AN ARCHITECTURE FRAMEWORK FOR SOFTWARE AND COGNITIVE RADIOS

Eric Nicollet*, Wolfgang König**, Markus Muck***, Didier Bourse***

* Thales Communications, Paris, France – eric.nicollet@fr.thalesgroup.com
Phone: +33 146 132 132
** Alcatel-Lucent, Stuttgart, Germany – wolfgang.koenig@alcatel-lucent.de
*** Motorola Labs, Paris, France – didier.bourse@motorola.com – markus.muck@motorola.com

ABSTRACT

Development of future Software Defined and Cognitive Radios, and dissemination of current technologies to other industrial domains, are strongly depending on the level of readiness of the associated technologies. The cost-efficiency of such technologies is a determinant driver, which can be improved thanks to standards emergence.
An Architecture Framework is proposed in this paper, aiming at capturing intrinsic structuring concepts of the SDR and CR domain, in order to facilitate synergies and, ultimately, facilitate emergence of more affordable solutions thanks to facilitated standards definition.
The key concepts introduced by the Framework are the notions of Reconfiguration Modules and Configurable Execution Modules, articulating together so as to cover Reconfiguration Architecture definition needs of any SDR and CR products. The importance of distinguishing Functional and Physical levels of abstraction is identified and underlined. A specific sub-classification is introduced, with a set of formal definitions associated.
Consideration on current technological standards and possible perspectives are provided.

1. INTRODUCTION

The continuing progress of SDR (Software Defined Radio) implementations in new generations of radio equipments, and the maturation of CR (Cognitive Radio), now reaching field demonstrators with closing industrialization of basic features, are adding one important dimension to the complexity of radio equipments.
This perspective is a structuring tendency in several domains of the radio industry, namely commercial, safety-of-life, defense and space. Each domain has specific priorities and issues in front of insertion of SDR and CR technologies. All are nevertheless sharing a common issue: reaching cost-efficient SDR and CR solutions, which is merely a matter of technology readiness.
In order to contribute to emergence of a shared cross-domains vision of the SDR and CR technological stakes, an Architecture Framework is presented in this paper.

This Framework is targeting to provide definitions and concepts capable to address the complete scope of CR and SDR technical issues. As such, the Framework is not proposing or promoting specific architecture solutions.

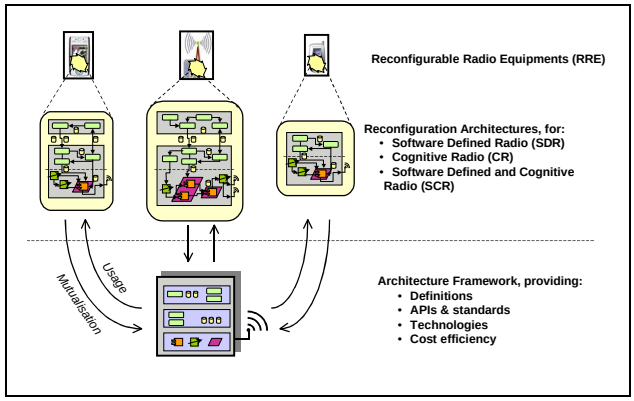

Figure 1: principle of an Architecture Framework

This Architecture Framework is drafting a future Domain Specific Language applicable to SDR and CR, from research to industrialization.

2. PRESENTATION OF THE ARCHITECTURE FRAMEWORK

2.1 Identification

The Architecture Framework if currently denoted as being the "REP", for "Reconfigurable Radio Equipment Platform Independent Model".
As its definition is progressing, this naming nevertheless appears to become obsolete and inaccurate. Thus, the denomination "the Architecture Framework" or "the Framework" will be used in the rest of this paper.

2.2 Reconfigurable Radio Equipments

The Architecture Framework is addressing Reconfigurable Radio Equipments (RRE). By RRE are meant any radio device or equipment featuring SDR and/or CR capabilities.

