A MODEL BASED METHODOLOGY FOR SCA WAVEFORM DESIGN ENHANCING PORTABILITY: APPLICATION TO THE FM3TR WAVEFORM APPLICATION

Frédéric LE ROY (ENSTA Bretagne, Brest, France, frederic.le_roy@ensta-bretagne.fr)
Joël CHAMPEAU (ENSTA Bretagne, Brest, France, joel.champeau@ensta-bretagne.fr)
Jean-Philippe DELAHAYE (DGA MI, Bruz, France, jean-philippe.delahaye@dga.defense.gouv.fr)

ABSTRACT

The software design of Software Defined Radio (SDR) systems in the military domain rely on the use of standardized and open technologies. The SCA was developed with regards to several goals among the following: the use of Commercial Off-The-Shelf (COTS) technologies, portability of software applications, modularity and reusability of software, hardware abstraction. SCA also allows scalability of its architecture over different types of platforms. Contributions on tooling for modeling, simulating, developing waveforms or for entire system design. COTS tooling has to constantly evolved and often coming from various domains tackle partially the many challenges of SDR waveforms and systems design. Thank to the MoPCoM project and its MDD based methodology, this paper will present an ongoing study that aims to contextualize the MDD flow for the SCA waveform design.

1. INTRODUCTION

1.1. The military SDR Context

The software design of Software Defined Radio (SDR) systems in the military domain rely on the use of standardized and open technologies.

The Software Communications Architecture (SCA) [1] is a software architecture developed and standardized in the Join Tactical Radio System (JTRS) military program that was initiated in early 1997. Synthetically presented in [2] the SCA is a software architecture to be deployed onto SDR platforms; it provides abstraction to the waveform application from the hardware architecture.

In may 2006, the JTRS has publicly releases the SCA v2.2.2 specifications that represent the major contribution of standardization in SDR. In august 2010, the JTRS with the WinnForum has organized a general meeting to discuss about the enhancements to the SCA. As the SCA v2.2.x is suitable for large radios, the SCA Next proposal has the objective to provide more flexibility to address more efficiently the design of low-power, low capacity radio.

At the European level the ESSOR program will propose a SCA-like standard compliant with the SCA v2.2.2 that defines Operating Environment extensions for DSP and FPGA devices.

Some other contributions on standardization are the PIM and PSM SWRadio Components Profile released by the OMG in 2007, and the SDR Framework from the WINTSEC EU project.

The SCA specification is a set of requirements that help to produce portable software code, but it does not ensure it. The software design process is also a key factor. Then in 2009 the JTRS releases the "Waveform Portability Guideline" document [3] that addresses the practices for software development to enhance the waveform portability.

1.2 The SCA Software content

The SCA defines an Operating Environment (OE) through a set of requirement specified in [1]. The Operating Environment is defined as a layered software architecture that provides the abstraction to separate the application software from the hardware layer. The POSIX Operating System (OS) is the lower level of the OE. A subset of OS functions is defined called Application Environment Profile (AEP) for direct application calls to the OS. The middleware services based on CORBA are used to manage exchanges between the software components distributed over the platform resources. On top of that the Core Framework (CF) is defined through four groups of interfaces, 1) Base Application Interfaces, 2) Base Device Interfaces, 3) Framework Control Interfaces, 4) Framework Services Interfaces. The Core Framework provides management, control, deployment and configuration services of all the system software components for application resources and hardware devices. The CF interfaces are implemented using the Interface Description language (IDL) of CORBA.

The SCA specifications also include the Domain Profile which is a structured set of XML files. These files describe all the resources within the SCA system in terms of components, ports, interconnections, properties, locations and capacity models.

