UML Framework for PIM and PSM Verification of SDR Systems

Samuel Rouxel, Jean-Philippe Diguet, Guy Gogniat, Nicolas Bulteau, Jonathan Carre-Gourdin, Jean-Etienne Goubard, Christophe Moy

▶ **To cite this version:**

Samuel Rouxel, Jean-Philippe Diguet, Guy Gogniat, Nicolas Bulteau, Jonathan Carre-Gourdin, et al.. UML Framework for PIM and PSM Verification of SDR Systems. SDR Forum Technical Conference'05, 2005, Anaheim, CA, United States. hal-00084148

HAL Id: hal-00084148
https://hal.science/hal-00084148

Submitted on 5 Jul 2006

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.

UML FRAMEWORK FOR PIM AND PSM VERIFICATION OF SDR SYSTEMS

Samuel ROUXEL, Jean-Philippe DIGUET, Guy GOGNIAT - Lester, Université de Bretagne Sud, France
<samuel.rouxel, jean-philippe.diguet, guy.gogniat>@univ-ubs.fr
Nicolas BULTEAU, Jonathan CARRE-GOURDIN - Softeam, France
<nicolas.bulteau, jonathan.carre-gourdin>@softeam.fr
Jean-Etienne GOUBARD - Thales Communications, France
jean-etienne.goubard@fr.thalesgroup.com
Christophe MOY – Supélec SCEE/IETR, France
christophe.moy@rennes.supelec.fr

ABSTRACT

The MDA concept was first introduced within the software community where it has shown its efficiency to deal with highly complex systems; however there is no equivalent in the embedded system domain where real time and power constraints have to be met. In order to fill the gap between UML specification of SDR systems and embedded platform, new formalisms and associated tools have to be defined. The A3S project addresses such a challenge which drastically leverages design productivity. A comprehensive framework that enables the specification of a SW application (PIM), an HW platform and an HW/SW deployment (PSM) has been defined and fully integrated. The A3S project relies on extensive verifications performed at three levels: component, application/platform, and deployment which enable fast prototyping of embedded systems. Designers have the ability to explore several potential deployments and to check if real time and power constraints can be met. A UMTS receiver/transmitter has been explored, implemented and tested in a few days which show the benefit of our approach to shrink design time.

1. INTRODUCTION

The A3S project which addresses Adequacy between Architecture and System Application, proposes a methodology to design SDR systems. The design steps of A3S are located after a functional validation and characterization of IPs (Intellectual Property – here understood as the implementation of processing blocks developed in a programming language as C or VHDL). The design steps consisting of assembling IPs on an embedded platform and to verify the achievable performance are error-prone and time consuming, especially when exploring several solutions. This is the aim of A3S to help the deployment of the IPs of an SDR system and to verify the correctness of the solution. The choice of UML ensures a user-friendly environment which is a commonly adopted standard in the community of system designers.

The A3S project aims at performing extensive non-functional coherence verification of software radio architecture specifications (hardware) and application requirements (software) with UML-based models. This paper describes the achievement of the A3S project. Intermediate steps of the project, concerning HW/SW modeling and A3S profile, have already been detailed in previous publications (respectively [1] and [2]).

A3S defines and proposes a methodology for the description in UML of both, hardware platforms made of DSP, FPGA, GPP, FIFO, buses, etc. and software applications for radio physical layers (SPS - Signal Processing System) and higher layers. This methodology is based on the OMG profile "PIM and PSM for Software Radio Components", extended through a new profile called A3S profile, in order to cope with low level description requirements which are mandatory for performances estimation. The A3S project is to our knowledge the most achieved approach for the design of low level SDR systems under UML. This paper particularly focuses on the verification of the system design with a step by step approach based on MDA principles.

The remainder of the paper is organized as follows: section 2 presents significant studies related to our work and stresses our contribution. Section 3 proposes an overall description of the three levels of verification within the A3S design flow. Section 4 presents the underlying UML meta-model used to support our approach and sections 5 and 6 detail each level of verification. Finally, section 7 demonstrates the A3S approach through a UMTS receiver/transmitter case study.