This definition of RRE is covering an extremely wide range of RRE products: from commercial terminal to cellular networks base stations, from public safety basic talkie-walkies to private networks base stations, from military handheld army radios to naval multi-channel communication units.
The nature of SDR and/or CR capabilities of RRE directly depends of the Reconfiguration Use Cases the considered RRE is satisfying. The balance between SDR and CR features can accordingly vary very much.

RRE with SDR capabilities may have no particular CR feature. For instance, a military handheld radio with manual selection of the transmission mode.
Reciprocally, RRE may have CR features without topping SDR capabilities. For instance, a smart commercial phone with two Radio Access Techniques (RATs) implemented in separated integrated circuits (no SDR), with advanced RAT selection at optimized operating frequency from a spectrum saving criterion.
Last, the most advanced architectures of future radios will certainly take advantage of both SDR and CR capabilities, with a significant coupling between the two technological dimensions. Such radios are considered in the sequel as SCR (Software-defined and Cognitive Radios).

2.3 Reconfiguration Architectures & Chains

The Framework is considering that each RRE has a specific Reconfiguration Architecture, which implements the SDR and CR capabilities of the RRE.
The complete Reconfiguration Architecture of a given RRE is grouping one to many elementary Reconfiguration Chains, each specifically answering to a Reconfiguration Use Case.
The Framework is addressing reconfiguration architectures at the unitary Reconfiguration Chain level, which are presumably decomposed in a series of modules.
Each Reconfiguration Chain is potentially integrating from high-level CR features down to low-level optimized SDR capabilities.

2.4 Main modules of a Reconfiguration Chain

Two cornerstone categories of modules are distinguished to classify the modules of a Reconfiguration Chain: Reconfiguration Modules (RM) and Configurable Execution Modules (CEM).

Reconfiguration Modules (RM) are covering all the treatments participating to the definition of the next configuration state to be applied on the RRE, without taking part to the configuration state once executing.

Configurable Execution Modules (CEM) are implementing the selected Configuration State, once the reconfiguration process is completed.
Reconfiguration Support Data are complex data participating to the Reconfiguration Chain, exchanged between the involved RMs and CEMs.

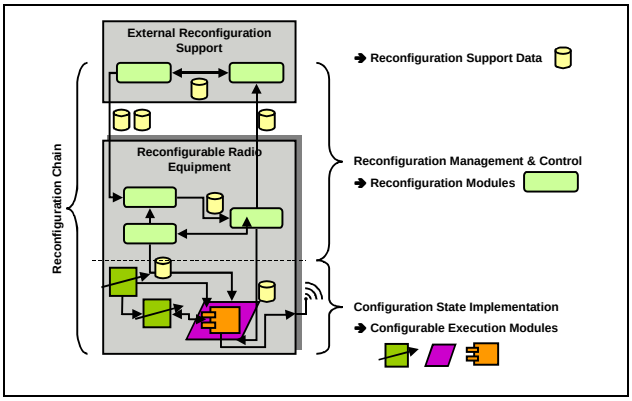

Figure 2: main constituents of the Architecture Framework

For non-autonomous Reconfiguration Chains, the External Reconfiguration Support is covering the set of external Reconfiguration Modules participating chain. It should be noticed that by definition no CEM may be external to the RRE.

CEMs are reconfigured into the appropriate physical implementation by some “last steps” RMs.
CEMs are composed of any configurable implementation items (hardware, software, or any composite) taking part to a given configuration state. Other implementation items, which are not configurable, are often complementing the implementation.

2.5 Functional & Physical abstraction levels

Two important structuring levels of abstraction from the implementation are identified to give supplementary characterization RMs:

- Functional abstraction levels
- Physical abstraction level.

Functional level of abstraction is making no explicit assumption relative to the implementation artifacts, while Physical level is directly considering the implementation artifacts available in the RRE.