1.3 Waveform software

Facing the multiplicity of the waveforms and the diversity of the platform architectures and form factors, the original aims of the SCA are to facilitate the waveform development in terms of portability and waveform deployments onto heterogeneous SDR platforms

Copyright(c) 2011 The Software Defined Radio Forum Inc. - All Rights Reserved

The waveform applications are composed of a collection of component interconnected to each others through ports. The ports provide or use services relying on interfaces defined by the SCA Base Application Interfaces. The JTRS API supplements could be additionally used by an application component ports as for example the Packet API.
The application is described in the Domain Profile, by the Software Assembly Descriptor (SAD) which includes the list of the application's components, their placements, configurations and interconnections. Each component is itself described into several XML files, the Software Package Descriptor (SPD), the Software Component Descriptor (SCD), and the properties files (PRF).
The remainder of the paper is structured as follow. The next section discusses about different design approaches related to the Model Driven Design (MDD) in the domain of SDR and the tools that specifically help to develop SCA applications. The section 3 will present the model based MOPCOM process on which our approach derivates. The section 4 presents our model based methodology based on MOPCOM process, the project process to perform the porting of the FM3TR waveform onto 2 different platforms and the first result on the methodology.

2. BACKGROUND

The interest in the MDD approaches is going in the same way that the goals purchasing by the military SDR standardization activities described in the above introduction. The MDD brings to these aims an increased level of abstraction needed for more separation of concerns during waveform design. By using high level abstraction language such as UML, MDD provides independence of the model from the sources code languages, facilitates the project collaboration by the shared views of models, generating an automatic documentation early in the design process.
Within the SCA scope, the possibility for modeling tools that embed a SCA profile to generate SCA source code, and Domain Profile files could also help to automate some parts of the certification process.
This part will overview some of the MDD approaches that address the modeling issues of the SCA applications and some of the existing tools that provide facilities for SCA waveform development.

2.1. Introduction to MDA

One of the model based design approach with high level of specification maturity is the standardized Model Driven Architecture (MDA) [4] coming from the OMG. OMG solution addresses the design of interoperable and distributed applications through the use of models. The MDA separates the application domain aspects from the technical specific aspects with the development of a methodologies based on model transformations within a Y-chart co-design flow. A Platform Independent Model (PIM) of the application is mapped onto a Platform Model (PM) representing the target architecture. The result of this models mapping is a Platform Specific Model (PSM) that represents the implementation.

2.2 Related works on the SDR SCA Domain

Coming with the SCA standardization that aims at providing more waveform portability, some domain specific tools have been developed. Commercially available tools such as Zeligsoft CE, Spectra Tools (first generation of tools), its successor Spectra CX or the SCA Architect form the CRC, aim at providing facilities to developers to produce SCA compliant code and to deploy it onto platforms. Despite the evolution of these offers that are mainly GPP centered, waveform portability is an active topics in the research domain with many contributions [5] [6] [7] [8] [9] [10], presented during the past 10 years at the conferences such as MILCOM or the SDR technical conference.

The Government Reference Architecture (GRA) [11] is a proposal supported, among others, by the CERDEC for SATCOM terminal system design. GRA further defines a process with MDA model approach using SysML/UML modeling tools through four level of abstraction to capture system's functionalities, architecture, interfaces. The first level of modeling allows capturing the system's behavior with scenarios, uses cases and the system's functional modules. It is the Computational Independent Model (CIM) level as defined by the OMG. The second level is the PIM refining the CIM, by defining system architecture with its abstract interfaces and CIM functional allocation onto the architecture.
The GRA is focused on these two first levels of modeling and also addresses two model refinements with the PSM level and the Platform Specific Implementation (PSI), more details on GRA available in [12]. It addresses waveform portability in the way that it facilitates code integration over multiple platforms by providing common set of modeling elements to represent them.
The GRA integrates the SCA as an operating platform basis on its testbed. The tools are Rhapsody for UML SysML Modeling, and Spectra CX for the SCA domain specific aspects.
In particular, the experience report [13] describes in detail a tool chain based on Rhapsody and Zeligsoft CE (now CX) tools and the associated waveform development process. Here the development process is iterative with three steps.