2. RELATED WORK

Some other projects or software have already been realized around software radio design aspects, such as code generator producing SCA artifacts for Corba compliant targets using the SCA core framework [3], or research projects targeting MDA for SoC design using ISP

profile [4][5] in order to produce a SystemC simulation code [6]. Some projects also exist for SW/HW analysis, which rely on UML and MDA such as [7], or on framework issued from UML specification either for software design space exploration based on performance and power estimation [8], or for communication conflict analysis in a SoC context [9]. UML2.0 profile for SoC is another initiative that reached the approval step in September 2005.

Each of these project or tools cover an aspect of a high level software radio verification system. Generally, it can be observed that some federating elements are missing, like for example SoC meta-model, or that the analysis method they present is limited to IDL, type compatibility and pure software concerns and does not address embedded systems issues such as memory, bus, real-time, power, or that the tool/concept is under development/definition.

In conclusion we observe that the necessity of high abstraction level for SoC design is steering the EDA community to UML as a general framework. Regarding the current initiatives we observe that our approach is original in the way that we combine an SDR UML profile for implementing a real MDA approach with EDA tools for multi-level verifications from the type compatibility to the scheduling and memory use analysis over an heterogeneous platform.

3. A MULTI-LEVEL VERIFICATION APPROACH

The verifications performed by A3S are categorized in three layers, each of them relying on the specifications defined in the A3S profile [2]. In order to increase the productivity of the designers several wizards have been implemented, driving their design to fit the verifications processes. All the verifications are automatically performed by the tools, mitigating the designer work. Two tools are used during the design process: the first one called modeling and specification tool enables the definition of the components, the specification of the application/platform and the HW/SW deployment. The second tool takes as input the results of the previous steps and performs non-functional verifications to check the schedulability of the system and to estimate the workload of each resource within the embedded platform. Figure 1 illustrates the A3S project and the interactions with the designers.

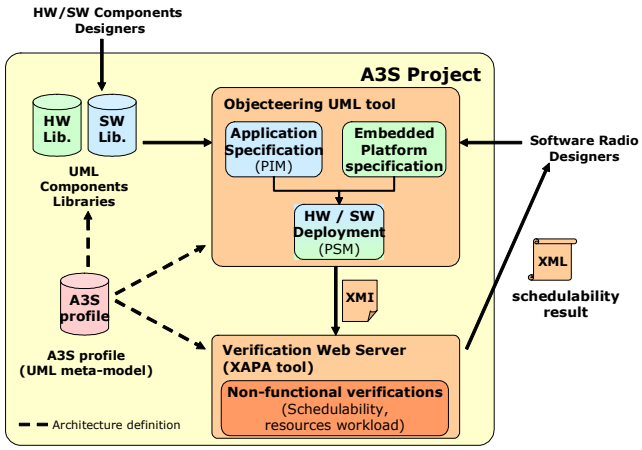

Fig. 1: A3S Project - Design and validation framework for SDR systems

3.1. Three kinds of verifications

The first level of verification deals with components definition (HW and SW) and takes place within the modeling and specification tool. It raises the errors due to the definition of component values (tagged values) that are not authorized by the A3S profile.

The second level of verification concerns the semantic of the application and platform models. It takes place within the modeling and specification tool and raises the errors due to a design violating the modeling rules established by the A3S meta-model.

The last level of verification concerns the non-functional aspects of the system. It raises the errors dues to a correct design but that wouldn't lead to an optimally functioning software radio due to not fulfilled constraints. These verifications take place within the non-functional verification tool.

3.2. Two kinds of verification actors

One important question to be raised is to know who is the final user of the tools. Two kinds of actors can take benefit of the A3S framework:

- **The component designers**: They are concerned by the first kind of verification, since they will create the software and hardware components libraries. For this kind of actors, some ergonomic wizards included in the design tool help them to input the correct values when creating new hardware and software components.

- **The software radio designers**: They are concerned by the second and third kinds of verification, since their goal is to design and tune the software radio platform and waveform. For this kind of actors, others ergonomics wizards help them to instantiate and place the A3S software radio components with conformity to the A3S profile. It is also possible to perform manually several HW/SW deployments in order to reach an optimized solution.

The verifications performed by the tools are related to the A3S profile. They allow the designer to see in a simple

glance the errors within his design during each step of the A3S design flow. It is always possible, in spite of the existence of the GUI, that the designer enters values that are not coherent. Thus, extensive verifications enable a faster and safer design flow.
Some errors can be related to the architecture of a platform, or the connection between the software application and the embedded platform. The designer can perform the verifications for the main points of a design (libraries of hardware components and software components, hardware platform and software application) or for a whole project.

