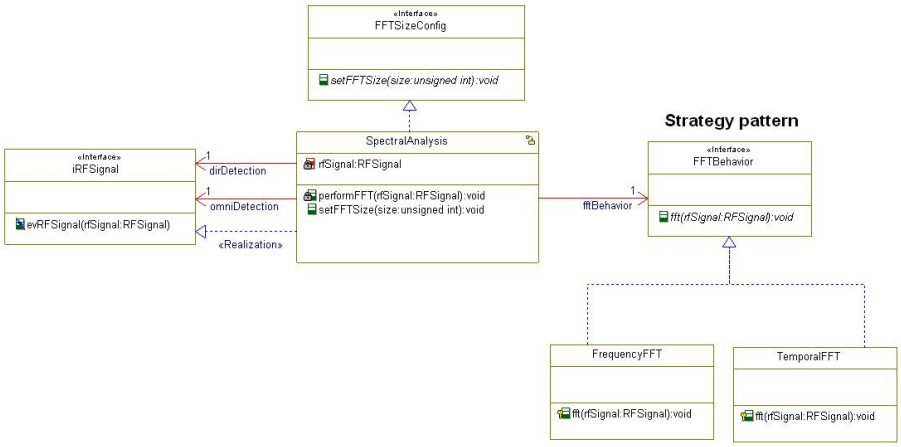


Figure 3. *Class Diagram related to UC "Locate RF Source"*

5.3. *Action language*

To perform validation, Requirement Analysis and Functional Analysis, models must be executable. That requires an action language for low-level expressions that complements the high-level UML semantic and diagrams, and to specify operation

bodies, trigger/guard/action on transition and in states, and data declarations. The selection of the most suitable action language action language raises questions about textual or graphical notation, and general versus HDL-specific language that should be accessible to system, software and hardware designers without to many problems related to its learning curve.

After analysis, C++ language turned out to be the most convenient choice. Indeed, It is a wide-spread and standard object-oriented language, supported by many high-performance development environments (including Rhapsody-in-C++ suite).

Only a C++ subset is used in the models (along with some macros for event and port handling). C language is fully mastered by hardware designers (similar syntax for non-OO subset) and next-generation SystemC language is basically a C++ library dedicated to hardware applications.

6. Abstract Modeling Level

While the aim of the functional analysis was the definition of the behavior of the system and its functional breakdown into functional blocks, the aim of this level is to identify the needed level of concurrency and define the way concurrent blocks communicate. The underlying goal of those identifications is analysis that can be performed like consistency or deadlocks analysis.

The notion of concurrency in UML is supported by the "isActive" meta-attribute meaning that corresponding classes manage their own thread of execution, but it does not say so much about what kind of interactions can occur between active classes. Still, there are missing information to make relevant analysis needed at this level. Fortunately, the MARTE profile completes the definition introducing the concepts of RTUnits or RTeConnectors in the HLAM package.

The RTUnit is a refinement of the Block concept introduced in SysML with additional real-time features. An RTUnit provides and requires an set of real-time services. In order to realize those services, a RTUnit owns a set of real-time behaviors with bounded or unbounded message queue for each of those behaviors. An RTUnit can also own a set of schedulable resources which are typed by other RTUnits, connected through RTeConnectors, allowing hierarchical description of the system. Then, the owning RTUnit provides an execution context (domain) for each of these sub-RTUnits and is responsible for managing their interactions and concurrency.

Since the AML model provides an execution framework for the system under study, it can be considered as the highest abstraction of the execution platform. Indeed, its identification constitutes the first step of the design space exploration.

At this level, every concurrent unit is stereotyped «RTUnit» characterized by its provided / required services set and its real-time behaviors. RTUnits communicate through real-time connectors «RTeConnector». Actions performed by an RTUnit are

stereotyped «RTAction». For each of those concepts, a quantitative or qualitative aspect information is provided: duration, priority, occurrence kind, etc.

The AML platform aggregates the set of needed functional objects to implement the system with additional information about concurrency and communication. To generate the real-time units, one needs to provide the functional design and allocation with associated constraints. Figure 4 describes the process of allocation while figure 5 gives, in MQL language (cf. section 9), an excerpt of the transformation rules that have been applied to generate the allocated platform.