The first step consists in modeling with Zeligsoft tool the SCA components and the waveform, then the source code corresponding to the skeleton (infra-structure code) and the Worker class (that will encapsulate the functional code) of components are generated by the Zeligsoft Code Generator.
The second step is a reverse–engineering operation that allows obtaining a class model of the previous Worker class into the Rhapsody tool. The functional code is then inserted to the UML model and next Rhapsody's generator generates the source code.
The last step consists to compile the component code with makefiles that take into account the specific constraints of the target platform (OS, ORB, CF).
Some other modeling approaches lean on the waveform simulation deployed onto virtual platforms. The project [14] from MITRE proposes a waveform porting environment where the platform is modeled at Transaction Level Modeling (TLM). The simulation of platform is also adopted in [15]. A SystemC TLM simulation is used for PIM and PSM model of SDR platforms. The proposal also defines a waveform development methodology that allows simulating waveform PIM within the SystemC virtual platforms.
As presented above, many works deal with waveform modeling, simulating, development approaches steps of the porting process. Despite these contributions to enhance portability, additionally to the standardization efforts, there are few contributions proposing tools to characterize the portability level and performance of SCA waveform code. A Datasoft has presented in [16] some tools that address the performance analysis and port complexity. The DataSoft Waveform Analysis Tool (DSWFAT) handles estimation of source code complexity of SCA waveforms, using complexity porting metrics. The tool helps to caracterise the porting complexity with complexity metrics applied to the domain of SCA waveform as listed in the Waveform portability guideline [3]. The Datasoft Software Probe (DSSP) probes the inter-component messages at run time on target platform. The measurement are processed by the tool to caracterise the component complexity, giving visualization capabilities to help during the porting effort. The authors of [16] also mention the possibility for DSSP to be integrated with the Zeligsoft tool.

Today, the SCA v2.2.2 is platform specific through the specification related to CORBA. So there are several issues to fully apply an MDA approach to the waveform development. This situation implies that a waveform could not directly be described as a PIM using the SCA v2.2.2 as a profile.
The SCA Next release has the objective to transform the SCA to a platform independent specification and in the same time SCA Next aims at providing different type profile to target different type of platform.
According to this future context, we believe that SCA waveform development will fully take advantages of MDA process, and gain on portability with also the condition of MDA tooling availability.

3. PROCESSUS DE DEVELOPPEMENT DE FORME D'ONDE SCA

This section presents the MoPCoM (http://www.mopcom.fr) MDA process developed in the scope the MoPCoM project. We choose the MoPCoM process and its associated tooling to apply it to the development of SCA waveforms.
Originally, the MoPCoM process has been formalized through the SPEM meta-model to bring the benefits of the MDA technologies to the Electronic Design Automation (EDA) domain. The goal of MoPCoM is to give ability to the system designers to manage the growing complexity of the SoC/SoPC design and to share common view early in the process with the application developers.

3.1. MoPCoM process overview

Thanks to the authors, an extract of our paper [17] shows below a rapid description of the MoPCoM process.

"*The MoPCoM methodology is a methodology defined to develop SoC/SoPC applications [17], [18]. This methodology is based on UML and MDD. It is a refinement of the MDA Y-chart dedicated to design space exploration and Platform Based Design. The MoPCoM process proposes a structured iterative process of modeling. It takes as input functional, non-functional and allocation requirements expressed in SysML. The Figure 1 gives an overview of the process, highlighting 3 modeling levels:*

- *The Abstract Modeling Level (AML) 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,*
- *The Execution Modeling Level (EML) is intended to provide a generic platform defined in term of execution, communication or storage nodes in order to proceed to coarse grain analysis,*
- *The Detailed Modeling Level (DML) is intended to provide a detailed description of the platform in order to proceed to fine grained analysis. It allows RTL code generation for harware (VHDL) and software (C) parts including glue logic (drivers).*"

