Title of research item

UML profiles towards waveform performances verification

Contact details of author/submitter

- Antoine Delautre (Thales, Colombes, France, antoine.delautre@fr.thalesgroup.com),
- Jean-Etienne Goubard (Thales, jean-etienne.goubard@fr.thalesgroup.com)
- Guy Gogniat, Samuel Rouxel, Jean-Philippe Diguet (Lester Université Bretagne Sud, Lorient, France, guy.gogniat@univ-ubs.fr)
- Christophe Moy (Mitsubishi Electric ITE-TCL, Rennes, France, moy@tcl.ite.mee.com)
- Nicolas Bulteau (Softeam, Rennes, France, nicolas.bulteau@softeam.fr)

Subject area: **WG6: Reconfigurability**

Relevance of the topic to the above subject area

In the scope of the RNRT A3S project, we are addressing the issue of a priori verifying right at the architectural design stage, the adequacy of waveform applications deployment on a hardware platform through the verification of functional requirement and architectural features (timing, latency, bandwidth, …). Even if this issue is transversal to various domains, we are specifically addressing the software radio domain. The result of such a project will be a set of UML tools being able to predict by comparing the UML models if a waveform will execute properly on a specific platform in terms of performances, those tools will be implemented in a UML toolbox for demonstration and validation purposes.

ABSTRACT

This paper intends to introduce a new methodology for software radio systems validation, entirely based on UML. Right at the modeling step it will be possible to perform non-functional coherence verification of software radio architecture specifications and application requirements with UML based models. The interest of such an approach is to give the designers the opportunity to investigate, before beginning any development phase, the array of potential solutions and enabling selection of some by verification of the coherency. This approach will enable design cost saving by drastic reduction of time and minimization of the number of prototypes. This paper will mainly focus on the feasibility of such an approach, highlighting the object oriented technologies that may be involved in the validation process (UML profiles, UML diagrams, standard model file description…) as far as the non-functional characteristics validation engine approach.

1 INTRODUCTION

One of the main objectives of software radio is to enable the development of waveform components independently of the hardware on which they will be executed. Even if this allows the functional portability, this does not guaranty at any cost that the waveform components will behave optimally and identically on two different platforms in term of quality of service. One of the main challenges will be to verify during the design phase (and more especially on the different models

described) if a waveform application can be executed on a specific platform in the required level of quality of service: execution time, latency, memory, etc.

In the A3S approach, we highlight the constraints composition part, focusing on a heterogeneous Digital Signal Processor (DSP) and Field Programmable Gate Array (FPGA) based system, and we place our interest in the verification at an early stage. We will thus propose an extension of the software radio profile elaborated by the OMG, allowing to design some software radio platform specific and independent models that take into account non-functional and quality of service aspects of the components behavior.

The paper is arranged as follows: Section 2 reviews existing methods and tools to improve the design cycle.

Section 3 focuses on UML, it highlights the benefits of using UML and its associated profiles to perform the design of software radio systems. Section 4 provides the information related to PSM in the context of MDA technology. In this section a brief presentation of the attributes required to characterize and to verify software radio systems are exposed. The UML models that we propose are also presented. Section 5 deals with constraints verification in order to validate a design at each step of the specification. Section 6 introduces the A3S project and gives its expected contributions and finally section 7 concludes the paper.

2 ACTUAL DESIGN IMPROVEMENT TRENDS

2.1 Performances verification.

Designers have access to plenty of high-level functional simulation tools: C, C++, Matlab Simulink, Synopsys, CoCentric Studio... but almost exclusively processor dedicated. Possibility exists to deal with some kind of heterogeneity, involving FPGA but it is quite limited. For instance, Simulink and Xilinx propose pre-developed equivalent algorithms for both processor and FPGA, which support so that the simulated results are effectively coherent with the implemented version. But this is limited in terms of the variety of algorithms and the hardware targets (components and platforms).

SystemC tries to impose itself as a standard for heterogeneous co-design. But it is only usable for functional co-simulation and not for developing nor for debugging environment. Moreover, synthesis from SystemC is not completely validated. If SystemC is convenient for functional simulation and verification, there is a break in the design flow to go through the implementation on embedded platforms, which are often multi-processing and heterogeneous for software radio. This break implies risks of mistake and the impossibility to re-introduce in the high-level design modifications done at embedded domain level. This impacts in term of development time, design quality and consequently cost.

Other ways have been investigated. The solution to directly work on the embedded system level itself is rejected because of the complexity and non-effective way of working it imposes. The above reasons lead to the conclusion that the high-level CAD tools have to be enhanced to encompass the multi-processing heterogeneous domain at both functional and non-functional level.

Either it will be integrated directly in the simulation environment, or as an independent tool, with possibility of a direct bidirectional bridge with the simulation tool.

2.2 Available tools and their use

Only some academic tools currently try to offer solutions to the issue. Several of them are discussed here.

At a platform level, CoFluent[i] Studio enables developers to capture and verify a fully-timed functional model of their system through graphics and C code, and explore various heterogeneous architectures.

The SoC Environment (SCE)[ii] provides an environment for modeling, synthesis and validation. SCE represents a flow that allows designers to capture system specification as a composition of C-functions. It is based on a set of tools to facilitate the design flow and perform refinement steps.