Two categories of Reconfiguration Modules are then introduced from this distinction:

- Configuration Management Modules (CMM): are only dealing with Functional level of abstraction

- Configuration Control Modules (CCM): are, at least partially, dealing with Physical level of abstraction.

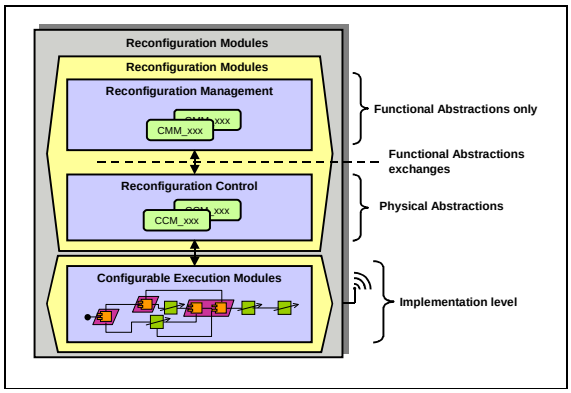

Figure 3: Configuration Management & Control

CMM are typically involved in CR capabilities or in the Decision Making necessary on top of SDR implementations, while CCM are typically involved on SDR-oriented use cases.
In case of CSR (Cognitive and Software defined Radio) equipments, the link between functional level of abstraction and physical implementation is undertaken by CCM modules, which are then capable to realize run-time associations between Functional requests and output choices towards Physical configurations.

The distinction between Functional and Physical level of abstractions is thus a cornerstone classifier of the Framework.

3. RECONFIGURATION MODULES (RM)

As introduced in § 2.4, Reconfiguration Modules (RM) are undertaking all the treatments of the reconfiguration loop, from the initial trigger to completion of the reconfiguration of the impacted CEM.
Four categories of Reconfiguration Modules are identified:

- Context Awareness,
- Decision Making,
- Reconfiguration Application,
- Reconfiguration Support Capabilities.

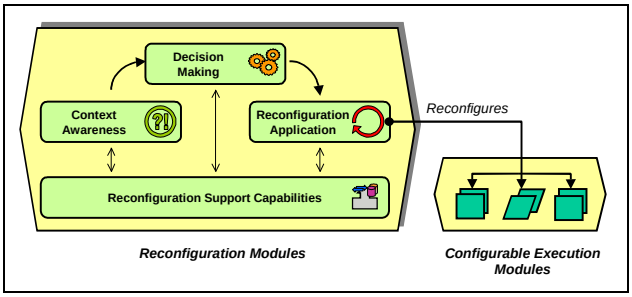

Figure 4: the notional loop involving Reconfiguration Modules

The introduced categories are intended to help the classification of RM of a given reconfiguration chain. In principle a given RM should belong to one of the four identified categories.

The relation between Context Awareness, Decision Making and Reconfiguration Application categories is a declination of the classical three-tier model “sensor/decider/actuator”.

If the arrows on the figure are underlining a principle relation of causality between those three categories, this does not imply anything concerning the exact interaction mechanisms between the Reconfiguration Modules, which may not be on a principle where a given stage would simply push the information towards the next stage. Protocols between decision modules can be much more sophisticated.

3.1 Context Awareness

Context Awareness modules are providing useful information to the decision loop regarding the context in which the RRE is evolving.
The context information can be of many different natures: information strictly internal to the RRE, information relative to the established radio communications supported by the RRE, or information relative to the decision context itself, such as applicable regulation and policies, etc.
The level of abstraction of the context information (cf. §2.5) can be at a functional (e.g. available RATs in a cell for network access, remaining battery life) or physical (e.g. defaulting software component identification, free memory on a given processor).
The interaction between Context Awareness modules and other RM (of any category, incl. Context Awareness) can be of many different natures: alarm-operation context awareness module are triggering events towards other modules, regular reporting modules are systematically delivering on a regular basis their information towards the decision making modules, on-demand responder can deliver information on an explicit request from another RM, etc.
The mechanisms to gather context awareness information can be strictly autonomous (internally to the RRE) or based

on a collaborative mechanism with external agents (e.g. over-the-air context information towards a terminal, or network-provided information towards a base station).