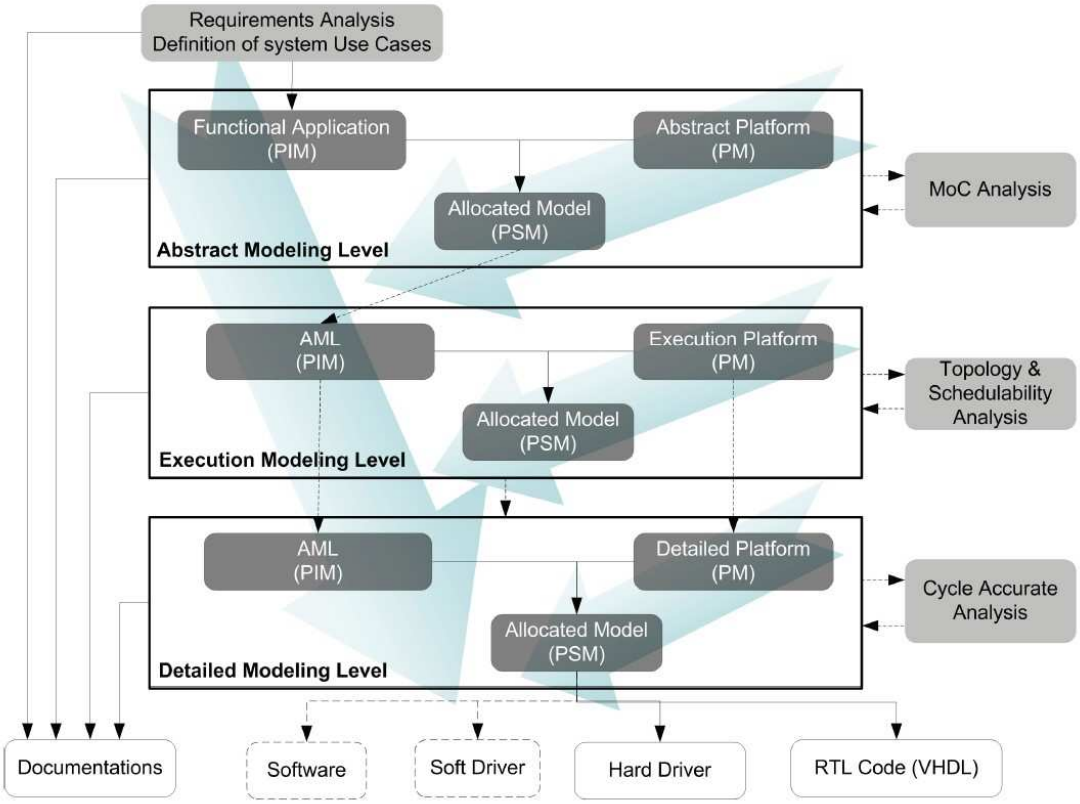

Figure 1 : MoPCom Process Overview [17]

3.2. MoPCom Process Tooling Support

The domain specific MDD tools could be classified as they accomplish different roles during the development process.

The first type of tool used in the MoPCoM process is the creation tool (modeler), a model development environment used to create and edit the models encompassing the set of meta-models used to describe system to design. The second type of model meet in the MopCoM process is model transformation tool by means of it, produces from the model different kind of artifacts like source code, documentation, or other models. The third kind of tool taking place in the process is the analysis tool with the goal of verifying the rules on models, check the completeness, inconsistencies, produces error and warning reports. This rules checking operates at the different level of abstraction of the process and also allows the measurement of metrics associated to the model. Finally the last type is the simulator tool in charge to simulate the execution of the model at the different level of abstraction. It allows to plays and verifies the uses cases defined in the functional level (CIM).

The panel of tools for model is wider, some other kind of tool exists like the test tools, composition tools (for merging), metadata management tool, or reverse engineering tool that could be use in the scope of MoPCoM Process but not mandatory.

One of the main interests in a MDA based methodology is the capability to associate through a process the previously described tools appropriated for a specific domain. The XMI standard aims allows exchanging models between MDA tools. Then, at each step of the process, the replacement of one tool by another is possible. The advantage is that the models are fewer dependants of the tools than with a traditional tooling chain.

During the MoPCoM project, the modeler and simulator tool was Rational Rhapsody from IBM. It allows the functional model capture, and model simulation at different levels of abstraction of the process. However, in some cases like dataflow oriented cases, SystemC code generation is preferable to accelerate the simulation using standard EDA simulation tools.

