A CO-DESIGN METHODOLOGY BASED ON MODEL DRIVEN ENGINEERING FOR SDR EQUIPMENTS

Stéphane LECOMTE[1,2], Samuel GUILLOUARD[1], Christophe MOY[2], Pierre LERAY[2], Philippe SOULARD[3]
[1](THOMSON R&D, Cesson-Sévigné, France, stephane.lecomte@thomson.net)
[2](SUPELEC/IETR, Rennes, France, christophe.moy@supelec.fr)
[3](SODIUS, Nantes, France, psoulard@sodius.com)

ABSTRACT

This paper presents the MOPCOM methodology, primarily developed to enable the efficient design of SDR – *Software Defined Radio* – equipments. Based on UML/MDE approach, it could be advantageously applied for the design of any real-time embedded systems. The MOPCOM methodology defines a set of rules to build UML models for embedded systems, from which HDL code is automatically generated by means of MDE – *Model driven Engineering* – techniques. The UML/MARTE profile is used to describe real-time properties and to perform platform modeling. Three abstraction levels are defined: abstract, execution and detailed modeling levels (AML, EML and DML, respectively). The second one will be particularly explained and the overall methodology will be evaluated through a SDR case study.

1. INTRODUCTION

Thanks to the ever-increasing performance of digital electronics, an embedded system can be integrated on a single chip: a SoC – *System on Chip* – or a SoPC – *System on Programmable Component* – inside FPGA reconfigurable components.

In parallel, to catch up with this hardware capacity, a dramatic enhancement of hardware design facility is required to fill the productivity gap. Another important challenge, induced by the design of SoC/SoPC, consists in reducing Time-to-Market and the cost due to the rapid evolution of the technology. To achieve those goals, SoC/SoPC design methodologies have to tackle co-design issues such as design space exploration, reuse of IPs - *Intellectual Property* - and high level synthesis. Besides ESL – *Electronic System Level* – modeling approaches, UML [1] – *Unified Modeling Language* – originally dedicated to software development has extended its scope to system or real-time embedded application development, including hardware design. 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 capabilities, such as UML MARTE profile [2]. MARTE has been standardized by OMG. Following a MDA methodology also contributes at capitalizing, as well, the achievements of the ESL community.

Moreover, MDA [3] – *Model Driven Architecture* – derived from MDE, promotes a development methodology based on models transformations at several levels of abstraction and that follows the well known Y-chart co-design approach.

The following section presents a state of art of the modeling approaches for codesign conception of RTES – *Real Time Embedded Systems* –. In the third section, we show an overview of our MOPCOM methodology based on MDA and UML. In the section 4, we explain with more details the middle level of the process. The two following sections present an overview of the tooling MOPCOM environment. Finally an experimentation of this approach, carried out in the frame of the MOPCOM SoC/SoPC project [4] is presented.

2. RELATED WORK

The developments of SoC/SoPC and RTES strongly need the availability of reliable methodology, which represents a wide research topic for the ESL community. Such a methodology, based on appropriate languages and tools, can help to handle simultaneously market pressure (time-to-market, competitiveness), fast changing technology grade as well as migration towards more complicated/sophisticated standards [5].

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 [6],

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 [7].

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 [8] 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.

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, execution platform, possibly represented at several levels of abstraction, and QoS – *Quality of Service*. Modeling and analysis activities must be placed 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, or 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. A few examples are given below.

The methodology MCSE – *Méthodologie de Conception des Systèmes Electroniques* –, proposed in [9], enables design space exploration through the use of the SystemC TLM language.

The Ptolemy environment from UC Berkeley [10] 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.

In the context of 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 list:

- the UML4SOC OMG's profile is dedicated to describe System on Chip [11];
- the UML for SystemC profile proposed in [12] gathers the capabilities of UML and SystemC;
- the UML for MARTE OMG's profile can be viewed as an improvement of the SPT profile [13].

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

The ACCORD/UML methodology [14] aims at using UML concepts to design RTES.

A first trial for SDR profile has been proposed in the A3S profile which was a first attempt to design SDR in a MDA perspective [15], but has not been followed by a standardization effort.