3.2 Decision Making 

Decision Making modules are participating to the decision itself, in accordance with the Reconfiguration Use Case associated to the Reconfiguration Chain, taking into account information provided by Context Awareness. The outcome of the Decision Making is to define the next Configuration State applicable inside the RRE.
Several stages of Decision Making can collaborate in a complex reconfiguration chain. The intermediate ones delivering intermediate information exploited by the further stages in order to determine the next Configuration State.
The nature of the Decision Making can vary from simple straightforward algorithms to high-end policy-based cognitive engines. In its simple expression, the decision could be delegated towards the end user of the RRE itself.
The Decision Making can be autonomously conducted at the RRE level, or involve in a collaborative scheme some external agents, such as a reconfiguration support server interacting with the managed RRE through a communication protocol. In case of mobile RRE, this would be over-the-air connection (e.g. *had hoc* networking), while fixed RRE (e.g. Flexible Base Stations) would rather use cabled connections to the core network.
The level of abstraction (cf. § 2.5) of the "next Configuration State" expressed by Decision Making can be at functional (e.g. request for a radio access technique detailed functional decomposition) or physical information (e.g. detailed hardware and software configuration for a certain radio access techniques, with associated low-level configuration values to be applied).
The complexity of the "next Configuration State" expressed by Decision Making can range from trivial (e.g. application of the appropriate value in a Configuration State selection register) to complex descriptors (e.g. XML files of a SDR waveform application in SCA-like approaches).

3.3 Reconfiguration Application 

Reconfiguration Application is in charge, once an expected Configuration State has been defined by Decision Making modules, to ensure its application towards the impacted Configurable Execution Modules.
This implies a transformation between the information expressing, at the output of Decision Making, the expected Configuration State, and the mechanisms to turn available CEM towards the appropriate configuration.
Such transformation can be based on specific approaches, or can use standard solutions.

To provide an example of a specific approach for Reconfigurable Application, one can assume that a Configuration State identification value is the output of Decision Making towards Reconfigurable Application. The Reconfiguration Application module, being developed with a priori knowledge of all possible Configuration States and the available CEM in the Reconfiguration Architecture of the RRE, can directly transform the Configuration State identifications into reconfiguration commands towards the CEM. No particular standard approach is needed in such a case, simple sequence of commands grouped according to "switch case" statements.

Examples of standard-based approaches for Reconfiguration Application are SCA or SWRadio Specification. See § 7.1.1.

The border between Reconfiguration Application (part of Reconfiguration Modules) and CEM deserves some clarification.
Reconfiguration Application is only active during the reconfiguration loop, which is making the RRE going towards the target Configuration State. Once the Configuration State is reached, Reconfiguration Application is not active until another reconfiguration process is triggered.
In front of the sensor/decider/actuator paradigm, the "actuator" dimension is thus distributed between Reconfiguration Application, which is undertaking the operations necessary to effectively reconfigure the CEM, and the CEM themselves, which are participating to the Configuration State itself, once they have been reconfigured.

3.4 Reconfiguration Support Capabilities 

Reconfiguration Support Capabilities are CR and SDR domain specific added value capabilities which are facilitating the operation of the Reconfiguration Management & Control chain. By themselves, such capabilities are not directly participating to the decision loop.
One example is Policy Management capabilities, which facilitates retrieving, updating, exchange, generation, of complex policies descriptors. Common protocols for such exchanges can be implemented by common Policy Management technologies.
A second example is OMA Device Management (cf § 7.2).