The transformation and code generation tool was the MDWorkbench from Sodius Company. This tool allows

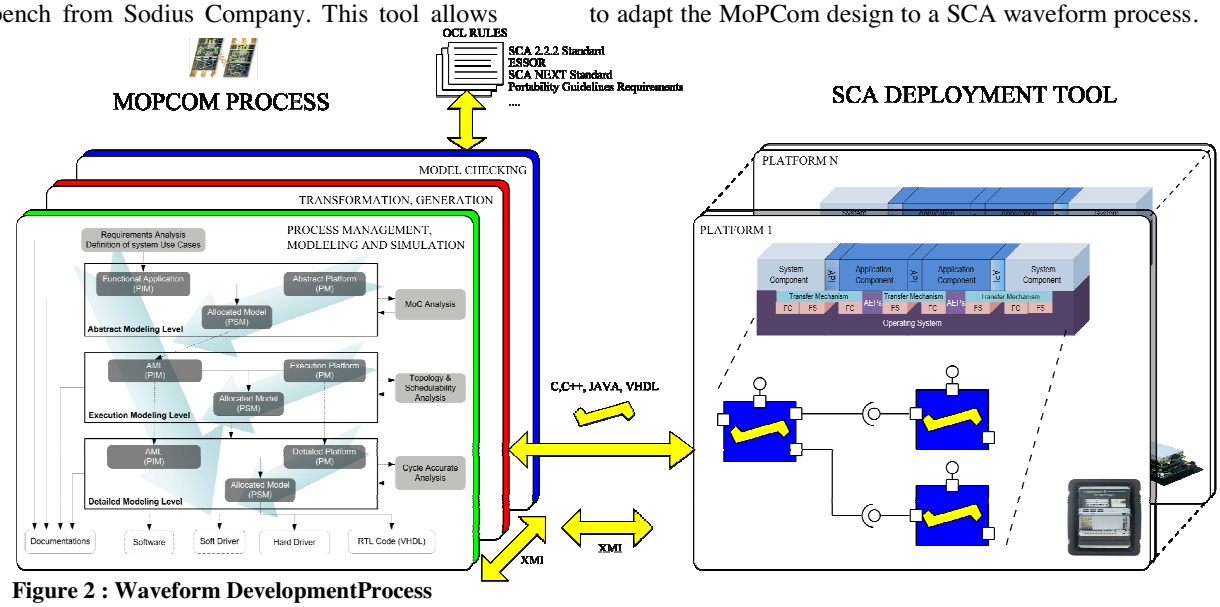

Figure 2 : Waveform DevelopmentProcess

generating from the AML and EML level, C++ or SystemC source code that could be useful for model simulation. It also allows generating C, C++ or VHDL code for the target platforms. This tool in integrated within Rhapsody through the OnDemend connector that allows import and export of models without the need of exchanging the whole model data each time. This functionality widely accelerates the import and export operations.

The tool for model analysis is Kermeta [19]. It is used during the process in order to verify the set of methodological rules captured in form of Kermeta constraints. The tool covers all the models by checking the rules at each level of abstraction.

4. MOPCOM PROCESS BASED WAVEFORM DESIGN

The first purpose of this section is to show if it is possible to adapt the MoPCom design to a SCA waveform process.

The second and third parts show a single simulated use case of an abstract model of the FM^{3}TR waveform. The last section gives an overview of our outcomes on this subject.

A signal processing application has been modeled in UML by MoPCom consortium. The simulation of system‘s uses cases couldn’t be executed by Rhapsody modeler. So, we chose to simulate the AML model of the waveform

outside the process MoPCom with *Simulink* tool chain

4.1. Design process

Figure 2 displays an illustration of our waveform design process proposal. This one is based on: MoPCom process [17], MDA tools and SCA deployment tool. Model