In [16], the authors proposes a development process for embedded systems and SoC called UPES – *Unified Process for Embedded Systems* –, based on UML for SystemC profile.

The Gaspard Methodology [17], based on MDA approach, is intended to provide a framework for developing parallel and distributed applications implemented on SoC.

3. MOPCOM METHODOLOGY

Defined in [18], the MOPCOM methodology is a refinement of the MDA Y-chart dedicated to design space exploration and Platform Based Design. MDA techniques coupled with UML are used to perform code generation (VHDL, C/C++ for HLS – *High-Level Synthesis* – and SystemC). It takes as input functional, non-functional and allocation requirements expressed in SysML [19]. Figure 1gives an overview of the process, highlighting 3 modeling levels:

- **AML** – *Abstract Modeling Level* – is intended to provide the description of the expected level of concurrency and pipeline through the mapping of functional blocks onto a virtual execution platform;
- **EML** – *Execution Modeling Level* – is intended to provide the topology of a generic platform defined in terms of execution, communication or storage nodes in order to proceed to coarse grain analysis;
- **DML** – *Detailed Modeling Level* – is intended to provide a detailed description of the platform in order to proceed to fine grained analysis. It allows RTL code generation for hardware (VHDL) and software (C) parts including glue code (drivers).

For each level, we identify which subsets of MARTE are mandatory and we capture modeling rules constraints applied to the process model.

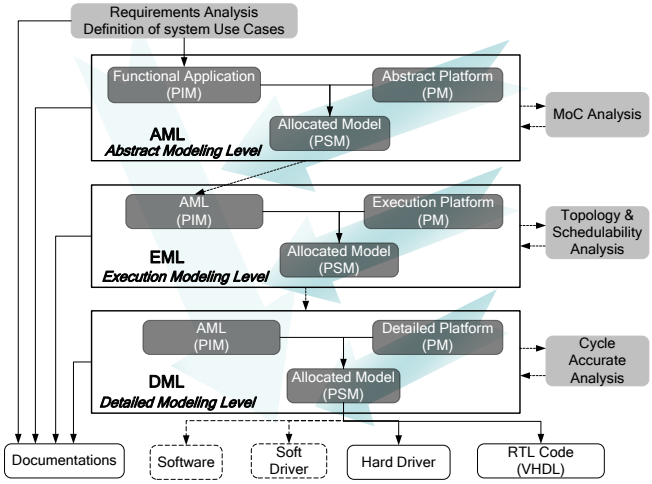

Figure 1 – MOPCOM process overview

4. EMBEDDED SYSTEM MODELING

In this section, we detail EML, the middle level of the MOPCOM process. The EML - *Execution Modeling Level* - 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. Platform and allocation are defined regarding trade-offs between implementation, performances and costs. In the context of design space exploration, all those aspects must be checked by a top-down refinement analysis.

4.1. The Platform Independent Model in EML

The PIM of EML is the output model of the AML. So the PIM model in EML is composed of MoC components on which functions of the application are allocated. The links between MoC components are point-to-point with specific semantic of data transfer [20]. A MoC component offers a service of concurrent execution for the function(s) allocated in it.

A refactoring of PIM in EML is possible if necessary. We can transform the functional architecture in order to allocate the PIM onto PM – *Platform Model* –. In fact, Analysis feedbacks drive potential refactoring on the platform as well as on the MoC or the functional architecture. Actually, those refactoring depend mainly on the skill of the engineers to proceed to the right adjustments.

4.2. 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. In fact, PM of EML design emphasizes the number of processing elements, organization of data or communication media and their characteristics. Interfaces between connected elements and communication protocols are considered from high level perspective: data are bit true and inaccurately timed. 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.

To model the PM, we use the class/object diagram or the component diagram of UML associated with MARTE profile.

The MOPCOM methodological tool at this level is MARTE GRM – *Generic Resource Modeling* – subprofile. Figure 2 shows an example of PM with the following stereotypes of MARTE: «ComputingResource», «StorageResource» and «CommunicationMedia». Each resource of PM is characterized by these services offered, several non-functional properties such as latency, frequency, datapath width.