SynDEx[iii] extracts itself the potential parallelism available in a system, based on measurement or estimation of the algorithms execution and communication timing. It makes scheduling and partitioning for multi-processing platforms involving DSP, GPP and FPGA (currently only as co-processors), and automatically generates the corresponding code for each of the processors (C or assembly for processors, VHDL for FPGA). The tool tries to optimize the processing efficiency and the user checks if it respects its constraints (in term of execution duration here)

At the level of a FPGA, GAUT[iv] generates a pipelined architecture implementing the VHDL-expressed digital signal processing function under timing constraints and a target technology. GAUT extracts the potential parallelism at the behavioral level e.g. before selecting, allocating, assigning, and scheduling hardware operations. It generates a processing unit containing the arithmetic and a control unit piloting the processing unit.

2.3 Return on investment

The concept of the a priori system coherency verification is to ease and accelerate the design phase, a simple solution is to use the most known modeling language in order to facilitate the control of the language. For that reason we will address UML[v] because it is a well-established standard, whose concept was proved on most of project designs, and is understood by a huge community of designers and developers. A tool dedicated to the system design and verification may be of great interest because it may guide the designer in its conception thankfully to some automations. 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 bridge or plug-in like ESTEREL [vi].

3 THE OBJECT ORIENTED CONCEPT

3.1 UML in global projects conception

UML is a very commonly used language in the system design process, but the diagrams it allows to take into account are differently used.

- The Use Case diagrams are frequently used to express the user's need. They describe the actors interacting with a system and the main use cases of these actors on the system.
- The sequence diagrams allow to describe the interaction sequences between many objects.
- The Collaboration diagrams, often used conjointly with objects diagram, allow to describe the objects, their links, values and cooperations.
- The Class diagrams are the most used in UML. They allow to describe the main notions of a system, their links, their attributes and their classifications. Used at all development steps, they allow to obtain some conceptual, architectural or close to code diagrams.
- State and Activities diagrams are used to describe the dynamicity of a system. While State diagrams are mainly used for technical specification of real time applications, Activity diagrams allow to model the process described by activity chains with information transmission, connection management, and activity responsibility description. They are used in tertiary applications, where process description is primordial, but also in the technical and real time application world

3.2 The UML profile: a solution for specifying the models to particular domain

UML profiles enable to specialize UML for each work context by introducing some notions more adapted to the current work. 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 Real time, Scheduling and Performances profile[vii] which describes the characteristics, is an OMG standard, focalized on the representation of properties bound to the time, like the duration, the performance and the planning. 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 any development. Figure 1 shows the meta-model described into the real-time profile for non-functional characteristics representation.

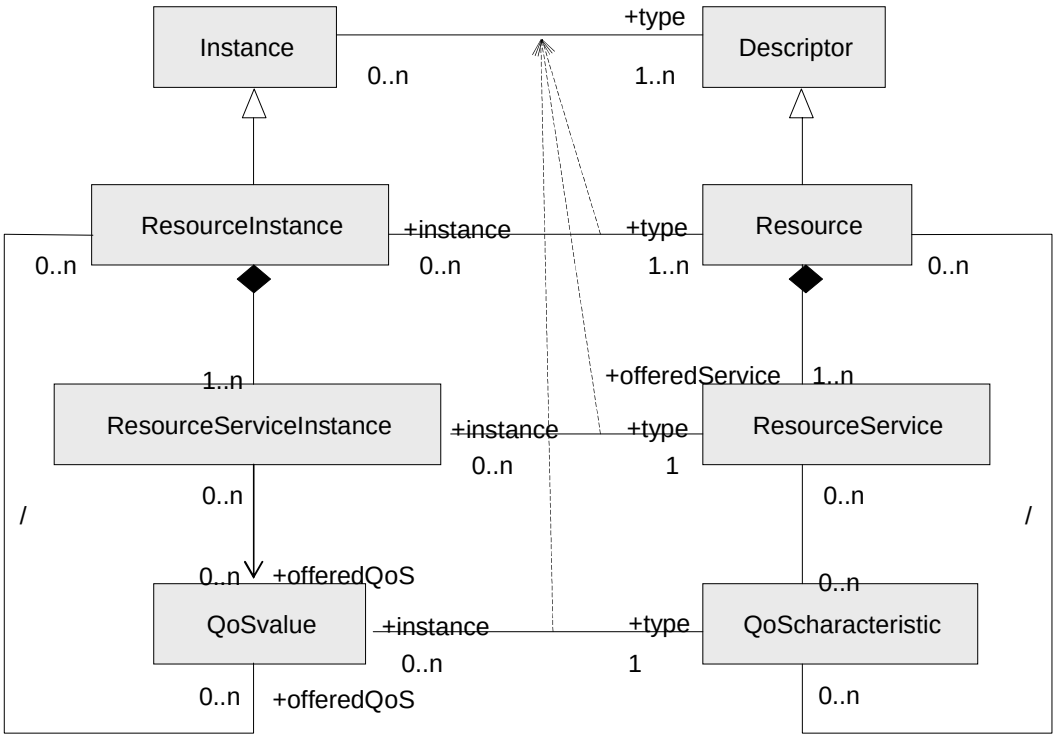


Figure 1: meta-model for non-functional characteristic representation

UML profile for software radio

The UML profile for software radio[viii] is in elaboration inside the OMG, it shall allow the description of the technology used in the software radio of the PIM model form, which specifies the interfaces between the waveform components and an environment constituted of radio devices (amplifier, antenna), radio services (filters, converters), radio management components (channels assignation) and operating system. Figure 2 describes one of the meta-models introduced by the software radio profile.