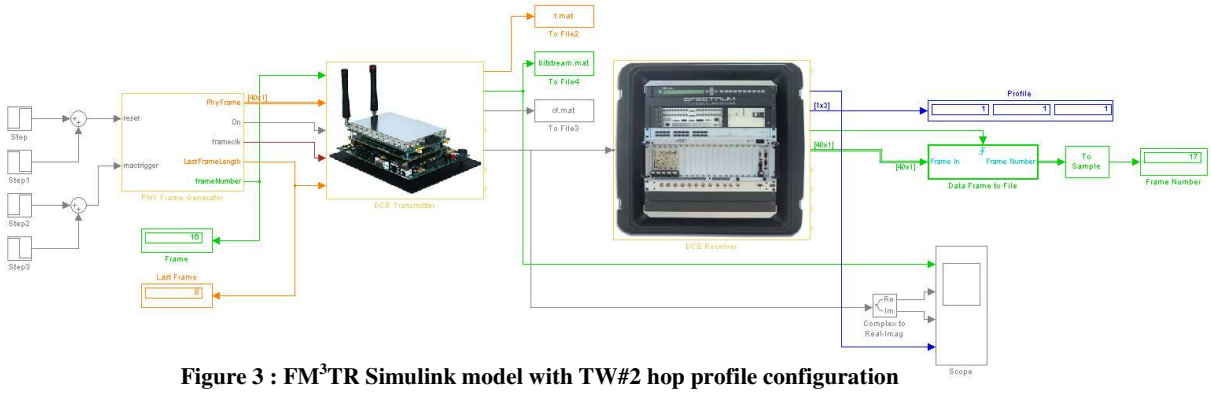

Figure 3 : FM³TR Simulink model with TW#2 hop profile configuration

exchange between tools is provided by the XMI standard interchange format. Unfortunately, such model exchanges between tools are not always correct and some information must still be added manually by developers. Codes generated and simulated (shown in yellow in this figure) under the control of the process are then integrated into the SCA components (sown in blue color)

4.2. FM³TR case study

The Future Multi-Band Multi-Waveform, Modular Tactical Radio (FM³TR) was specified in [20] and the first *Simulink* model was first published for voice transmission in [5]. This classical waveform consists of four network's layers PHY, MAC, DLC and NWK spread over the first three layers of the OSI model.

To simplify this case study, we chose to only model and simulate the physical layer of this waveform.

This layer uses a FHSS (Frequency Hopping Spread Spectrum) modulation scheme associated to a Minimum Phase Shift Keying (MSK) modulation with a data rate $R = 25\,kb/s$. The carrier frequency of transmitted signal follows the hop profile timings shown in Figure 4.

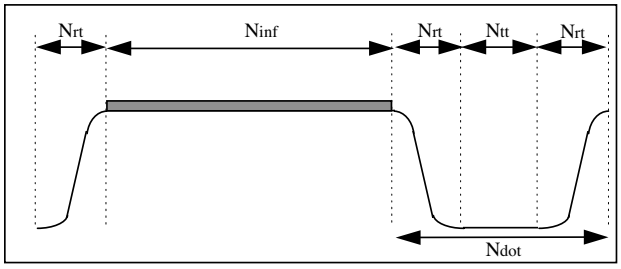

Figure 4 : Hop representation [20]

Table 1 and Figure 4, extract from specifications [20], defines three profiles of frequency hopping *TW#1a*, *TW#1b* and *TW#2*. The data transmitted by the physical layer during the dwell time period is *Ninf*.

Tableau 1 : Hop profile timings [20]

Timings in bits	TW#1a	TW#1b	TW#2
Ninf	80	40	10
Nrt	5	2,5	0,5
Ntt	10	5	1
Ndot	20	10	2
Ndot+Ninf	100	50	12
f (hop/s)	250	500	2083.3

Frequencies are selected from a set of 128 frequencies gotten by a secure transmission (TRANSEC). The message transmit by a physical layer on the radio channel is divided into frames. Each of one are composed of five frequencies hopping and only four of them are carrying useful information. The last hop ensures the management of synchronization between transmitter and receiver. Each message is preceded by a preamble synchronization of constant size and ends with two frames carrying an end of synchronization message (EOM: End Of Message).

4.3. Experiments

This section is composed of two parts, the first one describes simulation's results of *Simulink* model and second one gives feedback of this simulation on SCA waveform development.

4.3.1. FM³TR's modeling and simulation

We have modeled in *Simulink* (see Figure **3**) the exchange of a text file between two physical layers. In Figure 3, the text file to transmit is 76 characters long.