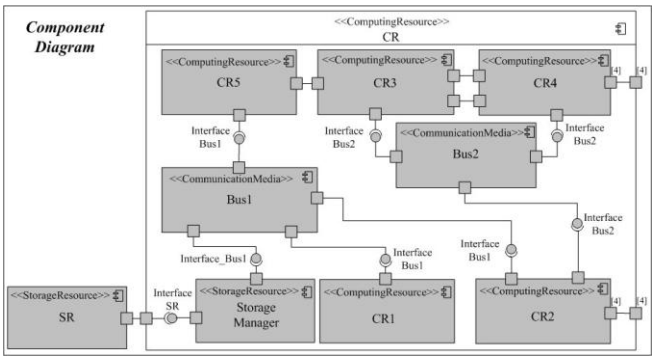

Figure 2 – Platform model in EML

4.3. The Platform Specific Model in EML

The PSM - *Platform Specific Model* -, allocation model, is built to allocate the functionalities of PIM model into the components of the virtual platform (PM).

The link of allocation between a MoC component (concurrency virtual component) of PIM and a component of PM is done using a UML dependency. This dependency is characterized by the «Allocate» stereotype of the MARTE alloc sub-profile. Thus this stereotype helps in specifying whether it is a space or temporal allocation. More than one

MoC of one virtual component of the platform can be allocated.
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 to each other, are mapped onto two different components of the PM, the semantic is not ensured because the communication between both hardware components can be done through a bus and not necessarily via a simple point-to-point link. Therefore, the semantics have to be guaranteed for bus communication.

4.4 Analysis model

To analyze the scheduling and the performances of the system, some information must be necessary 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 sub-packages, SAM – *Schedulability Analysis Modeling* – and PAM – *Performance Analysis Modeling* – are used.

Analysis scenarios have to be defined. For each scenario the context and the parameters are modeled, thus requiring an indication about the type of scheduling resource of each element of the model (shown in Figure 3).

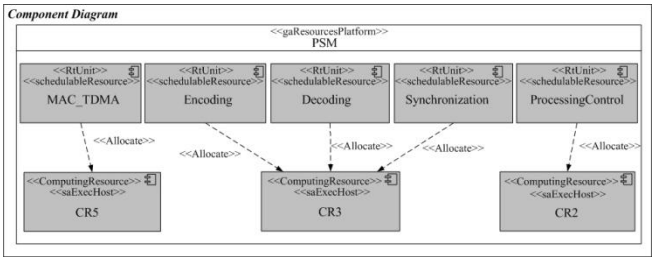

Figure 3 – PSM with SAM stereotype of MARTE

For example, Analysis scenarios emphasize data rates and latencies as well as memory size or context switching in the case of dynamic reconfiguration.

When the model analysis is completed, several possibilities to analyze the different scenarios are possible. We can use a dedicated analysis tool. Another possibility consists in generating the equivalent model in SystemC in order to simulate the analysis scenarios. In both approaches, it should be noted that the metamodels of UML and MARTE might be different from the metamodel of the syntax used in the selected analysis tool or SystemC language. Thus, it is necessary to translate EML model into another syntax. This transformation could be done with MDWorkbench environment (cf. section 5).

5. TOOLING SUPPORT

The MOPCOM process tooling relies on tools related to OMG standards (MDA, UML, MOF, XMI) and Eclipse (EMF, EMOF, Ecore):

- The KerMeta language from INRIA [21] is used to formalize and validate the metamodels (concepts and constraints);
- The Rhapsody UML Modeler [22] is used to model applications as well as platforms according to the defined levels;
- The Sodius MDWorkbench tool [23] is used to transform models (model-to-model) and generate code or documentation from models.

The generator is delivered as a white-box add-on, where all transformations and generation rules are available for any customization. In addition,

RTES modeling requires an action language for low-level expressions to complete the high-level UML semantics and diagrams, and to specify operation bodies, trigger/guard/action on transition and states as well as data declarations. The selection of the right action language raises questions about textual or graphical notation, and general versus HDL-specific language accessible to designers, taking into account learning curves. The C++ language turned to be a convenient choice and only a C++ subset is used in the models (along with some macros for event and port handling). C++ expressions are parsed for VHDL generation thanks to a C++ syntactic metamodel allowing grammar to model transformations.