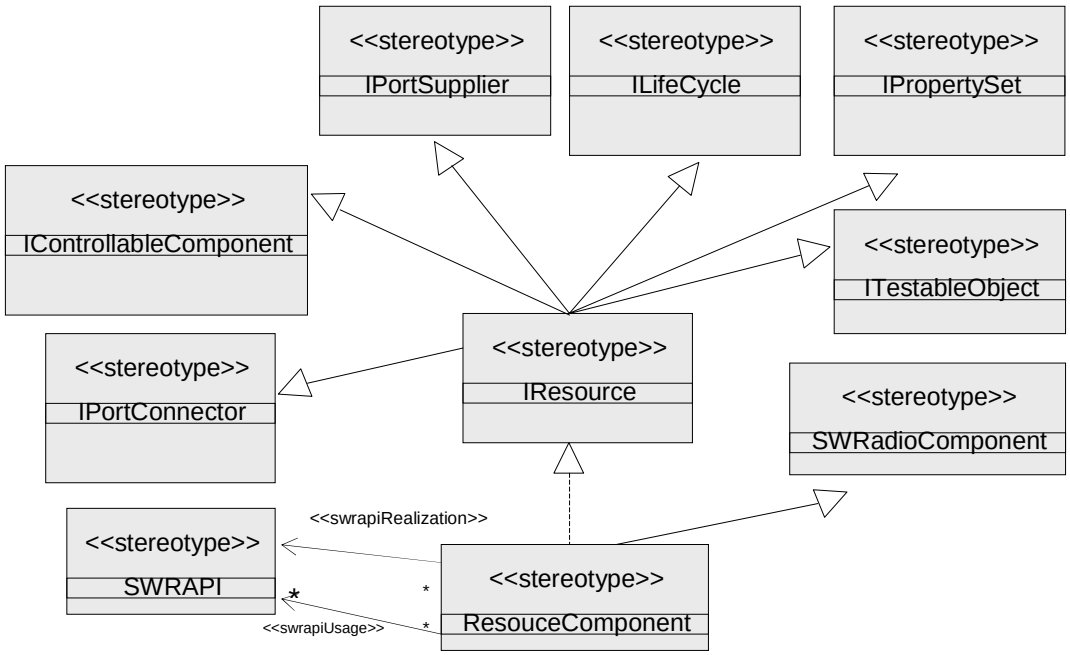
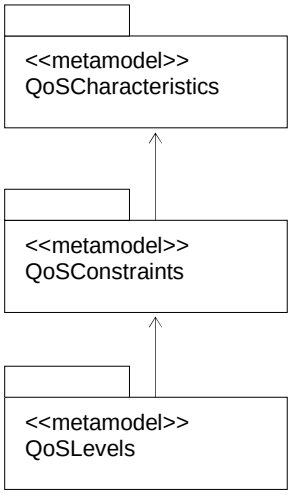


Figure 2: meta-model for software radio resource representation

UML profile for QoS and Fault tolerance



The QoS and Fault tolerance profile describes the elements allowing to bring non functional informations on a system UML model in term of quality of service, and the UML framework element allowing to compare and validate them.

The QoS tackles some notions allowing to answer to the statical aspect of the quality of service (that is: fixed at the conception) as far as its dynamical aspect (in the case where the QoS requirements may change during the program life).

All the notions tackled by the QoS profile framework can be grouped into three different packages linked each other by some dependencies, which are :

- The QoSCharacteristics package
- The QoSConstraints package
- The QoSLevels package

Figure 3 : composition of the QoS profile

The usage of such a framework can be illustrated in the following example : An entity called manager writing data into a memory resource, using a processor resource, as shown in Figure 4. The writing operation requires a worst time execution of 17.0 ms on a i86processor.

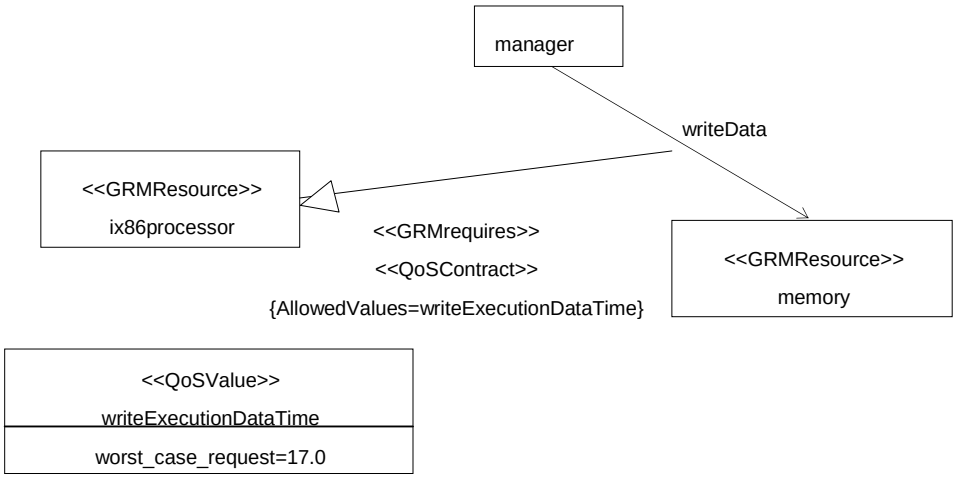


Figure 4 : example of QoS profile element usage.