4. A3S PROFILE AUTHORITY

The A3S profile [2] is the key point of the A3S modeling and specification tool and verification engine, in that sense it federates and imposes the rules to be followed for the different verifications. The A3S profile is based on a meta-model and can be considered as an extension of the profile "PIM and PSM for Software Radio Components" [10] defined in the OMG. It ensures the reliability of the works that may be performed using the A3S framework.
The meta-model targets the definition of stereotypes related to an embedded system attributes. Compared to the OMG meta-model it refines the model by introducing all the parameters required to specify and analyze embedded systems containing heterogeneous resources (FPGA, DSP, GPP, FIFO…). It also adds new stereotypes dealing with performance constraints as power consumption and throughput. The OMG meta-model mainly tackles high level aspects of an SDR system whereas our model enables the validation of an embedded platform and corresponding constraints. It also ensures the coherency between the design aspects and the non-functional verifications procedures.
The A3S profile enables the designer to check the following points:

- The designs performed in the A3S framework are conform to the UML formalism.
- The HW and SW components designed by 3rd parties are compliant to a template usable by the A3S tools.
- The platform and application models specification are coherent together, and realize a software radio design that may be verified regarding its non-functional aspects.
- All the elements of the design and verifications tools have some names and types that are conform to the elements fixed by the A3S profile.

Moreover, the A3S profile is able to generate a version of the entire software radio design (including hardware platform and software waveform), that is fully compliant with the notations defined in the recently approved OMG profile "QoS and fault tolerance".

5. UML FORMALISM VERIFICATIONS

All the verifications have been formalized for each step of the design flow. In the following sections we present each level of verification and highlight the help it provides to the designer.

5.1. Libraries

The first level corresponds to the definition of HW and SW components libraries (respectively HWComponentLib and SWComponentLib) and relies on the A3S profile. These libraries correspond to pre-defined components based on a comprehensive model of the HW and SW components. The goal of these pre-defined attributes is to guide the designer in order to fasten the definition of the available components and the related attributes. These attributes have to be filled in order to provide a consistent library of components. All the attributes are used during the verification of the SDR systems and enable the estimation of the achievable performances.
Example of stereotypes for HW components that have been highlighted as part of the A3S profile are the following ones:

- **MaxOperatingFrequency** which corresponds to the maximum frequency for a processor.
- **LogicalCellNumber** which corresponds to the maximum number of available logic cells for an FPGA.

Examples of stereotypes for SW components are the following ones:

- **DataType** which corresponds to the type data used by the SW component (signed / unsigned).

If new components have to be defined the designer has to modify the A3S profile (meta-model) in order to add some new stereotypes. This approach leverages the designer productivity as he does not have to take care of the definition of each component' attributes. The verification engine checks if all the stereotypes respect the A3S profile and if all pre-defined field are fulfilled. At this level almost 20 rules are automatically checked.

5.2. PIM verification

The application is modeled through a functional scheme based on the UML Activity Diagram [1] which is composed of a set of action states (SW component) and transitions. Transitions correspond to dependency relations between functions and have specific parameters related to the exchanged data (number, size, etc.). At that level several verifications are performed in order to guide the designer. The verifications are mainly related to data types and interconnections between SW components. Of course once again the stereotypes defined in the A3S

profile are checked. The verifications can be performed throughout the designer specifies the application which guarantees an UML A3S profile compliant specification.

5.3. Platform verification

The designer builds a platform by assembling hardware components instantiation (in UML sense) through a UML deployment diagram. Many hardware platforms can be realized, especially heterogeneous platforms. The verifications enable the definition of a consistent embedded platform. Examples of verification are related to interconnections between components. Each HW component has specific stereotypes that are checked.

5.4. PSM Co-verification of the SW deployed on the HW

After the software application and hardware platform modeling steps completed, the designer manually chooses which dedicated SW component is implemented onto which HW component. Several verifications are performed in order to check the correct mapping. More than 20 rules have been defined in order to perform the verification. Examples of rules are related to component instantiation, number of input/output ports. If a SW component communicates with another one, there must be a path which connects the associated HW components.