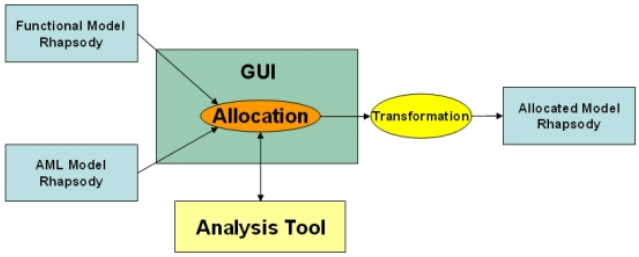


Figure 4. *Process of Allocation*

```
//references to the needed stereotypes in the target model
var clSt : rhapsody.Stereotype = target.getInstances("Stereotype").detect("name", "RTUnit");
var opSt : rhapsody.Stereotype = target.getInstances("Stereotype").detect("name", "RTService");
//Iterate over instances of the collaboration
foreach (i : mopcom.Instance in mcPack.getInstances("Instance")) {
    //detect corresponding RTUnit definition in the target model
    cl = target.getInstances("Class").detect("name", i.type.name);
    //if it does not exist, create it and add it in the target model
    if (cl == null)
    {
        rtu = i.type;
        cl = target.create("Class");
        cl.name = rtu.name;
        rtu#class.add(cl);
        //add "RTUnit" Stereotype to the generated class
        cl.stereotypes.add(clSt);
        cl.stereotype = clSt;
        //add the "RTUnit" in its owning package
        cl.owner = rhPack;
        rhPack.classes.add(cl);
    }
}
```

Figure 5. *Excerpt of Allocation to Allocated Model Transformation in MQL Syntax*

Functional blocks are turn into AML blocks stereotyped ≪RTUnit≫ and ≪Allocated≫ in purpose of traceability. Connectors binding ports implements communications related to the MoC indicated on the allocation constraints of the link. Special blocks are inferred for (de)multiplexing data between RTUnits when several functional blocks are executed sequentially.

7. Execution Modeling Level

The MOPCOM Execution Modeling Level (EML) is made up of three different models (Figure 1). The main goal of this level is to model the topology of the virtual hardware platform and to analyze the system scheduling.

7.1. *The Platform Independent Model/Application Model in EML*

The PIM model in EML is similar to the PSM model in AML with a refactoring if necessary. We can transform the functional structure in order to allocate the PIM onto PM. In fact, if the analysis results do not respect the specifications, the functional structure or/and the topology of the virtual hardware platform must be changed.

7.2. *The Platform Model in EML*

In EML, PM only represents the topology of the virtual hardware platform based on high-level generic components. Indeed, the objective of the virtual platform is to hide the physical platform to the application. This PM cuts itself of superfluous details such as the protocols description, the type of computing resources and storage resources used in the physical platform model. The first interest of such a modeling is to represent the nodes of computation, of storage, of communication and the services offered by the platform to the application. A natural modeling concept of PM in EML is a transactional-level modeling, as promoted by Gajski and SystemC community in general. Thus the communications between the components of the platform are represented by calls of functions and not by a detailed modeling of the protocol and the connectivity which are represented in the RTL level. The MOPCOM methodological tool at this level is MARTE GRM – *Generic Resource Modeling* – subprofile. Figure 6 shows an example of PM with the following stereotypes of MARTE: ≪ComputingResource≫, ≪StorageResource≫ and ≪CommunicationMedia≫.

7.3. *The Platform Specific Model/Allocation Model in EML*

In the PSM, the MoC components (of the PIM) are mapped onto the components of the PM. The allocated methodology is the same as for AML (Figure 4). Moreover, the mapping of PIM onto the PM to form the PSM must not damage the semantic of the MoC. Actually, if more than two MoC components are mapped onto one component of the PM, the semantics of the point-to-point communication between the MoC components is not affected. But if two MoC components, which communicate between them, are mapped onto two different components of the PM, the semantic is not assured because the link of the communication between both hardware compo-