UML profile for Real Time, Scheduling and Performances

This profile is an OMG standard, focalized on the representation of the temporal properties, like the length, the performance and the scheduling. 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 its 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 to apply the analysis tools that may detect incoherencies and providing performances predictions.

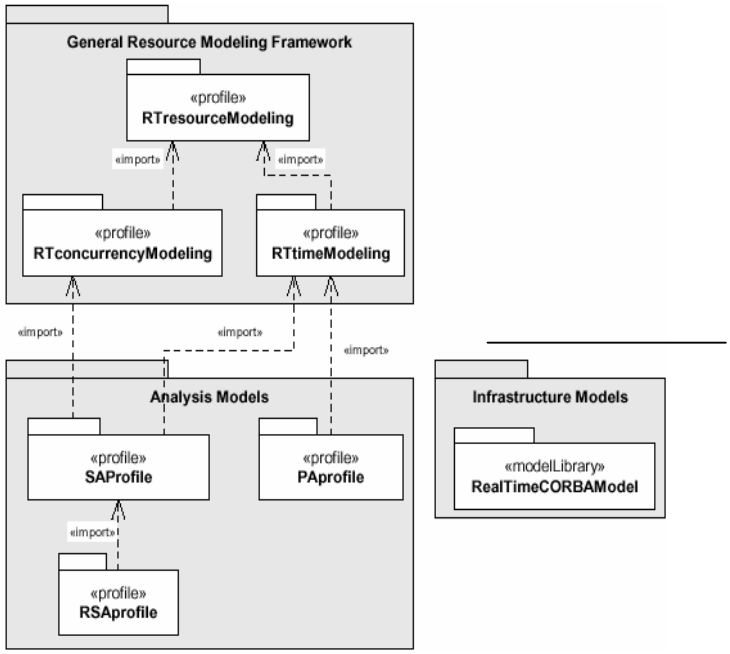


Figure 5 : composition of the SPT profile

This profile is composed of three sub-profiles addressing respectively the descriptions elements, the analysis elements, and the infrastructures models, as show in **Figure 5**.

The Figure 6 illustrates the use of temporal description of and exchange represented by a sequence diagram. In this example an entity called “Manager” send a message to an entity called “Executor” to process an action. Before the end of the action, a message is sent from the executor to the manager, stipulating that the order has been taken into account. Because each behavior has got a length, it is necessary to represent it on the sequence diagram in the case of a real time system. In this example, the sending of the process message, start at t0, and take 1.5 ms to be sent. Once received, the executor starts the action 0.5 ms later; at t0+4.7, a message is emitted by the executor to the manager, this takes 5.5 ms, and the executor’s action ends at t0+11.

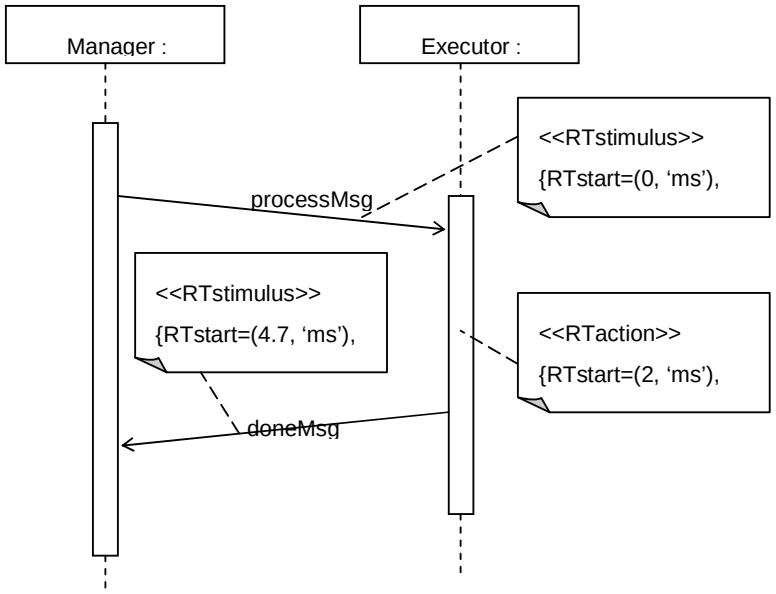


Figure 6 : example of temporal representation in a diagram sequence

The **Figure 7** illustrates the concept of scheduling with two entities, a clock and a manager, the clock telling periodically (every 100 ms) to the manager to gatherData(), the manager has 100 ms to respond to this order. The manager is a process with realtime priority 2, the main function of which takes 33.5 ms to execute, and 93 ms in the worst case.