6. CODE GENERATION

The MOPCOM process associated with MOPCOM tools offers three different generation of code: VHDL code, C/C++ code for HLS tools, and a SystemC code.

6.1 VHDL code generator

VHDL code generator input is a DML model, lowest abstraction level within the process (as shown in Figure 1), which includes the application and platform packages, as well as the allocation of the application class instances on the platform class instances. Structural parts are derived from the platform model, where VHDL entities are derived from hierarchy of instances. UML ports are translated to VHDL ports thanks to communication protocols and data. Behavioral parts are derived from application models, where VHDL architectures are mainly issued from attributes, operations and state machines.

6.2 C/C++ code generator for HLS tool

In order to reuse an external IP, a VHDL black-box can be generate with the right interface of system element. VHDL IP block come from IP libraries, or from a High Level Synthesis tool, such as Catapult [24], which generate a VHDL IP from C/C++ code.

So the C/C++ generator code is based on the C++ description of an algorithm in the functional (PIM) model. Then the C/C++ for HLS code generator generates only the code for the element of the model tagged with <<HLS>>.

6.3 SystemC generator

In section 4, we have presented the EML level, inside which SystemC model could be used to perform the simulation. This requires a generation (from UML model to System C code), which has been developed and integrated in the MOPCOM methodology.

7. A CASE STUDY EXAMPLE

The case study is a TDMA/OFDM/MIMO wireless IP based on IEEE 802.16a standard and for which proprietary extensions have been proposed [25]. Our approach has been applied to the MIMO processing at the receiver side. Figure 4 shows a synoptic of the MIMO decoder.

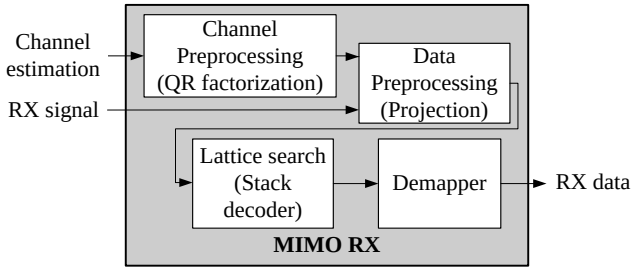

Figure 4 – Synoptic of the MIMO decoder

The user of the system can configure the system before starting the transmission, or the system can reconfigure itself with the properties of the received signal.

Figure 5 shows the interaction between the actors and the system, the Source corresponds to the Channel estimation and the RX signal while the Sink corresponds to the decoded RX data.

Figure 6 shows the different functions of the MIMO decoder. The *itsDecoderMIMO* object (instance of Decoder_MIMO class) controls the steps of the decoding, with a statechart. The *itsPreprocChannel*, *itsPrepocData* and *itsDemapper* objects are pure computing objects (no control). And the *itsStackDecoder* object is composed of computing and control.

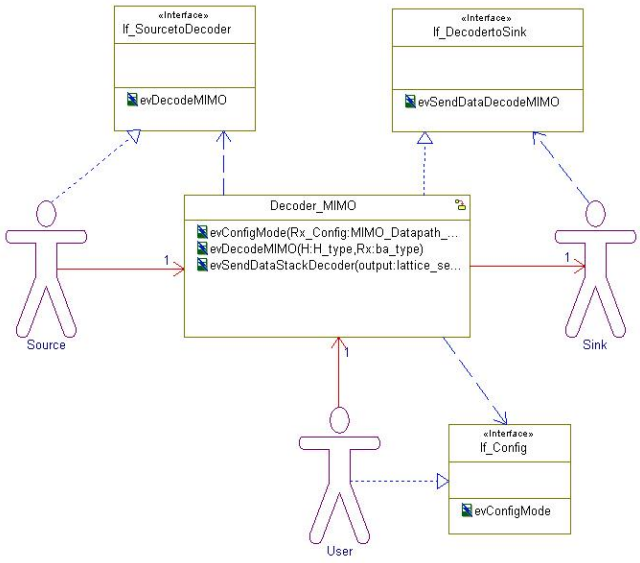

Figure 5 – Interaction view between actors and system

Figure 2 and 3 display two samples of the EML level of a subset of the proposed UML models (part of the virtual platform model and allocation of a part of the functional model to the virtual platform model in figure 2 and figure 3 respectively).