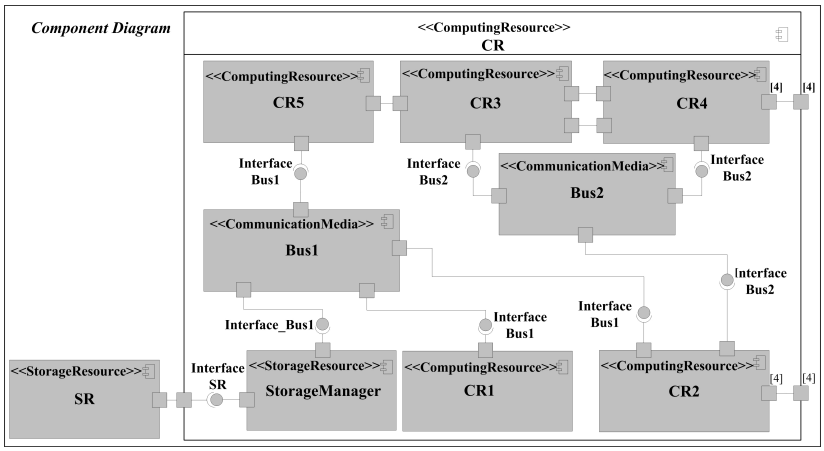


Figure 6. *An example of PM in EML with MARTE stereotypes*

nents can be a bus and not a point-to-point link. Therefore the semantics have to be guaranteed in a bus communication.

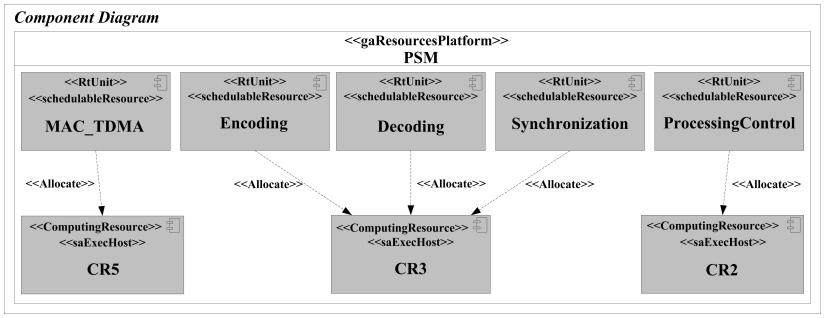


Figure 7. *An example of PSM with SAM stereotypes of MARTE*

7.4. *Analysis model*

To analyze the scheduling and the performances of the system, some information must be added on the models of this level. What is the signification of schedulability analysis? It provides the ability to evaluate time constraints and guarantee worst-case behavior of a real-time system. For the schedulability analysis, MARTE SAM sub-profile is recommended. This sub-profile offers the elements to add annotations in the different views of the model in order to evaluate the scheduling. The way to use the SAM sub-profile is explained below:

– In the PIM, the behavioral descriptions are annotated by time constraints and by the size of exchanged messages with the stereotypes of SAM; the sequence diagram is well adapted to add these annotations; the different scenarios of behavioral description form the workload behavioral model;

– In the PSM, the objects diagram is annotated in order to indicate the type of scheduling resource of each element of the model (Figure 7);

– The last step is to add a view in order to model the analysis context and the parametric analysis context. The analysis context model indicates the start and the stop conditions of the different behavioral scenarios. And the parametric analysis context is an instance of this analysis context model in which the values are set in order to simulate the scheduling.

After these three steps, the SAM of the system is defined for EML. The MOPCOM process does not specify the process and the tool, such as the Cheddar [SIN 04] tool, used to analyze the scheduling . It should be noted that the metamodels of UML and MARTE may be different from the metamodel of the syntax used in the selected analysis tool. Thus, it is necessary to translate EML model into another syntax (Figure 8). This transformation could be done with MDWorkbench environment (cf. section 9).

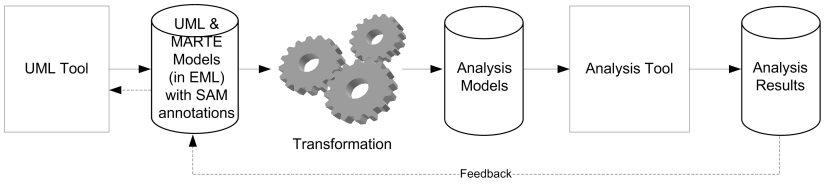


Figure 8. *Process of models analysis*

In addition, in this level, the PSM can be annotated with time constraints and elements of MARTE PAM sub-profile in order to analyze the system performances. The results of this analysis are still approximate. Precise performance analysis can be carried out in the Detailed Modeling Level.