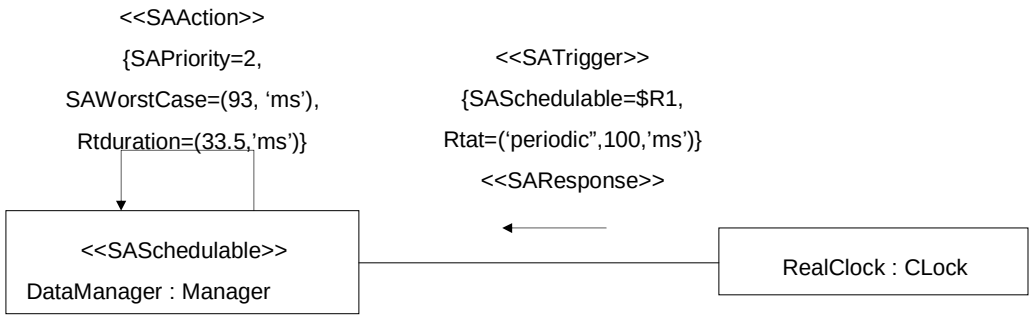


Figure 7 : example of scheduling representation

A3Sprofile:

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 extends or use some elements extracted from the previously explained OMG standard profiles, as illustrated in Figure 8. This warranties the timelessness, the interchange and the reusability of the A3S models. Since the interfaces can be standardized by this way, it is then possible to work and verify any A3S models assuming that the tools has got the A3S profile.

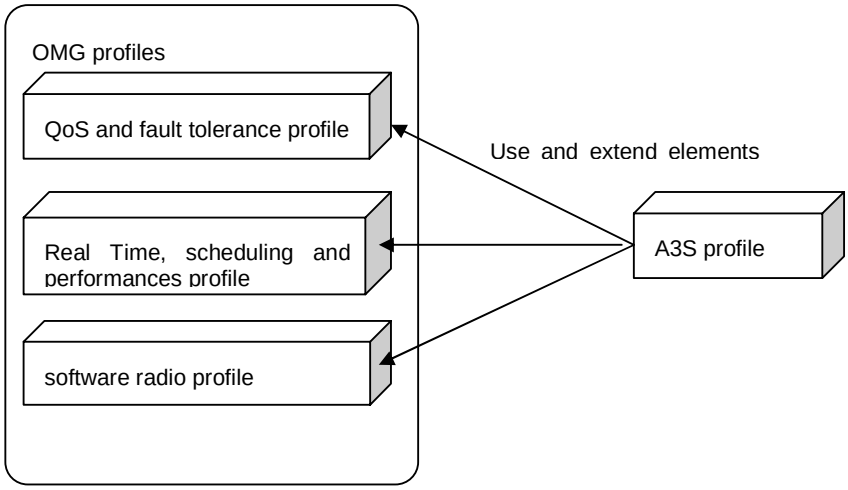


Figure 8 : relation between A3S profile and the OMG standard profiles

This A3S profile's 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 syntax language.

3.3 Current UML place in the software radio

Currently, major software radio projects are described by UML class diagrams for architecture and sequence diagrams for chaining behavior. The UML profile for software radio proposes a set of stereotypes for allowing the description of platform independent or dependent architectures of radio systems on a functional view side which is relatively close to the SCA specification. To allow the fine definition of signal processing behaviors, it can be extended by the additions of stereotypes coming from QoS profile[ix] 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 specific domain of study, it can be possible to extend the software radio profile by introducing DSP and FPGA stereotypes representing DSP and FPGA components 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. In the software radio profile and real time, performance and scheduling profile, some aspects are described in a CCM (Corba Component Model) view, allowing to perform Corba [x] implementations.

3.4 Legacy Component

Using standardized profiles and the components they introduce, we will allow the designer to reuse some legacy components by wrapping them into a standard component exhibiting the compliant interfaces. It will make the designer possible to focus on the architecture or system composition, instead of being compelled to discover and/or create new components from scratch. This method is

already used for a long time by software developers to de-couple from third-party provided components. 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.

4 UML MODEL BASED OF VERIFICATION OF COHERENCE ARCHITECTURE - APPLICATIONS, THE CONCEPT

4.1 The MDA technology in software radio design

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.

4.2 The A3S validation service, an object approach

One of the main goal of A3S is the validation of a model built with the A3S profile. This can be performed through a web-service concept, that is to say a local or distant server, accepting models compliant to the A3S profile, and parsing and processing them in order to extract the informations that will be used to perform the validation. In order to warranty the portability and the exchange of the model, one of the envisaged solution will be to specify that the output model format of the UML design tool will be XMI, which is the standard format for UML model interchange. This XMI file will be the input of the A3S validation web service, which will transform the XMI file into an XML file, using an XSLT engine taking for rule extraction parameters an XSLT sheet. The XML file obtained will be called the A3S XML information file, and will be the input of the verification engine as far as for the process of generation of configuration files that may be used to reconfigure some of the hardware components of the platform, as illustrated in **Figure 9**.

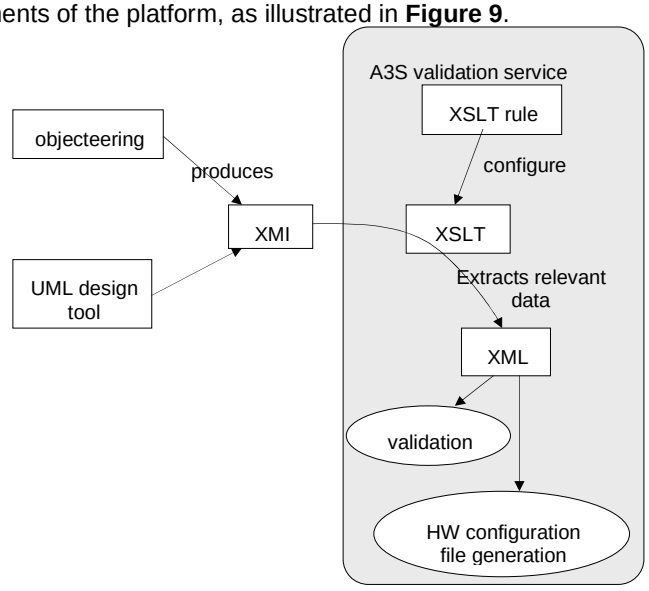


Figure 9 : validation process

4.3 Modeling of hardware characteristics and application requirements

As software radio offers a quasi-direct access from simulation to implementation on a hardware embedded platform, A3S aims at adding these considerations into the design flow. 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 on the multi-processing architecture.