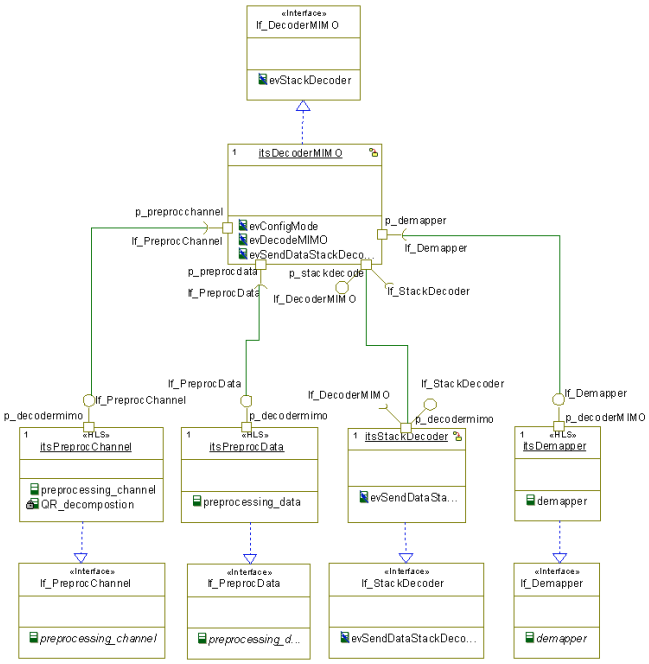

Figure 6 – Objects model diagram for the MIMO decoder

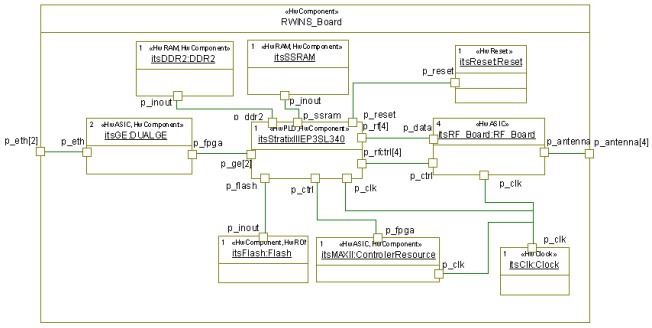

Figure 7 – Sample of model of target platform

The targeted platform for the implementation is a reconfigurable component that can perform self-reconfiguration. Figure 7 shows a sample of the modeling of the target platform on DML level (section 3). With MARTE profile [2], the type of each component is specified (with a stereotype) and the characteristics of this one (by completed the associated tags to the stereotype).

8. CONCLUSION

A subset of the MOPCOM Soc/SoPC methodology has been presented. The main objective of this approach is to offer a co-design methodology for RTE systems based on MDA approaches, using UML 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, the selected MARTE stereotypes and the related constraints have been presented. This process has been applied on a SDR case study, for which a C/C++ code and a synthesizable RTL code have been generated. Our future works will consist in integrating the modeling of the partial reconfiguration (PR) of FPGA in the process and to simulate the analysis model of EML with SystemC language.

9. ACKNOWLEDGMENT

The UML/MDA approach presented above is experimented in the RNTL06 research program MOPCOM SoC/SoPC supported by the French National Research Agency (ANR – contract 2006 TLOG 022 01), the “cluster of clusters”, “Media and Networks” and the Brittany and Pays de la Loire regions.