8. Detailed Modeling Level

The DML defines the platform at a clock cycle tick precise definition, where the final target RTL model can be generated. At this level, hardware specification is finalized relying on generated hardware components or existing IP block.

8.1. *Platform model*

The platform defines the structural hardware components, which is a refinement of the PM defined in the above level, EML. A component diagram is used to model it.

MARTE HRM – *Hardware Resource Modeling* – sub-profile is used to define which kind of elements each object represents, such as ASIC – *Application Specific Integrated Circuit* –, PLD – *Programmable Logic Device* –, Clock, etc. MARTE SRM – *Software Resource Modeling* – sub-profile is used to model operating system properties, like tasks and virtual memory. MARTE SRM elements are not addressed here. All components of the platform must be stereotyped with MARTE HRM elements.

A platform is defined as a set of components connected through ports. For each port, a stereotype, which defines a communication protocol, is attached. A library is associated to each protocol stereotype, used in code generation. Figure 9 shows the elements in a platform model.

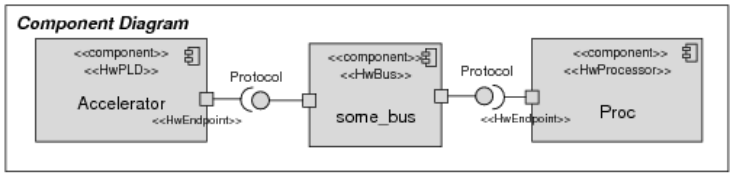


Figure 9. *Platform model*

A component with a ≪HwClock≫ must be present in the platform that is used to allow performance analysis and synchronous component code generation.

Component diagram The component diagram contains the platform resources. At least two stereotypes must be present for each component: ≪HwLogical≫ and ≪HwPhysical≫. Both must be present to characterize DML. Components are used to model the platform as they are considered to be reusable units that offer services, abstracting their behavior. Each component must be identified by an IP number and version, which allows IP definition and reuse.

Components are connected together by UML ports, where the ports contain the stereotype ≪HwEndPoint≫. An endpoint is an interaction point to communicate with the component.

Protocol definition Inter-component communication is done by communication protocols. Protocols are defined as interfaces, where buses ports offer a protocol and other components ports require it. The set of available protocols is platform-dependent and the code generation tool must be aware of the available protocols.

8.2. *Allocation model*

Allocation at this level involves defining where functional objects – MoC Components – are placed in platform ones – MARTE HRM stereotyped objects –. Functional components are allocated onto platform ones (Figure 10). Application components are logical units with behavior. Such behavior will be executed in/by a platform component. It is important to remark that a component whose behavior definition contains a state machine must be allocated to a component connected to a clock or to a processor.

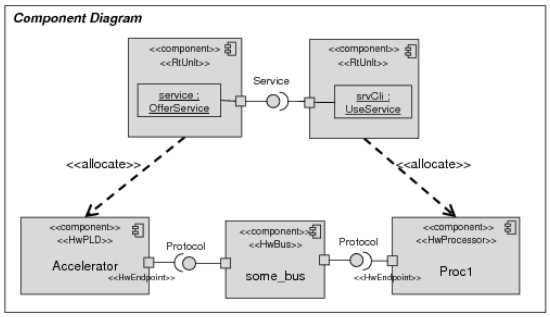


Figure 10. *Allocation model*

9. Tooling Support

The MDD/MARTE design process presented above is defined to be as much as possible independent from the implementation tools, so that it could be instantiated into any other tool (open-source modeler, java/EMF model transformer, etc.). Anyway for the MOPCOM project, this process relies on three main tools:

– Rhapsody [TEL], a UML Modeler targeted to RT – *Real Time* – embedded applications, to model the application and the platform,

– Kermeta, a Metamodeler [INR , MUL 05], to formalize (concepts and constraints), validate the metamodels and specify the transformation steps,

– MDWorkbench platform [SOD], a model-driven workbench to transform models (model-to-model) and to generate code or documentation from models (model-to-text).

Figure 11 depicts our tooling workflow; all tools being based on MDA standards from OMG (MDA, UML, MOF, XMI) and Eclipse (EMF, EMOF, ECore). For MOPCOM, an implementation of MARTE profile in Rhapsody has been developed.