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 used regarding the hardware 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, 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, and the middleware of the platform which is a part of the system itself, and may have influence on the system performance.

4.4 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 parameters of each hardware component are filled as shown by the Figure 10.

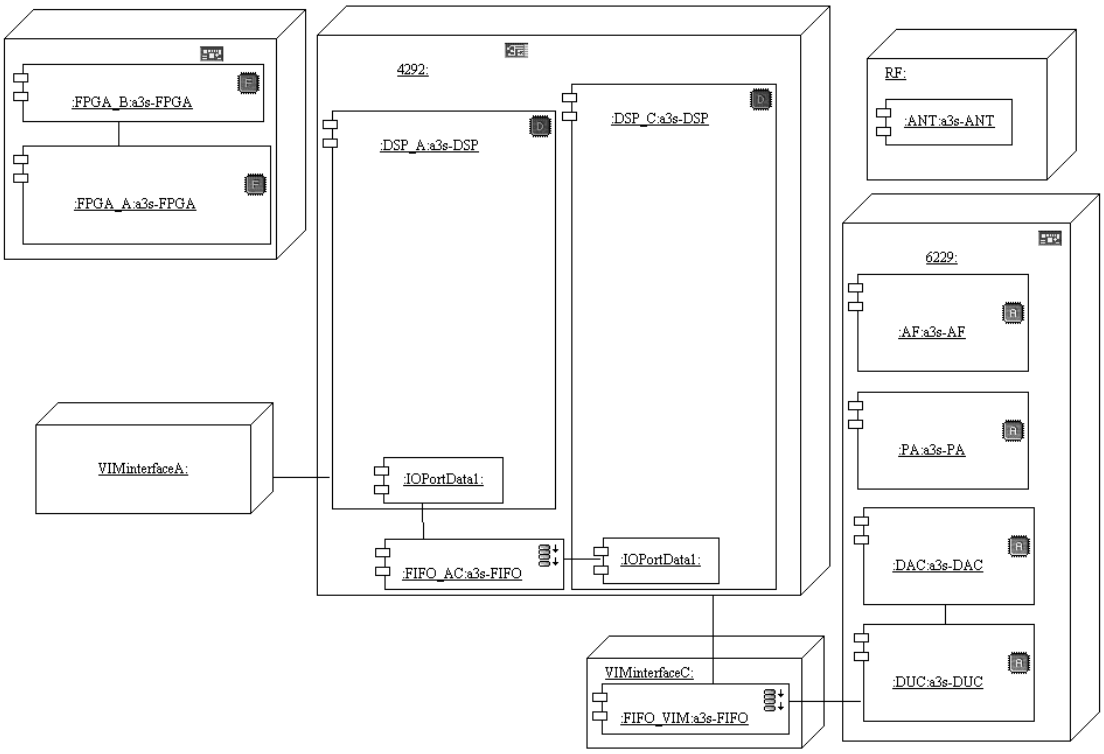


Figure 10: deployment diagram - platform description

The software graph of Figure 11, which is an UML 1.4 activity diagram, addresses the logical links between the different software 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).