4 CONFIGURABLE EXECUTION MODULES (CEM)

As introduced in § 2.4, Configurable Execution Modules (CEM) are undertaking, once reconfiguration process is completed, the selected useful processing participating to run-time execution of a Configuration State.

If CEM concepts are mostly used for SDR implementations, the notion is applicable to Configuration States which are in themselves featuring Cognitive Radio features (e.g. a specific waveform with context-aware frequency planning).

Three categories of CEM are distinguished:

- Software-Defined Subsystems,
- Execution Environments,
- Software Execution Modules.

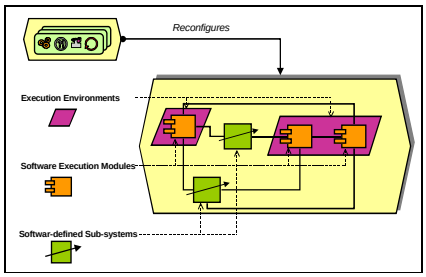

Figure 5: a simple configuration involving CEM

While Software-defined Subsystems are items exclusively reconfigured through parameters tuning, Execution Environments and Software Execution Modules are introducing reconfiguration capabilities through software programming.

4.1 Software-defined Subsystems

Software-Defined Subsystem are statically available CEM, generally embedding flexible hardware processing capabilities. They can be static software libraries as well.
They are reconfigured between the appropriate Configuration States through the parameters setting, under requests provided by the Reconfiguration Application.
Reconfigurable Transceiver sub-systems are typical examples of Software-Defined Subsystem, which are mixing digital and analogue processing capabilities to flexibly adapt to the desired Configuration State requirements.

4.2 Execution Environments

Execution Environments are attached to specific software execution units of the RRE, in order to enable flexible execution of Software Execution Modules to provide RRE with software programming reconfiguration.
On top of the processor hardware and the accessible peripherals, Execution Environments are usually providing a more or less elaborated of software services for Software Execution Modules support (e.g. an operating system, a middleware, mathematical processing libraries, etc.).
In terms of execution units, the scope of possible Execution Environments is varying from workstations processors to highly embedded processing cores such as low power dedicated DSP units.
In terms of programming languages, the scope varies from high end object oriented languages to assembly code.
The software services available are varying very much as well. Considering the Operating System case, things can range between Windows or Linux based processing environments down to embedded micro-cores with no operating system support due to memory restrictions.
It shall be mentioned that FPGA development is considered as a relevant family of execution environments for software programming reconfiguration, despite its specificities compared to “genuine” software paradigms.

4.3 Software Execution Modules

The Software Execution Modules are participating to implementation of the selected Configuration State, executing in accordance with the development environment and constraints of the hosting Execution Environment.
The diversity existing in Execution Environments is directly impacting the possible kinds of Software Execution Modules.
Component-based programming is acknowledged to play a increasingly important role for high-end generations of Execution Environments and Software Execution Modules.
Important stakes are related to software qualities attached to Software Execution Modules, which are often desired to exhibit good portability and re-usability features in order to increase cost-efficiency thanks to development costs savings. Those aspects are a key enabler of overall economical efficiency of SDR and CR solutions.

5. RECONFIGURATION SUPPORT DATA (RSD)

Reconfiguration Support Data (RSD) are characterizing the complex information data structures exchanged among RM and CEM.
Reconfiguration Interfaces are the natural complement to Reconfiguration Support Data, characterizing through messages the interactions between RM and CEM.
Capturing an information exchanged between modules as Reconfiguration Support Data or as an argument of a Reconfiguration Interface message depends on the complexity of the information.
Simple information are put as arguments of messages, while complex information are captured as self-standing Reconfiguration Support Data. The associated messages are then containing references to the Support Data.

5.1. Cognition Support Data

This category covers the data specifically supporting the interactions between Reconfiguration Modules.