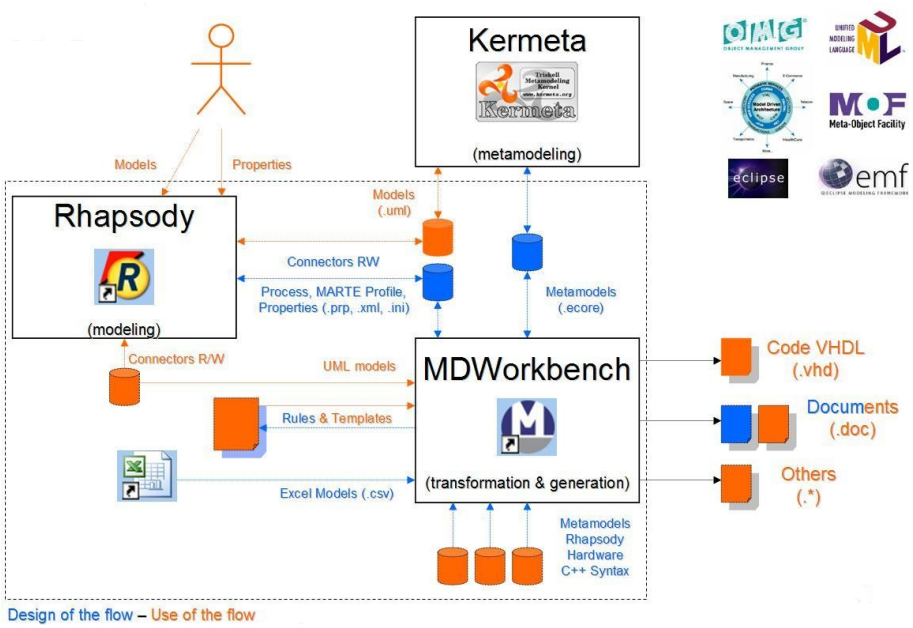


Figure 11. *MOPCOM process tools interactions*

9.1. *Process validation through metamodeling with Kermeta*

Kermeta environment from INRIA is devoted to model manipulations (composition, merge, etc.). It relies on an imperative object-oriented language and gives operational semantics to metamodels in a non invasive way, taking advantage of a built-in *aspect mechanism* to weave Kermeta code to Ecore model elements. For model validation, Kermeta provides the same capacities than OCL to define rules and check models. So, a Model Checker based on Kermeta environment has been developed. It was used to validate models transformations and to check models compliance to MOPCOM modeling rules at each step of the MOPCOM process (cf. figure 1).

9.2. *Model transformation and generation with MDWorkbench platform*

MDWorkbench platform from Sodius includes a complete environment to handle metamodels and models, and to design, execute, test and deploy model-to-model transformation rules and model-to-text generation rules. It is seamlessly integrated into Rhapsody, known as RulesPlayer and RulesComposer. The RulesPlayer can be

seen as a black-box runtime generation engine, while the RulesComposer is the rule editor, for designing and modifying the transformation and generation rule sets. MDWorkbench is delivered as an Eclipse plugin, and built as a model-driven extension to this powerful environment with many helpful capabilities (edition, windowing, debug, data handling, versioning, etc.). It includes the required mechanisms based on EMF/Ecore, to build, browse and import/export any metamodel, such as the Rhapsody metamodel, and model.

The generator is delivered as a white-box Rhapsody add-on. All transformation and generation rules are available for customization with MQL – *Model Query Language* – dedicated to model transformation, and TGL – *Text Generation Language* – for code or doc generation. MQL and TGL offer java-like main constructs (declarations, selections and loops) and a high-level dotted notation to handle lists of model elements.

MDWorkbench incorporates a powerful document generator, based on a gateway with Microsoft Word®. An XML – *Extensible Markup Language* – document schema is provided that enables users to define their own document templates within Word, in compliance with their company's graphic policy or with any standard. This greatly enhances the power of a model-driven design approach where the application/platform models become the reference, and where the development documentation is automatically generated from the model.

10. HDL Code Generation

Code generation is a capacity generally supported by a MDD process. For ESL, the target language is an HDL such as VHDL or SystemC. A VHDL code generator for Rhapsody, presented hereafter, has been developed in the MOPCOM project.