5.5. Ergonomics – wizard – GUI

To provide an intuitive verification tool, the checking preserves the hierarchy of the elements within a project (components, application/platform, deployment) and indicates through a message the possible errors or warnings. Thus, it is easy for the designer to analyze where the problem comes from and to further help him the tool points out the element affected by the error when clicking on an error message. Figure 2 shows the consistency report as it is provided to the designer. As we can see an error is highlighted which enable the designer to correct his specification before going through the non-functional verification tool that analyzed the schedulability of the system.

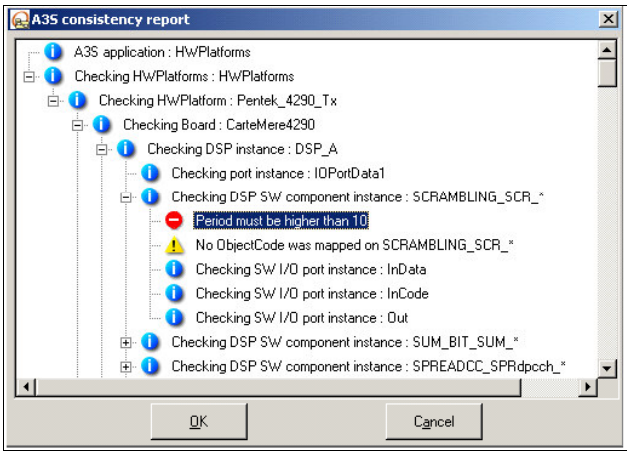

Fig. 2: Checking result of a SW application after deployment

6. SDR-RELATED NON-FUNCTIONAL VERIFICATION

As highlighted in Figure 1, the verifications presented in the previous sections are performed in the Objecteering tool. Then, the non-functional verifications related to performance evaluation are performed outside of Objecteering by XAPAT (XMI A3S Profile Analysis Tool). For that purpose, system information (application, embedded platform, deployment) is extracted from Objecteering in an XMI file format which is then handled by XAPAT through a web service application. XAPAT verifies the adequacy between the application and the architecture and returns scheduling analyses results.

6.1. Multi-processors scheduling extraction

6.1.1. PIM/tasks graph translation

To check if a scheduling solution exists, we first build a tasks graph for the application from the activity diagrams by parsing the XMI file (which is a comprehensive representation of the UML models). This tasks graph is incomplete because it only contains PIM information concerning the application. Thus, it needs to be extended with implementation information (hardware implementation and software component choice). At the PIM level, the tasks are not identified yet, we just know which tasks communicate with each other and the characteristic of data/control signals exchanges. The goal of the tasks graph is to identify input/output and intermediate tasks, to locate where are convergences and divergences within the application, and the direction of data/control signals. The graph is created in Java language, so tasks correspond to objects corresponding to A3SobjectCode UML elements which compose UML Activity Diagram. The graph is also an object which is composed of A3STask objects and Connection objects. A3Stask objects have attributes relative to PIM and PSM characteristics and contain A3STransition objects specific to connection properties. As we only know PIM characteristics all A3Stask attributes are not yet defined and will be refined in the next steps. The PIM/tasks graph

translation permits to provide mandatory parameters needed to scheduling computation.

6.1.2. Platform characteristics extraction

Platform characteristics extraction is performed in the same way as the information extraction for the graph, we parse the XMI file. Objects are created (A3SHWComponent, A3SHWConnect) to provide architectural characteristics and functional information about the components and the platform. We obtain the required information from the XMI file part relative to UML Deployment Diagram components. The definition of the platform permits the identification of which hardware components are connected to each others and through which HW communication components. Java objects are created to model the platform. The number of specific CommEquipmentEquipement (FIFO) is also computed for each connection between two hardware components since this information must be taken into account when computing the overhead due to communications between heterogeneous tasks.

6.1.3. Extrapolation tasks graph/PSM

Once the tasks graph has been constructed, the A3Stask objects identified, and the platform information built, the extrapolation through the PSM can be performed. It is done by finding in the XMI file on which hardware component a given task was implemented. The A3STask attributes are then fulfilled with the implementation constraints.