5.1.1 Policies

Policies are influencing the behavior of Reconfiguration Modules. They are not part of the I/O flow of the Reconfiguration Module.
Policies can influence any sort of RM, since policy-varying behavior is not limited to Decision Making. For instance, a Reconfiguration Support Capability may behave differently according to applicable policies.
Policy usage is not mandatory, they are heavily depending on the nature of the RM. Simplest ones will not use policies.

5.1.2 Profiles

Profiles are viewed as the I/O flow support data attached to Reconfiguration Modules.
Any sort of RM can use profiles, for instance a Context Awareness module can regularly provide as an output a refreshed version of a profile of Free Frequencies.
Profiles usage is not mandatory, simplest RM will not take Profiles as I/O.

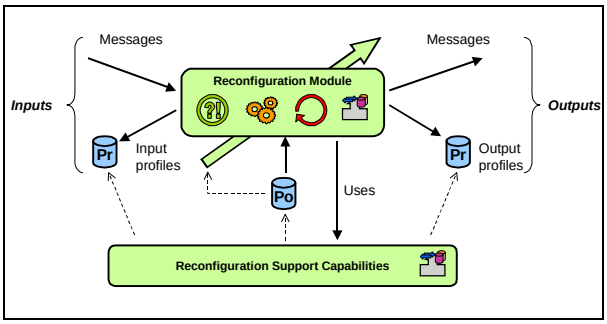

Figure 6: Policies and Profiles among Reconfiguration Modules

5.2. Configuration Descriptors

5.2.1. Functional Configuration Descriptors

§ 2.5 has underlined the importance of functional level of abstraction for operation of CMM (the RM operating under fully-functional considerations). Functional Configuration Descriptors are the data constructs formally capturing such functional abstractions, in case complexity of the concepts prevents usage of arguments in messages.

One usage of such information can be to progressively elaborate, through successive modification steps, to define through a Functional Configuration Descriptor the applicable functional Configuration State.
This description can then be used, by next stages (typically CCM modules) in order to transform the Functional configuration into a Physical configuration.

Formalization of such data constructs is the purpose of Functional Description Language (FDL).

5.2.2. Physical Configuration Descriptors

Physical Configuration Descriptors are data constructs formally capturing a physical configuration applicable in the RRE.
This typically enables automated exploitation of such formal inputs. The grammar associated to those description can be specific to a RRE, or based on standards.
The degree of precision in the description can vary from one Reconfiguration Chain to another.

Typical usage of Physical Configuration Descriptors are the SCA or SWRadio Specification XML files (application and platform descriptors) used by Core Frameworks to set up the desired applications.

5.3. Execution Support Data

5.3.1. Object codes

Object codes are the data constructs capturing the software program to be executed in an Execution Environment.
They are well-known artifacts of programming-based reconfiguration. Once executed into a given Execution Environments, they are giving birth to the corresponding Software Execution Module.
The object codes are generated by the programming environment attached to the considered execution environment. Executable files or dynamically linkable libraries are flagship examples of object codes.

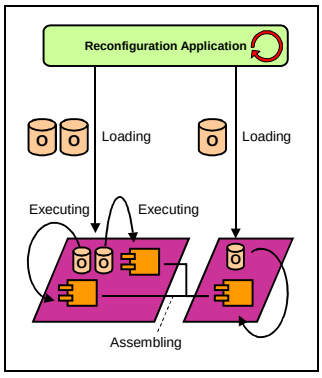

Figure 7: from Object Codes to Software Execution Modules

The object codes can be statically installed in a RRE at factory output, while participating to SDR capabilities.
More advanced CR-oriented features are considering remotely upgraded or retrieved object codes enabling

advanced capabilities such as bug fixes or operation and maintenance upgrades.

6. FORMALIZATION

6.1 Graphical conventions

Specific graphical conventions are defined by the Framework as support to the concepts introduced.