10.1. *VHDL code generation*

VHDL code generator input is a DML model, lowest abstraction level within the process, which includes the application and platform packages, as well as the allocation of the application class instances on the platform class instances. As usual, package class/object and statechart diagrams feed code generation, which targets synthesizable VHDL code.

The structure part is derived from the platform model, where VHDL entities are derived from instances (and instance hierarchy) of platform classes. Obviously, a UML port is not at all mapped to a VHDL port, but corresponds to a communication channel between system blocks, which also represents the entity port set. Data and control VHDL ports are determined according to the UML port and its mapping on a model of computation (and communication protocol) and can be imported from existing libraries. Additional VHDL properties enable definition of a clock (with edge), and optional asynchronous/synchronous resets (with polarity).

The behavioral part is derived from the application model, where VHDL architectures are mainly issued from attributes, operations and state machines. All the required data types are defined into associated packages, according to the UML types (enumerations, language-defined, structure and hierarchical types, constants, etc.). The VHDL processes handle internal signals and variables that are declared according to data definitions in the scope of each class. Nearly all concepts of UML state machines are supported by the generator. Briefly, a finite state machine leads to the definition of an enumerated type for the active state, one per composite state (containing sub-states). The code structure is based on an edge-clocked case VHDL statement, and all trigger, guard and action expressions (on transitions, entering, in or exiting states) can be generated either in line, or in single procedures.

The allocation package brings additional information about the mapping of the application on the platform. The generator combines the declared entity ports and the data/control needs of the architecture to map the components and if required (if not point-to-point), instantiate the control code (or state machine) of the communication channel protocols. Depending on the communication channel, several basic mechanisms are provided to handle events (transient or registered) and the required glue logic is automatically inserted.

The generation process consists on two steps: the first step is a model transformation from Rhapsody to an intermediate hardware model; the second step is a generation from hardware model to VHDL code. The hardware model is in conformance with a hardware metamodel which gathers the main semantic concepts that are required to describe a complex electronic device at the Register Transfer Level.

10.2. *Rhapsody integration*

The HDL generator is currently embedded into Rhapsody in C++ and VHDL generation is just another environment of the Rhapsody configuration. Extra properties have been defined, in order to setup the generator, and to select the coding style, naming rules and generation parameters. VHDL code can be edited directly into the UML tool, as each in-the-scope class is associated to a specification file (VHDL package) and an implementation file (VHDL entity/architecture). The usual build-and-make phase has been converted into a possible call to a VHDL synthesizer, and a current bridge with web-free Xilinx XST. The bridge allows displaying any warning and error messages back into the Rhapsody build window, with dynamic link to the specified code line.

It is also possible implement interfaces with some other tools, either adjacent modeling tools such as Doors®(Telelogic) or Matlab/Simulink®(the Mathworks), or EDA – *Electronic Design Automation* – downstream tools such as EDK®(Xilinx), Altera®(SoPC Builder) or any other modeling tools from EDA tool suppliers.

