UML PROFILE FOR WAVEFORM SPS ABSTRACTION

Christophe MOY – Ph.D., Mickaël RAULET
(Mitsubishi Electric ITE-TCL, Rennes, France - <moy, raulet>@tcl.ite.mee.com)
Samuel ROUXEL, Jean-Philippe DIGUET – Ph.D., Guy GOGNIAT – Ph.D.
(Lester, Université de Bretagne Sud, Lorient, France - <samuel.rouxel, jean-philippe.diguet, guy.gogniat>@univ-ubs.fr)
Philippe DESFRAY, Nicolas BULTEAU
(Softeam, Rennes, France – <philippe.desfray, nicolas.bulteau>@softeam.fr)
Jean-Etienne GOUBARD, Yann DENEF
(Thales Communications, Colombes, France - <jean-etienne.goubard, yann.denef>@fr.thalesgroup.com)

ABSTRACT

This paper proposes to address the issue of a priori verifying, at the architectural design phase of both software and hardware, the adequate operation of a software radio protocol stack application on a hardware platform in terms of real-time execution, power consumption, memory size, circuit surface (gate), communication media dimensioning... A methodological approach is given for describing in UML both hardware platforms made of DSP, FPGA, GPP, FIFO, buses, etc. and software applications for radio physical layers (SPS - Signal Processing Subsystem) as well as higher layers. This gives the designer the possibility, right at the UML modeling step, to investigate the array of potential solutions in order to select some verifying the coherency.
This operation will be performed by verification engines. The aim is also to provide to the community an UML profile that includes all the non-functional characteristics of the hardware processing and communication devices on the one hand, and those associated to the software components on the other hand. The methodology should accelerate the SDR design process by pre-dimensioning the hardware platform before its effective prototyping.

1. INTRODUCTION

RNRT is the French Ministry of Research and Industry funding program on telecommunications. A3S responds to the priorities of the RNRT 2002 call for proposal and aims at realizing a tight coupling between specification and system design constraints to enable a priori verification of the software (SW) architecture adequacy with the hardware (HW) platform for software radio. Its goals are to promote the tools and development environment enabling:

- Virtual representation of heterogeneous reconfigurable systems (DSP, GPP, Logical devices),
- High level system design sustained by languages able to address both technologies software and hardware,
- Tight coupling with development environments.

A3S (Adequacy between Algorithm and System Application), aims at performing non-functional coherence pre-verification of software radio HW architecture specifications and SW application requirements with UML (Unified Modeling Language) based models. Non-functional aspects in the scope of A3S concern execution time, memory size, FPGA gates and communication media. This set of parameters, in association with the description graphs of the platform and the SPS (Signal Processing Subsystem) application, permit to verify if the coherence and performance constraints of the software radio system are respected, depending on the mapping of SW functions on HW components.

This work aims also at contributing at a large scale to the SDR community by promoting SDR design through sharing UML profiles and specification methodology. It is also in the scope of A3S to make its tools compliant with the Software Communication Architecture (SCA) and to take into account the elements specified in the existing Object Management Group (OMG) standardized profiles.

2. UML FOR SDR DESIGN

High level tools dedicated to system design are of great interest as they guide the designer in his tasks thankfully to some automations. The concept of a priori system coherency verification tries to ease and accelerate even more this phase. An effective solution is to use a well-established standard as UML, whose concept have already been proven on many project designs, and is understood by a huge community of designers and developers. So the gain is twice: sparing time for modeling language control, and for design time (due to the tools automations). Moreover UML is a formal language enabling refinement of the description of the system up to code generation. Main tools provide code generation for C/C++, Java and also some dedicated language through bridges or plug-in like ESTEREL [1].

2.1. UML profile for Software Radio

UML profiles enable to specialize UML for each work context by introducing some notions adapted to the field of application. An UML profile regroups coherently the extensions of the UML model and defines their coherency rules. There may be dependencies, inheritances, or groupings between UML profiles, the main interest being the reuse of domain specific notions in a standard way. Some standards profiles are emerging, each of those being an UML profile dedicated to an application domain or a technical environment.

The UML profile for software radio [2] is in elaboration at the OMG. It shall allow the description of the technology used in software radio. It specifies the interfaces between the waveform components of an SPS and an environment constituted of radio devices (amplifier, antenna), radio services (filters, converters), radio management components (channels assignation) and operating system.