6.2. Timing verification

Once all the information is retrieved from the XMI file a scheduling can be computed to analyze the performance of the solution. Execution time, period, data size exchanged, platform and implementation constraints are known, which permits to verify and compute the scheduling of the tasks onto the hardware components. The scheduling is obtained using a deterministic algorithm (exact analyze) [1]. If a schedule exists, the DSP and FPGA workloads are computed. A Gantt diagram is provided to enable designers to analyze the quality of the solution (Figure 3).

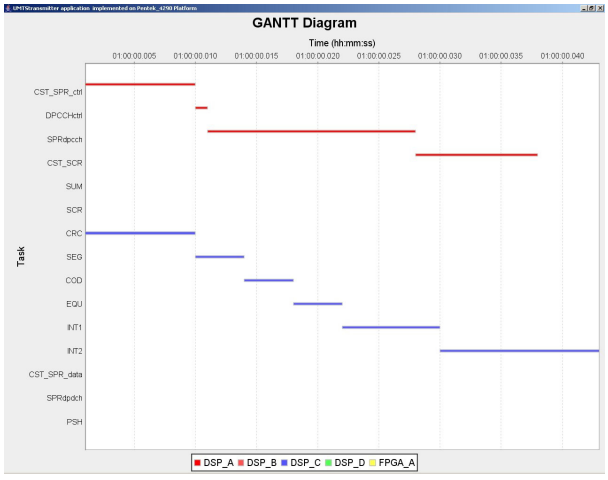

Fig. 3: Gantt diagram provided to enable the analysis of the resources workloads

6.3. Resource selection verification

All the IPs characteristics corresponding to the tasks modeled in Objecteering are known when starting the analysis within the A3S framework. Thus once the Radio system has been scheduled the tool is able to provide estimations about memories size, FPGAs use rate... Communication links usage is also computed to provide designers with performance of the whole system.

7. RESULTS

An uplink UMTS FDD transmission link has been chosen to validate the A3S framework. The UMTS transmitter and receiver applications have been modeled through two different activity diagrams. To prototype the UMTS FDD transmitter and receiver we have considered an hardware platform (Pentek board 4292) composed of four TMS320 C6203 DSP running at 300Mhz. Each DSP is connected to a XILINX Virtex XC2V3000 FPGA running at 100Mhz. Each DSP is also connected to an external shared SDRAM memory. The partitioning is determined manually by the designer. The two UMTS FDD software applications, transmitter (SW 1) and receiver (SW 2) are implemented into the HW platform described above. SW1 is composed of 20 functions (pulse shaping, scrambling, coding, spreading, integrating, etc.), one of them is repeated 4 times and 13 are repeated 15 times. It corresponds to 205 executions of the SW components for the whole transmitter. SW2 is composed of 18 functions (matched_filter, rake, descrambling, despreading, decoding, etc.), one of them is repeated 4 times and 11 are repeated 15 times. It corresponds to 175 executions of SW components for the whole receiver.

Non-functional attributes are first checked to verify the system coherency. The total number of rules that have been defined to check the coherency of the system is around forty which leads to more than 2000 verifications. Different deployments have been considered to verify and validate the efficiency of the A3S framework. The first experience consists of implementing all the functions for SW 1 and SW 2 into the DSPs (software solution), and then in modifying the data rate frequencies to see the limits of such a solution. The second experience consists

of partitioning the functions implementation between DSPs (DSP_A, DSP_C) and their respective associated FPGAs (FPGA_A, FPGA_C). The critical functions within each application are implemented into the FPGAs and the remainder into the DSPs. For both experiences, the two different data rates (117 kbits/s and 950 kbits/s) are tested. The UMTS design under consideration is not a complete fully realistic UMTS system but comprise enough processing element to evaluate the tool. For each experience, a possible scheduling was determined and the corresponding HW components workloads were computed.
The results for one radio frame are given in table 1.

	data rate 117 kbits			data rate 950 kbits		
	DSP_A	DSP_C	time (ms)	DSP_A	DSP_C	time (ms)
tansmitter (SW 1)						
1st experience (all DSP)	96,6%	3,4%	9,99	96,6%	5,1%	10,33
2nd experience (DSP + FPGA)	11,4%	3,4%	7,96	11,4%	5,1%	8,29
receiver (SW 2)						
1st experience (all DSP)	185,5%	4,6%	19,27	185,5%	5,0%	19,33
2nd experience (DSP + FPGA)	17,1%	4,6%	9,44	17,2%	5,0%	9,49