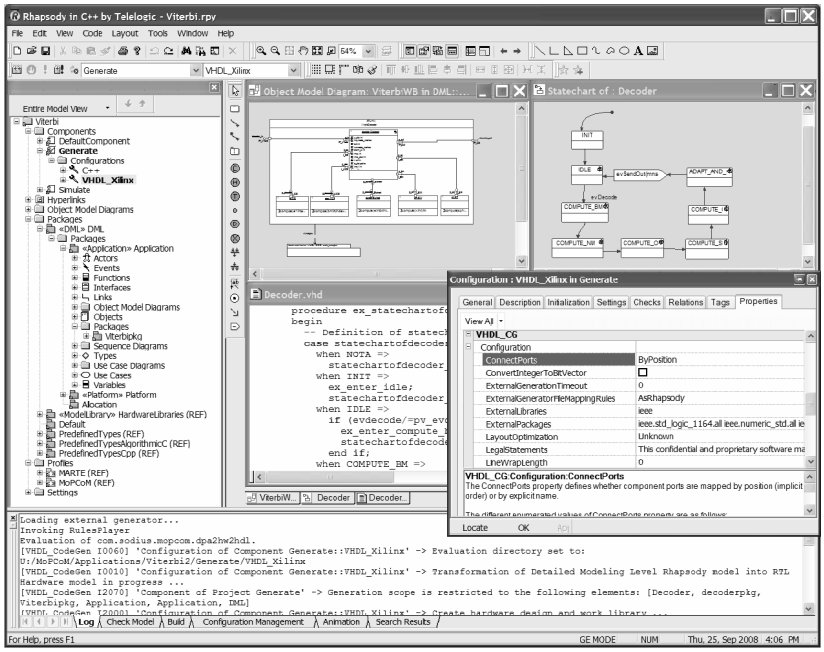


Figure 12. *VHDL code generator integration into Rhapsody*

11. Conclusion

In this paper, we have discussed an ESL process based on MDD and MARTE profile. This process emphasizes application and platform modeling at different levels of abstraction and the allocation of the application models to the platform models. For each level, we presented the selected MARTE stereotypes and the constraints related to their use. We have also outlined the MDD tooling developed to support the process: for example a MARTE profile implementation in a UML modeler, and a VHDL code generator.

We reckon that the emergence of MARTE profile will widespread the use of MDD methodology in ESL domain. UML and MDD methodology are supported by a large number of commercial and freeware development tools that will offer new possibilities to the ESL community.

Acknowledgements

The UML/MDD approach presented above is experimented in the RNTL research program MOPCOM SoC/SoPC supported by the French Agence Nationale de la Recherche (contract 2006 TLOG 022 01), the "Media and Networks" "cluster of clusters" and the Brittany and Pays de la Loire regions.

12. References

[ART 08] ARTHURS G., "White paper: Model-based system engineering", IBM, October 2008.

[BOE 88] BOEHM B. W., "A Spiral Model of Software Development and Enhancement", *Computer*, May 1988, p. 61-72.

[BUC 02] BUCK J., HA S., LEE E. A., MESSERSCHMITT D. G., "Ptolemy: a framework for simulating and prototyping heterogeneous systems", *IEEE*, vol. 10, 2002, p. 527–543, Kluwer Academic Publishers.

[CAL 08] CALVEZ J., "The MCSE Methodology overview", report , 2008, Cofluent Design.

[CEA] CEA, "Papyrus UML2 Modeler", `http://www.papyrusuml.org`.

[COF] COFLUENT_DESIGN, "CoFluent Studio", `http://www.cofluentdesign.com/`.

[GAJ 83] GAJSKI D. D., KUHN R. H., "New VLSI Tools", *Computer*, vol. 16, num. 12, 1983, p. 11–14, IEEE Computer Society Press.

[GAM 95] GAMMA E., HELM R., JOHNSON R., VLISSIDES J., *Design Patterns: Elements of Reusable Object-Oriented Software*, Num. ISBN 0-201-63361-2, Addison-Wesley, 1995.

[GER 02] GERARD S., TERRIER F., TANGUY Y., "Using the Model Paradigm for Real-Time Systems Development: ACCORD/UML", SPRINGLINK, Ed., *Advances in Object-Oriented Information Systems*, vol. 2426/2002 of *Lecture Notes in Computer Science*, 2002, p. 260-269.

[GER 03] GERARD S., TERRIER F., "UML for real-time: which native concepts to use?", *ACM*, vol. 13, 2003, p. 17–51, Kluwer Academic Publishers.

[HAC 07] HACHEMANI R., PALICOT J., MOY C., "A new standard recognition sensor for cognitive radio terminals", *EURASIP*, KESSARIANI, GREECE, 2007.

[INR] INRIA, "Kermeta metaprogramming environment", `http://www.kermeta.org`.

[ITR 07] ITRS, "Design", report , 2007, INTERNATIONAL TECHNOLOGY ROADMAP FOR SEMICONDUCTORS.

[KAO 04] KAOUANE L., AKIL M., GRANDPIERRE T., SOREL Y., "A methodology to implement real-time applications onto reconfigurable circuits", *J. Supercomput.*, vol. 30, num. 3, 2004, p. 283–301, Kluwer Academic Publishers.

[KAS 07] KASUYA A., TESFAYE T., "Verification methodologies in a TLM-to-RTL design flow", *DAC '07: Proceedings of the 44th annual conference on Design automation*, New York, NY, USA, 2007, ACM, p. 199–204.