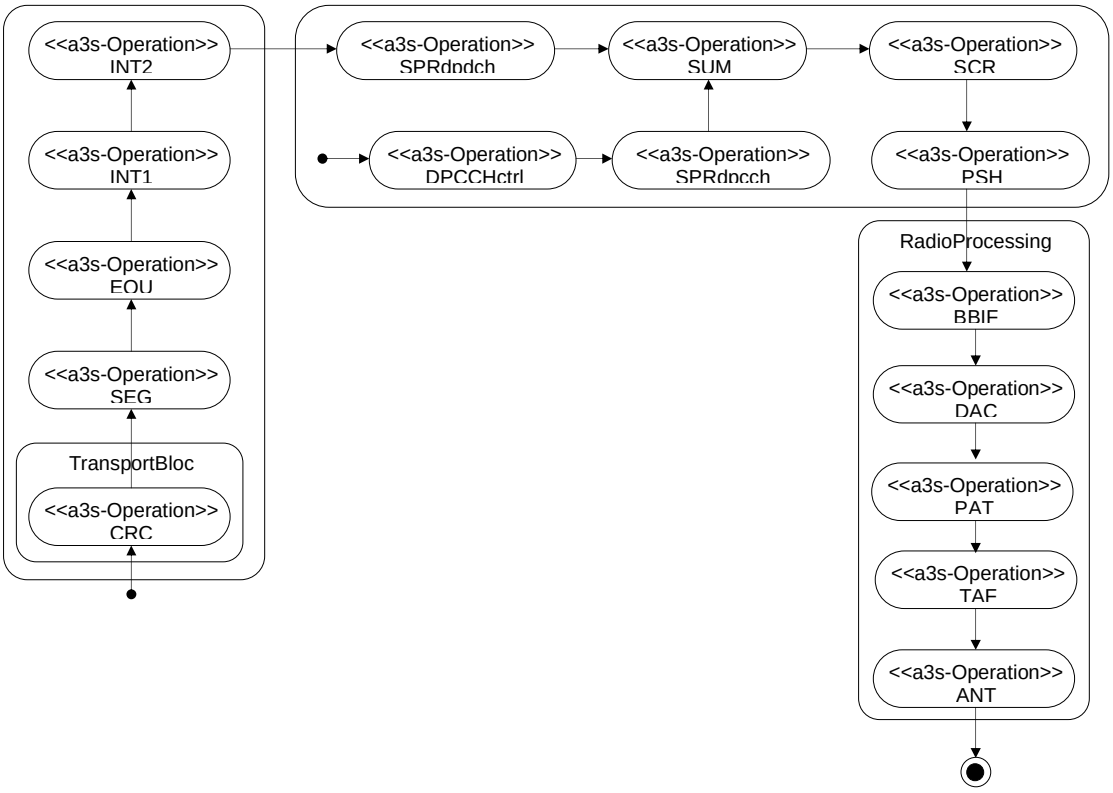


Figure 11: activity diagram for a UMTS transmitter - application description

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 (Figure 5).

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. These informations are provided through a constraints composition approach as resented below.

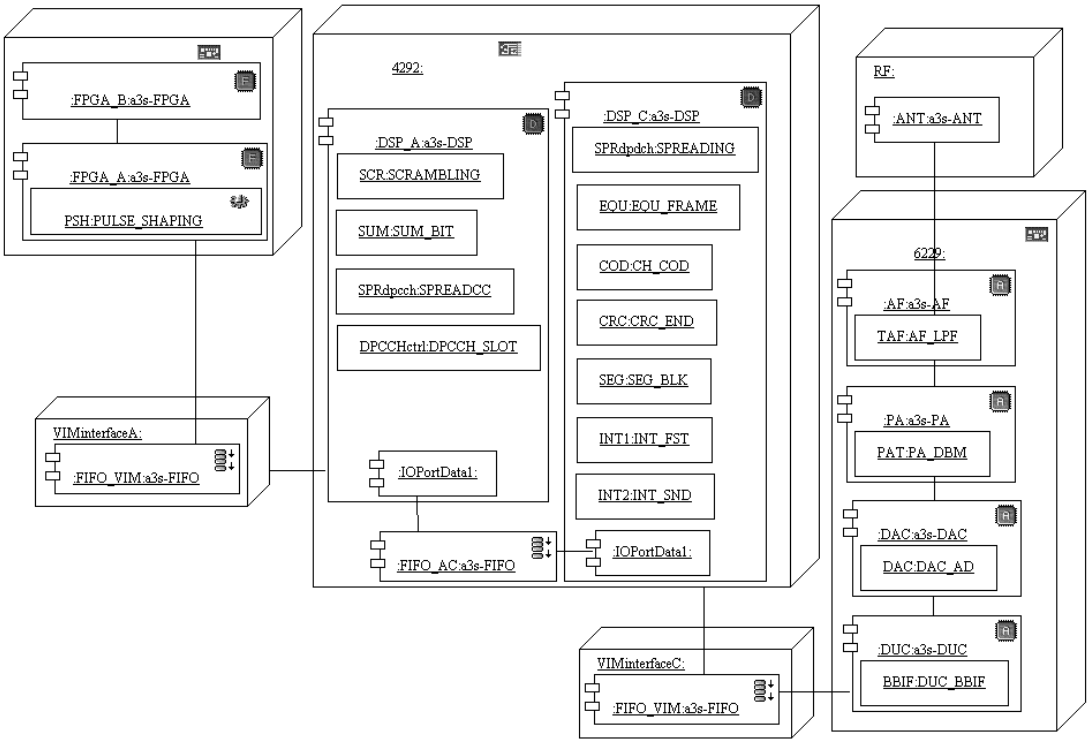


Figure 12: deployment diagram - platform description with software components

5 CONSTRAINTS COMPOSITION

Different types of constraints must be considered in order to validate the QoS of a system: functional and non-functional constraints. Functional constraints verification leads to verify (mainly by simulation or execution) that the application correctly perform the expected computation. Non-functional constraints verification, aims to verify (mainly by simulation, prototyping, analytically or formally) that the system satisfies constraints related to its coherency and its performances (mainly timing, area and consumption). In the following, we focus on the non-functional constraints for each step of the design flow (application, platform and mapping).

5.1 Software architecture verification

The software architecture modeling, which consists in determining and connecting software components, requires a phase of analysis to validate the representation carried out, as errors can occur in the representation. It is thus necessary to provide the designer with the detected errors. They can be can be related to the symmetry of the inter-connected functions. A certain number of constraints of coherence must thus be 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 bring answers in term of feasibility. Hence, with the attributes, on which the application is mapped, it is possible to check if the model of execution and the coherence of the periods of the various SW components are corrects. The constraint being that the interdependent functions should not overlap, otherwise a system dysfunction will occur. Some components can also execute an operation depending on a signal resulting from another component, which results in constraints of synchronization that must also be taken into account.

The verifications done at the software architecture level are related to specification and execution model coherency.