Table I : Hardware component workload

An overall 100% means that all the processing power of all HW devices is necessary to run the application in real time. Less than 100% means that real-time is also reached. In the UMTS standard, a radio frame must be computed every 10ms. In this first experience, we only consider the execution time issue. It shows that software-only solution is adequate for SW 1 as the rate is correct (<100%) for the two configurations (117Kbits/950Kbits). Actually this solution is not correct for the 950Kbits configuration since the timing constraint is not respected as the execution time exceeds 10ms (10.33>10). In the case of SW 2, the software-only solution cannot be realized because of the DSP overload (185%). Thus to respect both the DSP load and the timing constraint we have to define a new implementation.
The result analysis helps to identify function and/or data exchanges that affect the global system performance. Changing the implementation is straightforward with the A3S tool since it just requires to modify some links (corresponding to critical functions) and not to rebuild the whole system. Thus only two critical functions that were previously implemented onto the DSPs are implemented onto the FPGAs (hardware solution corresponding to the 2nd experience). The remainder functions are still implemented onto the same DSPs. The new results show that for each case (SW 1, SW 2), the DSP workload was reduced (e.g. from 96% to 11% for SW 1 and from 185% to 17% for SW 2). This implementation also reduces the execution time, and the timing constraint (<10ms) is respected in each case. Thanks to the tool, the designer performs a fast analysis and is able to compare the most appropriate implementations satisfying the application and architecture constraints.

8. CONCLUSION

The technical description covered by this paper clearly demonstrates the reliability and the efficiency of the A3S tool for the design of software radios in UML and the verification of their feasibility regarding their non functional aspects. The A3S tool is quite an end-to-end product, providing ergonomic assistance in software radio design coupled with a deep analysis of performances at design time. Such a tool drastically reduces the software radio design time cost that could be quite reduced to zero for the prototyping phase, if A3S would be completed with a code generation module.

9. ACKNOWLEDGMENTS

Authors would like to acknowledge the French RNRT funding program that supports A3S project.

10. REFERENCES

[1] A. Delautre, J-E. Goubard, G. Gogniat, C. Moy, N. Bulteau, "**Verification of System coherency at early Architecture Design Stage**", SDR Forum contribution to HAL RFI, Mainz, Germany, April 2004

[2] C. Moy, M. Raulet, S. Rouxel, J-P. Diguet, G. Gogniat, P. Desfray, N. Bulteau, J-E. Goubard, Y. Denef, "**UML Profile for Waveform SPS Abstraction**", SDR Forum Technical Conference, Phoenix, Arizona, USA, November 2004

[3] "**Code Generation for SCA Components**", white paper, http://www.zeligsoft.com/, 2005.

[4] "**SCA Deployment Management**", White paper, http://www.zeligsoft.com/, 2005.

[5] P.Boulet, J-L.Dekeyser, C.Dumoulin and P.Marquet. "**MDA for soc design, intensive signal processing experiment**". FDL'03, Frankfurt, September 2003. ECSI.

[6] E.Riccobene, P. Scandurra, A. Rosti, S. Bocchio, "**A SoC Design Methodology Involving a UML 2.0 Profile for SystemC**", Design, Automation & Test in Eur. Conf. (DATE), 2005.

[7] B. Steinbach, Ch. Dorotska, D. Fröhlich "**Hardware Synthesis of UML-Models**". Workshop on UML for System-on-Chip Design (UML-SOC'05) at Design Automation Conference (DAC'05), Anaheim, USA

[8] M.Oliveira, L.Brisolara, L.Carro, F.Wagner, "**Embedded SW Design Exploration Using UML-based Estimation Tools**", Workshop on UML for System-on-Chip Design (UML-SOC'05) at Design Automation Conference (DAC'05), Anaheim, USA

[9] A.Viehl, O. Bringmann, W.Rosenstiel, "**Performance Analysis of Sequence Diagrams for SoC Design**", Workshop on UML for System-on-Chip Design (UML-SOC'05) at Design Automation Conference (DAC'05), Anaheim, USA

[10] "**PIM and PSM for Software Radio Components**" Final Adopted Specification, dtc/04-05-04, http://www.omg.org/docs/dtc/04-05-04.pdf