[KOU 05] KOUDRI A., MEFTALI S., DEKEYSER J.-L., "IP integration in embedded systems modeling", *14th IP Based SoC Design Conference (IP-SoC 2005)*, Grenoble, France, Dec. 2005.

[KOU 08] KOUDRI A., AL., "Using MARTE in a Co-Design Methodology", *DATE*, 2008, Workshop MARTE.

[LEB 08] LE BOLZER F., GUILLOUARD S., GUGUEN C., FONTAINE P., MONNIER R., "Prodim@ges - A new Video Production Environment based on IP wireless and optical links", *NEM'SUMMIT*, Saint-Malo, FRANCE, October 2008.

[MIT 01] MITOLA JOSEPH I., "Cognitive radio for flexible mobile multimedia communications", *Mob. Netw. Appl.*, vol. 6, num. 5, 2001, p. 435–441, Kluwer Academic Publishers.

[MOP 07] MOPCOM, "MoPCoM SoC/SoPC Project", `http://www.mopcom.fr`, 2007.

[MUL 05] MULLER P.-A., FLEUREY F., JÉZÉQUEL J.-M., "Weaving Executability into Object-Oriented Meta-Languages", *Proc. of MODELS/UML*, LNCS, Montego Bay, Jamaica, 2005, Springer.

[OMG 03] OMG, "MDA Guide Version 1.0.1", report , 2003, Object Management Group.

[OMG 05a] OMG, "UML Profile for Schedulability, Performance, and Time, version 1.1", report num. formal/2005-01-02, 2005, Object Management Group.

[OMG 05b] OMG, "A UML Profile for SoC", report num. Realtime - 2005-04-12, 2005.

[OMG 06a] OMG, "Object Constraint Language", report num. formal/2006-05-01, 2006, Object Management Group.

[OMG 06b] OMG, "UML 2.1 Infrastructure", report num. ptc/06-04-03, 2006, Object Management Group.

[OMG 07] OMG, "UML Profile for MARTE, Beta 1", report num. ptc/07-08-04, 2007, Object Management Group.

[OMG 08a] OMG, "Systems Modeling Language Specification v1.1", report num. ptc/2008-05-16, 2008, Object Management Group.

[OMG 08b] OMG, "UML Profile for Modeling QoS and Fault Tolerance Characteristics and Mechanisms", report num. formal-2008-04-05, 2008, Object Management Group.

[OSC 05] OSCI, "IEEE Standard SystemC Language Reference Manual", report num. IEEE Std 1666-2005, 2005, IEEE Computer Society.

[PIE 08] PIEL E., ATTITALAH R. B., MARQUET P., MEFTALI S., NIAR S., ETIEN A., DEKEYSER J.-L., BOULET P., "Gaspard2: from MARTE to SystemC Simulation", , 2008.

[RIC 05] RICCOBENE E., SCANDURRA P., ROSTI A., BOCCHIO S., "A SoC Design Methodology Involving a UML 2.0 Profile for SystemC", *Proc. of the conference on Design, Automation and Test in Europe*, Washington, DC, USA, 2005, IEEE Computer Society, p. 704-709.

[RIC 07] RICCOBENE E., SCANDURRA P., ROSTI A., BOCCHIO S., "Designing a Unified Process for Embedded Systems", *In the Fourth International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES)*, Braga, PORTUGAL, March 2007, IEEE Computer Society.

[SAN 04] SANGIOVANNI-VINCENTELLI A., CARLONI L., BERNARDINIS F. D., SGROI M., "Benefits and challenges for platform-based design", *DAC '04: Proceedings of the 41st annual conference on Design automation*, New York, NY, USA, 2004, ACM, p. 409–414.

[SIN 04] SINGHOFF F., LEGRAND J., NANA L., MARCÉ L., "Cheddar : a Flexible Real Time Scheduling Framework", *Proc. of International Conference on Special Interest Group on Ada (SIGAda)*, Atlanta, Georgia, USA, November 2004, ACM.

[SOD] SODIUS, "MDWorkbench platform", `http://www.mdworkbench.com`.

[TEL] TELELOGIC, "Rhapsody UML modeler", `http://www.telelogic.com/products/rhapsody/index.cfm`.