5.2 Hardware architecture verification

The platform way of modeling 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 concerned since it is the platform coherency that needs to be highlighted. Hence, the coherency of types and connections must be validate. For instance, a simple but useful test consists in automating the checking that some inputs are not connected directly to other inputs without first being connected to outputs. The architecture coherency is the first step of a global verification process for which an automatic tool can efficiently assist the designer.

5.3 Hardware-Software mapping verification

The mapping of a SW architecture on a HW one enables to check if the system meets the required performance constraints. Within this stage, all the attributes are fulfilled and the system can be validated. We can then have a large number of metrics about the system performances. Those values can be obtained after a preliminary verification of the coherence constraints since a SW component that requests a certain amount of resources cannot be implemented on a HW component that cannot offer the required resources.

As a software component placed on a specific hardware component do not have the same performances as if it is placed on another hardware component the evaluation of the system performances constraints must be carried out. The verification corresponds first to the execution time of the application, which is the combination of execution time[xi-xii] 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, localization of critical points (overuse of a resource whereas others are available) are 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.

The functions are also subjected to constraints of scheduling which depends 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. Indeed, an execution of a task cannot take place if the input data are not ready. It is thus necessary to identify the possible dependency constraints in order to try to reduce or eliminate their consequences by finding scheduling solutions that overcome the problem.

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.

6 THE A3S PROJECT

6.1 Goal of the project

The A3S Project responds to the priorities of the RNRT 2002 call for proposal, by highlighting the high level step of the systems design. It aims at realizing a tight coupling between the specification and the different constraints to enable an a priori verification of the software architecture adequacy to the target platform in the software radio domain, especially for the 3G and 4G telecom. 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 of both technologies software and hardware,
- Tight coupling with development environments.

6.2 Constitution of an UML profile dedicated to non-functional coherence verification

The capabilities presented above correspond to the basic tools needed to design any software radio system. Furthermore, it is the choice and the accuracy of the set of parameters associated with each hardware or software component that makes the design a success or not.

This is what aims at providing A3S: an effective profile for SDR.

Coupled with already existing efforts on the domain done at the OMG for instance around the Swradio DSIG and the SCA. An organization of SDR design around such a tool favors reuse and design time inside a company, as well as cooperation between companies using the same tool and method.

6.3 The Demonstrator

A UMTS-FDD communication system for dedicated channels has been chosen to investigate the requirements of such a design and to highlight the features expected from the A3S tool.

This example provides a complexity and a set of constraints high enough to be considered as a good evaluator of the A3S tool accuracy.

Confronting the tool to such a use case will prove the efficiency of the solution. It will concretely show to the software radio designers of 3G systems the benefits affordable with appropriate tools.

Major challenges offered by the use case are:

- processing speed demanding,
- embedded memory allocation to be optimized,
- hardware (reconfigurable or not) circuits surface to be evaluated,
- communications means between processing units dimensioning,
- embedded middleware to be chosen and associated overhead,
- taking into account power consumption constraints.

7 CONCLUSION

Strong progresses have been made since the project's beginning, demonstrating the basis of our approach. The team involving some major radio manufacturers as well as a well-known academic research team and a leader of UML workshop editors has achieved the definition of the basics of the case study, and alpha version of the A3S UML tool is already running. We attempt to finalize some work in the next months and to publish it in a wide dissemination optic, the purpose being to

expose the benefits of our researches to the software radio community and promote our results for standardization.

We propose to give our publishes and results to the WWR Forum and collect their feedbacks for the benefit of the A3S project as well as the software radio and wireless communities.

ACKNOWLEDGEMENTS

This work has been performed in the framework of the RNRT French funded project A3S. The authors would like to acknowledge the contribution of their colleagues from A3S consortium.

LIST OF REFERENCES

[i] CoFluent Design: http://www.cofluentdesign.com/

[ii] A. Gerstlauer, R. Dömer, J. Peng, D. D. Gajski, *System Design: A Practical Guide with SpecC*, Kluwer Academic Publishers, Boston, MA, ISBN 0-7923-7387-1, June 2001.

[iii] SynDEx: http://www-rocq.inria.fr/syndex/

[iv] E. Martin, O. Sentieys, H. Dubois, J. L. Philippe, "GAUT, an Architecture Synthesis Tool for Dedicated Signal Processors", Proceedings of International EURO-DAC Conference, pp. 14-19, 1993.

[v] OMG Unified Modeling Language Specification, final adopted specification January 2002 UML1.4 OMG document

[vi] http://www-sop.inria.fr/meije/esterel/esterel-eng.html

[vii] UML profile for Schedulability, Performance, and Time Specification – ptc/02-03-02 OMG Adopted Specification

[viii] UML profile for Software radio – OMG draft

[ix] UML profile for Quality of Service and Fault Tolerance Characteristics and mechanisms – OMG revised submission

[x] http://www.corba.org

[xi] Ali Dasdan, Dinesh Ramanathan, Rajesh K. Gupta

"Rate Derivation and its applications to reactive , real-time Embedded Systems"

Proc. the 35th Design Automation Conference (DAC) pp. 263-8 , June 1998

[xii] Richard Gerber, Seongsoo Hong, Manas Saksena

"Guaranteeing Real-time Requirements with Resource-Based Calibration of Periodic Processes"

IEEE transactions on software Engeneering, July 1995.