2.2. UML profile for Real time, Scheduling and Performance

The Real time, Scheduling and Performances profile [3] which describes the characteristics, is an OMG standard focalized on the representation of properties bound to time, like the duration, the performance and the planning. It allows to apply many approaches of the real time analysis, by allowing them to express the essential attributes for the system definition and the analysis of their performances.

The goal of this profile is to allow the description of these temporal properties and to be able to predict the temporal aspects of the software before the implementation. The annotations give some elements linked to the quality of service like the deadlines or the priorities. The intention is to allow the exploitation of these detailed indications on the model in order to apply the analysis tools that may detect incoherence and provide performances predictions.

2.3. UML profile QoS and Fault tolerance

The QoS and Fault tolerance profile [4] describes the elements allowing to bring non functional information on a system UML model in terms of quality of service, as well as UML framework elements allowing to compare and validate them. The QoS tackles some notions allowing to answer to the static aspects of the quality of service (that is: fixed at design time) as far as its dynamic aspects (in the case of modifications of the QoS requirements during the program life). In a SDR context, the dynamic evolution of the QoS may be linked to reconfiguration operations of the system functionality as illustrated in [5] for instance.

2.4. Model Driven Architecture (MDA)

MDA allows to integrate all the middleware technologies (such as CORBA, EJB, XML, SOAP, .NET), the languages and the type of applications, federating them around the application model. The MDA principle consists on defining some domain adapted models, independent of the implementation technology, called PIM (Platform Independent Model), and to transform these models in some more specialized models closer to the technology (PSM = Platform Specific Models) until being able to produce automatically the final code. The Software Radio profile of the OMG is looking forward this way in order to specify some stereotypes that may be used to produce software radio PIM and PSM models. One of the goals of A3S is to apply non-functional description elements on PSMs and on PIMs and to confront them during the verification phase.

3. A3S PROFILE

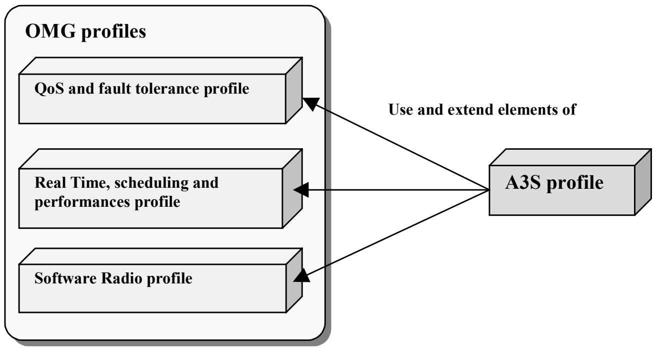
Currently, major software radio projects are already described in UML class diagrams for architecture and sequence diagrams for chaining behavior. The UML profile for software radio proposes a set of stereotypes that allow the description of platform independent or dependent architectures of radio systems on a functional view side which is relatively close to the SCA (Software Communication Architecture) specification. To allow the fine definition of SPS behaviors, it can be extended by the addition of stereotypes coming from QoS profile and Real Time profile on each of the components addressed by the Software Radio profile, describing their quality of service behavior offered or desired. In order to address the DSP/FPGA/ASIC (devices usually present in SDR designs) specific domain of study, it is possible to extend the software radio profile by introducing DSP and FPGA stereotypes derived from the processor stereotype of the software radio profile. These new stereotypes will be then tagged with stereotype extracted from the QoS profile to describe the quality of service behavior of the DSP and the FPGA.

3.1. Legacy components

The use of standardized profiles and the components they introduce, allows the designer to reuse some legacy components by wrapping them into a standard component, exhibiting the compliant interfaces. The designer focuses on architecture or system composition, instead of being compelled to discover and/or create new components from scratch. This method has already been used for a long time by software developers to de-couple provided components from third-party. Such an approach is the only way to enable a smooth transition from existing methods to new one. It also makes possible the integration of non-compliant external provided component.

3.2. A3S profile and OMG profiles

The A3S profile defines some elements that will be used to build the software radio architecture models that may be verified by the A3S tool. These elements extend or use some elements extracted from the previously explained OMG standard profiles, as illustrated in figure 1.