11. REFERENCES

[1] OMG, “UML 2.1 Infrastructure”, report num. ptc/06-04-03, Object Management Group, 2006.
[2] OMG, “UML Profile for MARTE, Beta 2”, report num. ptc/2008-06-08, Object Management Group, 2008.
[3] OMG, “MDA Guide Version 1.0.1”, report num. omg/2003-06-01, Object Management Group, 2003.
[4] MOPCOM partners, “MOPCOM SoC/SoPC Project”, www.mopcom.fr, 2007.
[5] S. Gerard, F. Terrier, “UML for real-time: wich native concepts to use?”, ACM, vol. 13, p. 17-51, Kluwer Academic Publishers, 2003.
[6] D. D. Gajski, R. H. Kuhn, “New VLSI Tools”, *IEEE Computer*, vol. 16, num.12, p. 11-14, IEEE Computer Society Press, December 1983.
[7] A. Sangiovanni-Vincentelli, L. Carloni, F. D. Bernardinis, M. Sgroi, “Benefits and challenges for platform-based design”, Proceedings of the 41st annual conference on Design Automation (DAC), New-York, USA, ACM, p. 409-414, 2004.
[8] OSCI, “IEEE Standard SystemC Language Reference Manual”, report num. IEEE Std 1666-2005, IEEE Computer Society, 2005.
[9] J. Calvez, “The MCSE Methodology overview”, report, Cofluent Design, 2008.
[10] J. Buck, S. Ha, E. A. Lee, D. G. Messerschmitt, “Ptolemy: a framework for simulating and prototyping heterogeneous systems”, IEEE, vol. 10, p. 527-543, Kluwer Academic Publishers, 2002.
[11] OMG, “UML Profile for System on a Chip”, report num. formal/06-08-01, Object Management Group, 2006.
[12] E. Riccobene, P. Scandurra, A. Rosti, S. Bocchio, “A SoC Design Methodology Involving a UML 2.0 Profile for SystemC”, Proc. of the conference on Design, Automation and Test in Europe (DATE), Munich, Germany, IEEE Computer Society, p. 707-709, March 2005.
[13] OMG, “UML Profile for Schedulability, Performance, and Time”, report num. formal/2005-01-02, OMG, 2005.
[14] S. Gerard, F. Terrier, Y. Tanguy, “Using the Model Paradigm for Real-Time Systems development: ACCORD/UML”, *Advances in Object-Oriented Information Systems*, vol. 2426/2002 of Lecture Notes in Computer Science, p. 260-269, SPRINGLINK, Ed., 2002.
[15] C. Moy, M. Raulet, S. Rouxel, J.P. Digeut, G. Gogniat, P. Desfray, N. Bulteau, J.E. Goubard, Y. Denef, “Uml profile for waveform Signal Processing Systems Abstraction”, SDR Technical Conference, Phoenix, USA, November 2004.
[16] E. Riccobene, P. Scandurra, A. Rosti, S. Bocchio, “Designing a Unified process for Embedded Systems”, In the 4th international Workshop on MOdel-based Methodologies for Pervasive and Embedded Software (MOMPES), Braga, Portugal, IEEE Computer Society, March 2007.
[17] E. Piel, J.L. Dekeyser, P. Boulet and Al., “Gaspard 2: from MARTE to SystemC Simulation”, 2008.
[18] A. Koudri et al. “Using MARTE in a co-design Methodology”, Workshop MARTE in DATE, Munich, Germany, March 2008.
[19] OMG, “Systems Modeling Language”, report num. ptc/2008-05-16, Object Management Group, 2008.
[20] A. Koudri, J. Champeau, D. Aulagnier, D. Vojtisek, “Processus de codesign UML/MARTE”, Neptune, Paris, France, May 2009.
[21] INRIA Triskell Team, « KerMeta », www.kermeta.org
[22] Telelogic, an IBM Company, “Rhapsody UML modeler”.
[23] SODIUS, “MDWorkbench platform”, www.mdworkbench.com.
[24] Mentor Graphics, “Catapult Synthesis tool”, www.mentor.com
[25] F. Le Bolzer and Al., “Prodim@ges - A new Video Production Environment based on IP wireless and optical links”, NEM’SUMMIT, Saint-Malo, France, October 2008.