As shown in this simulation, when transmitter and receiver are configured to *TW#2* hop profile, transmitter segments the file into 16 frames. The last one just contains 8 bits of data over the 40 bits transmitted.
In Figure **3**, the receiver has received 17 frames instead of the 16 planned (see Table 1). This difference came from a ghost's frame added by transmitter to confirm end of message synchronization.

4.3.2. Feedback

This model was first performed to generate test files of elementary modules of the physical layer.
We also wanted to use this model to make a rapid prototyping of modules embedded by non-CORBA components or under SCA API.
Unfortunately, model designed couldn't be executed by *Simulink* under 5 minutes if simulation engine wasn't setting up to discrete mode with a variable sampling time. In this pattern model, sampling time depends on signal's vectors size and frequency's parameter of source's bloc. Thus, RTW tool that can generate C language source code can't be used if simulation engine isn't setting up to fixed sampling time. Using our model for code generation is then possible with manual recoding over cost.

Thus, simulations are more difficult to perform above the physical layer because protocols used are difficult to model by the Model of Computation (MoC) Synchronous Data Flow (SDF) of *Simulink*. Indeed, in this model's type, sampling period can be made implicitly defined by tuse of buffer / unbuffer blocs. For such a model, development time is quite difficult to control. For example, protocol's layers like FM^{3}TR DLC layer implement a state machine that makes software components inactive during a period that depends on radio channel's load. As most of communication systems, FM^{3}TR can therefore be classified as Asynchonous Globally Locally Synchronous (GALS) systems that are particularly difficult to model and particularly on OSI layer which implements protocols.

In addition, SCA's components are currently based on CORBA middleware. In draft specification of future standard SCA Next, this constraint was relaxed and software architecture core framework has accordingly been adapted. In this kind of model, data processing is made by asynchronous call to distribute services. Size of CORBA packets transmitted by this software bus can be configured by setting ORB (Object Request Broker) before starting it on platforms. Furthermore, latency induced by CORBA use is difficult to predict [21] as it strongly depends on performance of execution platform. As a result, queues often must be inserted between ports of the components of PSM model to compensate delays CORBA message contention on hardware buses and network layers.

Whether at PIM or PSM models system's simulation is not so easy to do. Indeed, model execution (PSM model deployed) must verify by use cases (installation, configuration, starting or stopping SCA waveform). These checks could be made only if executables models of ORB and operating system are available for platforms. This choice based on virtual platforms modeled in SystemC has been proposed in [15]. This proposal is very attractive because it offers several levels of abstractions that can be executed. Nevertheless, for a developer team this proposal implies to hold man power just for maintenance of platforms' models.

We defend SystemC code generation through MDWorkbench tool of process MoPCom. Then models generated can be simulated for requirements validation defined by the UML diagrams (Use Case type) of the CIM model. Unfortunately, test environments generation based on use case model is not yet possible (choice of action language, operational semantics, tool integration …). Additionally, process MoPCom can't generate executable models composed by several MoC.

5. CONCLUSION AND FUTURE WORKS

An overview of our future works can be extract from Figure **2**. The analysis tool of MoPCom process can check a set of rules that models should respect. Some requirements associated with specifications such as SCA 2.2.2, ESSOR can be formalized using OCL (Object Constraint Language) and checked by the analyzer tool. In the same way, we are working on metric computation introduced in paper [3] to provide an idea of the portability of SCA models.
In such MDA process, this separation of concerns should soon improve the certification of software radio terminals with standardized software architectures. Thanks to a custom and flexible tool, MDA technology used in SCA's context allows multiple trades to work at different step of waveform‘s life cycle design

6. ACKNOWLEDGEMENTS

This research is in part supported by the French DGA (General Armaments Directorate) the French procurement agency and by the ENSTA Bretagne Laboratory.
The views expressed in this paper are those of the authors and cannot be regarded as stating an official position of the DGA or the French DoD.

7. REFERENCES

[1] JTRS Standard, Joint Program Executive Office of the Joint Tactical Radio System, "Software Communications Architecture Specification", Final/15 May 2006 V.2.2.2

[2] C. R. Aguayo Gonzalez, C. B. Dietrich, J. H. Reed, "Understanding the Software Communications Architecture", *In proceedings of IEEE Communications Mag*., September 2009.