OMG profiles
QoS and fault tolerance profile
Real Time, scheduling and performances profile
Software Radio profile
Use and extend elements of
A3S profile

figure 1: Relation between A3S profile and the OMG standard profiles

This method warranties the durability, the interchange and the reusability of the A3S models. Since the interfaces can be standardized this way, it is then possible to work and verify any A3S models assuming that the tools integrate the A3S profile. This A3S profile main interest resides in the fact that all the interfaces may standardized, and that all the elements are redefined from the basic types, warranting an automatically generation of interface specification through the IDL (Interface Description Language) syntax language.

3.3. UML Meta-model for A3S

The meta-model notion is a formalism allowing the description of the models. In the scope of A3S, a meta-model has been defined, illustrated by four main views. The first one characterizes the A3S project architecture, and the associated main packages. The second one represents the software part of the elements definition. The third one illustrates the mapping of the software components on the hardware components. The last one defines the needed elements to build the HW platform. Links and dependencies between each HW component that can be instantiated during the modeling phase are depicted in figure 2. A3S meta-model is connected to OMG profile for Software Radio through the gray boxes at the top of the figure.

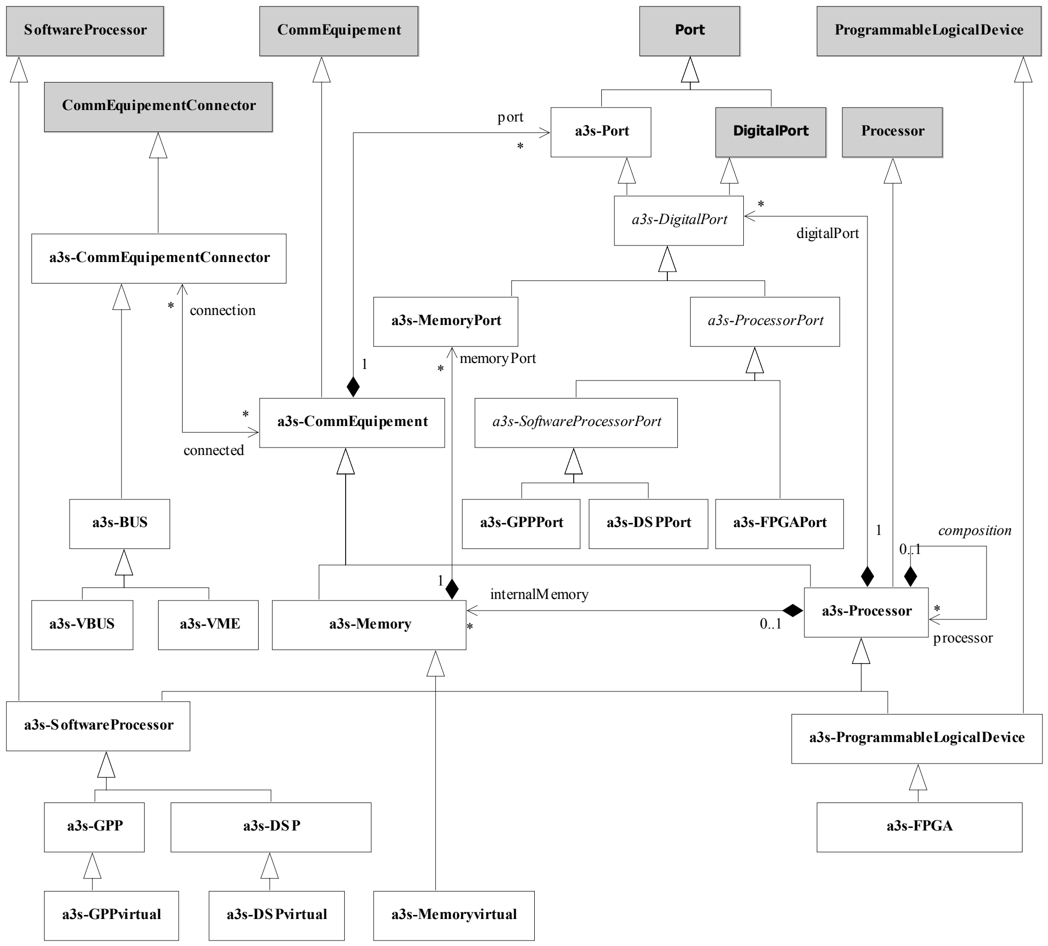


figure 2 : A3S meta-model for HW components

4. SOFTWARE RADIO DESIGN

4.1. HW and SW description

A3S aims at adding to the design flow new considerations that make the gateway between functional simulation and implementation on a hardware embedded platform. Concretely, this consists in adding to each of the algorithms that have been functionally checked, non-functional characteristics depending on the hardware target it will be supposed to run on. Note that several solutions of implementation (DSP, FPGA, ASIC) can be considered for each algorithm. As algorithms are built in software components they can be easily moved in the multi-processing architecture. This consideration can be extended to the communication media nature (bus, FIFO, TCP links, shared memories…).