[3] JTRS Standard, Joint Tactical Radio System Network Enterprise Domain Test & Evaluation, "Waveform Portability Guidelines", December 2009 V.1.2.1, http://sca.jpeojtrs.mil/_downloads/20091228_1.2.1_NEDTE_PORT_GUIDE.pdf

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

[5] M.S. Gudaitis et R.D. Hinman, "A waveform Description Language for Software Defined Radio", *In proceedings of SDR'02*, November 2002.

[6] V.J. Kovarik, "SDR Architecture Impacts On Waveform Portability and Cost Modeling", *In proceedings of SDR'06*, Wyndham, 13-17 November 2006.

[7] Y. Zhang, S. Dyer, N. Bulat, "Strategies and Insights into SCA-Compliant Waveform Application Development", *In proceedings of MILCOM'06*, December 2006.

[8] T. Kempf, E.M. Witte, V. Ramakrishnan, G. Ascheid, M. Adrat, M. Antweiler, "A Workbench for Waveform Description Based SDR Implementation", *In proceedings of SDR'07*, November 2007.

[9] J. Hogg, F. Bordeleau, "Optimizing Portable SDR Software", *In proceedings of SDR'07*, November 2007.

[10] S. Singh, M. Adrat, M. Antweiler, "Nato RTO/IST RTG on SDR: Demonstrating Portability and Interoperability of SCA-Based Waveforms", *In proceedings of SDR'09,* December 2009.

[10] P. Johansson, Z. Cao, W. Hodgkiss, "Rapid Porting of an FMRTR Waveform", *In proceedings of SDR'09*, December 2009.

[11] T. Rittenbach, V.J. Kovarik Jr, R. Krause-Aiguier and C. Steward, "Complex Terminal Systems Design : Minimizing Time to Deployment", *In proceedings of IEEE MILCOM'10*, San Jose, October 31 – November 3 2010

[12] T. Rittenbach, H. Satake, E. Redding, K. Perry, M. Thawani, C. Dietrich and R. Thandee, "GRA Model Driven Process", *In proceedings of IEEE MILCOM'10*, San Jose, October 31 – November 3 2010.

[13] S-P Lee, M. Hermeling, C-M Poh, "Experience Report: Rapid Model-Driven Waveform Development with UML", *In proceedings of SDR'08*, December 2008.

[14] K. Skey and J. Atwood, "Virtual Radios – Harware/Software Co-design Techniques to Reduce Schedule in Waveform Development and Porting", *In proceedings of IEEE MILCOM'08*, San Diego, 17-19 November 2008.

[15] G. Gailliard, E. Nicollet, M. Sarlotte and F. Verdier, "Transaction Level Modelling of SCA Compliant Software Defined Radio Waveforms and Platforms PIM/PSM", *In proceedings of DATE'07*, Nice, 16-20 April 2007.

[16] L. Dunst, S. Aslam-Mir, B. Duthler, E. Miles, A. Goff and M. Lerch, "A Novel Approach to Diagnosing Problems in SCA Waveforms During Development and Porting", *In proceedings of SDR'09*, Washington, DC, 1-4 December 2009.

[17] A. Koudri, J. Champeau, D. Aulagnier, and P. Soulard, "MoPCoM MDD Process Applied to a Cognitive Radio System Design And Analysis", *In proceedings of ECMDA-FA*, Twente, 23-26 June 2009.

[18] A. Koudri, J. Champeau, D. Aulagnier and D. Vojtisek, "Processus de développement UML/MARTE MoPCom pour le codesign", *Génie Logiciel, 2009*.

[19] INRIA, "Kermeta metaprogramming environment", *http://www.kermeta.org*.

[20] P. Mou, "Test Waveform Specifications Issues 2.0", FM3TR Technology Group, 1999.

[21] G. Abgrall, F. Le Roy, J.P. Delahaye, J.P. Diguet, G. Gogniat, "Predictibility of inter-components latency in a Software Communications Architecture Open Environment", 24th IEEE International Parallel and Distributed Processing Symposium, Atlanta, 19-23 April 2010.