Concerning the software components, whatever the nature of the hardware device that will execute the algorithm, some non functional parameters are necessary like I/O data type or repetition of the algorithm, even if some characteristics are more or less important regarding the device on which the algorithm will be executed.

Concerning the hardware components, different characteristics will be considered regarding the HW device we use. For example a DSP will be described at least by the clock frequency, code memory size, data memory size, co-processor number and nature, I/O ports number and nature, existence of a DMA..., but an FPGA will be described at least by its gate availability, internal dedicated RAM block availability, internal dedicated multipliers block availability, and clock frequency. We also address communication means (and external memory) like FIFO, bus, dual-port memory, DMA. Middlewares of the platform are a part of the system itself, and may have influence on the system performance.

4.2. UML representation

A two diagrams form representation has been selected to specify the application and the platform. This couple can support the entire description of both software and hardware characteristics in an integrated and clear manner. This could be rapidly described as a hardware graph that deals with the platform and a software graph dealing with the application.

The hardware graph, which is an UML 1.4 deployment diagram, describes the physical connections that exist between the hardware devices located on the platform. The example of figure 3 represents a processing node composed of a DSP, connected through its ports to different kinds of memory chips with a Vbus. Connections between the processing node and the rest of the system are also visible here.

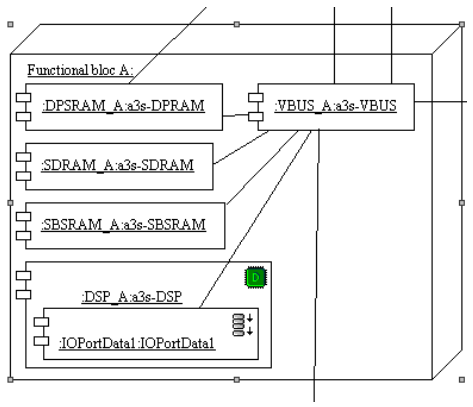


figure 3 : Deployment diagram for a processing node.

The software graph is an UML 1.4 activity diagram as shown on figure 4, which describes UMTS baseband processing operations at the transmission.

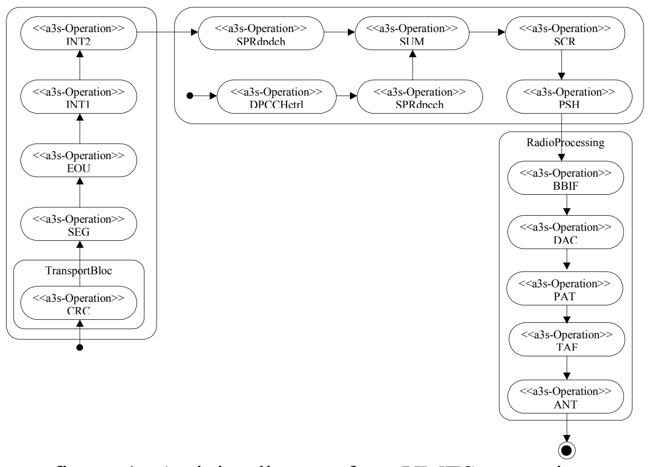


figure 4 : Activity diagram for a UMTS transmitter

It addresses the logical links between the different SW components that constitute the system radio functionalities. It includes the non-functional characteristics that are independent of the nature of the hardware device that will execute the application (for example the number of I/O of a software component, its period and its number of execution).

4.3. Deployment of the SW application on the HW platform

The targeted hardware devices for a software component are defined in a table where each instance of the software component of the activity diagram is described. This allows to fill the parameters which are dependent of the hardware devices concerning each software component. Moreover, this approach enables to highlight on the deployment diagram the repartition of the software components on the hardware platform as illustrated on figure 5 which updates figure 3 including the software components instances.

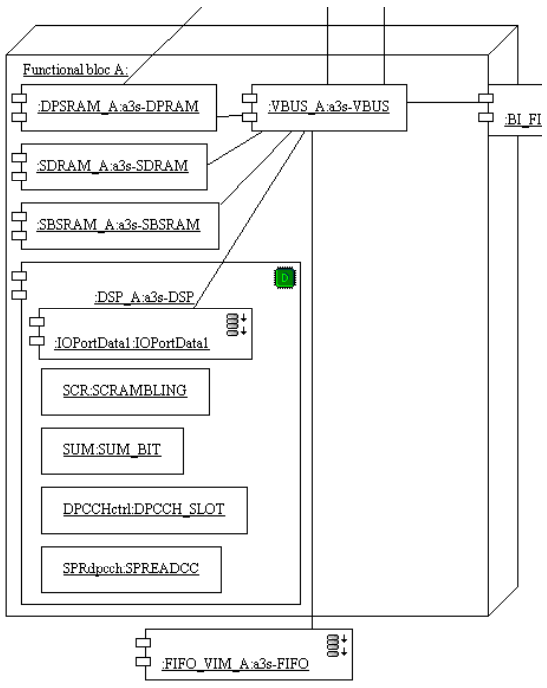


figure 5 : Activity diagram for a UMTS transmitter

At each step of the design flow: application specification, platform specification and hardware-software mapping the designer needs to verify the coherency and the performance of his solution. This information is provided through a constraints composition approach.

5. VALIDATION PROCESS

Software application and hardware specification as well as mapping software into hardware are performed within the Objecteering UML software of Softeam company. Non-functional verification is then performed with an external tool that exchanges information with Objecteering UML software through a XMI interface.

5.1. XMI interface

Software/hardware architecture and parameter values are stored in Objecteering UML software as a XMI format, that is a structural system representation of the graphical system view. Attribute values of software/hardware components and activity diagram from the XMI file are analyzed to build a General Task Graph (GTG). The method to extract information from the XMI file is based on the JDOM API that parses the file [6]. The extraction rule only consists in exploring the branches of a tree.
The GTG is used to obtain an efficient scheduling of the application tasks. Two main steps are performed in order to obtain the performance of the entire system. First, a period derivation algorithm like [7] permits to determine all the tasks period which compose the GTG. Then, a static scheduling of tasks is determined thanks for example to a Rate Monotonic Scheduling algorithm [8]. Once the tasks are scheduled performance results are computed and provided back to the Objecteering UML software as shown in figure 6.

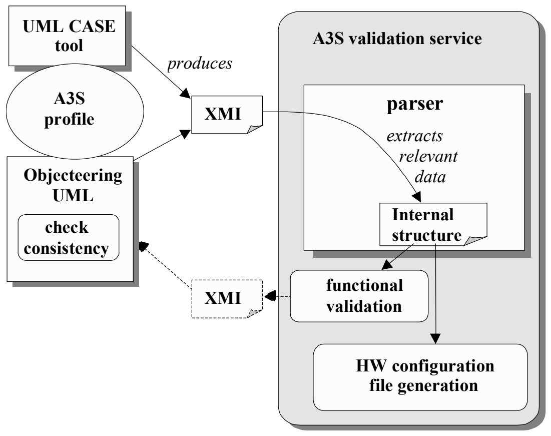


figure 6 : Validation process

5.2. SW architecture validation

The software architecture modeling, which consists in determining and connecting software components, requires a phase of analysis to validate the representation under specification. It is thus necessary to detect structural errors and provide them to the designer. Thus, a certain number of constraints of coherence are analyzed. For instance, it is necessary that, on both sides of inter-connected SW components, the interface as well as the type of the exchanged data are identical (throughput, data-width).

Modeling must also provide answers in terms of feasibility. It is possible to check if the model of execution and the coherence of the periods of the various SW components are correct from the attributes associated to each software component. For instance the detection of an overlapping between interdependent software components is performed. Some components can also execute an operation depending on a signal coming from another component, which results in constraint of synchronization that must also be taken into account. Hence, the verifications done at the software architecture level are related to specification and execution model coherency.

5.3. HW architecture validation

The modeling method for the platform is very similar to the application one. The difference is that instead on SW components, the representation is based on HW components. Thus, the same kinds of verification are required. Actually, structural constraints of the system are targeted in order to check the platform coherency. Hence, the coherency of types and connections must be validated. The architecture coherency is the first step of a global verification process for which an automatic tool efficiently assists the designer.

5.4. HW/SW mapping verification

The verification corresponds to the global execution time of the application, which is the combination of the execution time [7] [9] of each software component running on its corresponding hardware component. This step takes into account the constraints of operation, communication, memory, OS and middleware. Furthermore, in a complex system where several functions can be carried out in parallel in a heterogeneous environment, it is important to provide information on the resources used. Indeed, if the system does not respect the constraints, the bottlenecks identification (overuse of a resource whereas others are available) is important in order to find solutions which may improve the system. The addition of traces goes in this direction. The traces give the temporal evolution of a certain number of selected parameters. It is thus possible to see the temporal resources occupation, the functions activation and duration.

Functions are also subjected to scheduling constraints which depend on the architecture (interconnection and computation resources) and on the performances that the designer wants to obtain. They are also related to the constraints of data dependencies. This is also in the scope of the tool verification toolbox.

All these verifications enable the designer to validate early in the design cycle his hardware-software mapping. If constraints are not met, another mapping can be tested or a new platform defined in order to converge rapidly to an efficient solution. The unified UML HW/SW component modeling and mapping provide a framework where both platform and application can easily be modified without a tedious and error prone design effort.

5.5. Web-service validation

The A3S validation of a model built with the A3S profile is intended to be performed through a web-service portal. A local or distant server, accepting models compliant to the A3S profile, will parse and process designs in order to extract the information that will be used to perform the validation. So as to warranty the portability and the exchange of the model, one of the envisaged solution is to select XMI as output model format of the UML design tool since XMI is the standard format for UML model interchange. This XMI file will be the input of the A3S validation web service, which extracts all parameters needed by the verification engine.

6. CONCLUSION

A3S provides a design methodology and toolboxes for the modeling and the verification of SDR systems with respect to predefined rules. A computation process evaluates solutions and provides figures to detect design bottlenecks. This approach does not require a complete description of each hardware and software component. Only their basic features are needed in terms of non-functional behavior. SDR systems costs are drastically reduced while minimizing design time and number of prototypes. Moreover, it also impacts analysis of software or hardware architectural modifications with regards to the functional and non-functional system requirements. A3S graphical description currently follows UML1.4 standard, but already anticipated UML 2.0 evolutions. This transition has been considered since the very beginning of A3S project.

A3S supports SCA architecture through its compliance with the OMG profile for software radio, and A3S consortium made a proposal to the SDR Forum hardware abstraction layer request for information.

7. ACKNOWLEDGMENTS

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

8. REFERENCES

[1] http://www-sop.inria.fr/meije/esterel/esterel-eng.html
[2] UML profile for Software radio – OMG draft.
[3] UML profile for Schedulability, Performance, and Time Specification – ptc/02-03-02 OMG Adopted Specification.
[4] UML profile for Quality of Service and Fault Tolerance Characteristics and mechanisms – OMG revised submission
[5] C. Moy, A. Kountouris, A. Bisiaux, "HW and SW Architectures for Over-The-Air Dynamic Re-configuration by Software Download", *Software Defined Radio workshop of RAWCON'03*, August 2003, Boston, USA
[6] Processing XML with JAVA : "A guide to SAX, DOM, JDOM, JAXP and TrAX", *Elliotte Rusty Harold. Addison Wesley*, 8 nov 2002.
[7] A. Dasdan et. Al ."Rate Derivation and Its Applications to Reactive, Real-time Embedded Systems", *ACM/IEEE*, 1998
[8] Christian Bonnet, Isabelle Demeure, "Introduction aux systèmes temps réel", *Hermès Sciences*, 1999
[9] Richard Gerber, Seongsoo Hong, Manas Saksena "Guaranteeing Real-time Requirements with Resource-Based Calibration of Periodic Processes*", IEEE transactions on software Engeneering*, July 1995.