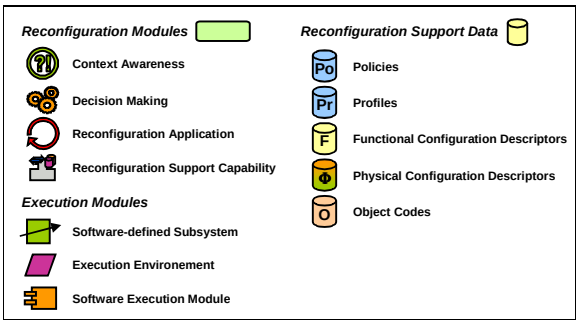

Figure 8: Graphical conventions

They have appeared as icons alongside the figures of this document.

6.2 UML profile

A basic UML profile is defined as a support for the Framework usage in modeling environments.

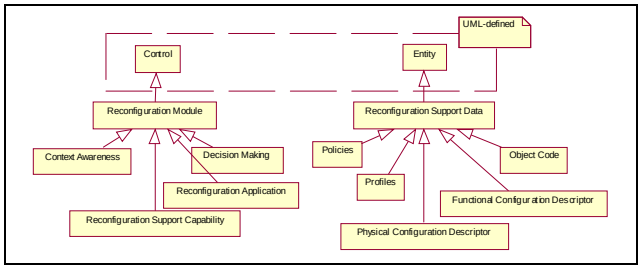

Figure 9: Formal stereotypes definition

Are specifically introduced:

- *Reconfiguration Module* as a virtual stereotype inheriting from the UML standard *Control*; *Context Awareness*, *Decision Making*, *Reconfiguration Application* and *Reconfiguration Support Capability* as stereotypes inheriting from *Reconfiguration Module*.
- *Reconfiguration Support Data* as a virtual stereotype inheriting from the UML standard *Entity*; *Policies*, *Profile*, *Functional Configuration*, *Physical Configuration*, *Object Code* as stereotypes inheriting from *Reconfiguration Support Data*.

Further developments of the Framework will identify richer constructs and classifiers.

7. APPLICATION TO STANDARDS DISCUSSION

7.1 Existing standards characterizations

7.1.1. SCA & SWRadio Specification

The JTRS SCA [2] and OMG SWRadio Specification [3] are SDR standards, originated in the defense radio industry domain, which define similar reconfiguration infrastructures, based on the Core Framework concept.
Core Frameworks are standard Reconfiguration Application solutions, operating the deployment of distributed waveform applications, on top of standard Physical Configuration Descriptors (the XML platform and application descriptors).
Core Framework are parsing and interpreting the XML descriptors of the selected application. Those are prepared at development time.
The Configurable Execution Modules are accessed through standards physical abstraction interfaces: Resource (for Software Execution Modules), Executable Devices (for Execution Units) and Devices (for the Software Defined Sub-systems).

7.1.2 OMA Device Management (DM)

The OMA (Open Mobile Alliance) Device Management [4] is a cellular phones industry standard, which facilitates the management of the software available into cellular phones of a mobile environment. The solution is based on any over-the-air HTTP solution supported by the terminals. The type of managed software modules is indifferent.
OMA DM is thus a standard Reconfiguration Support Capability, enabling to obtain Reconfiguration Support Data from over-the-air operation.
The useful data then depends on the implemented Reconfiguration Chains. Object Codes updates for patching and bug fixing of the Software Execution Modules is one evident example of application.

7.1.3 P1900.4

P1900.4 activities [5] are addressing collaborative mechanisms and protocols supporting CR decision making at terminal level. Those protocols and the associated data constructs are subject to give birth to artifacts the Framework characterized as Reconfiguration Support

Capabilities, Context Awareness and Decision Making Modules. An overview is provided in [9].
Specific Reconfiguration Support Meta-data and Reconfiguration Support Interfaces may emerge from this standardization effort.

7.1.4 Transceiver Facility & OMG Digital IF

The SDR Forum activity undertaken to promote standardization of a Transceiver Facility, based on works presented in [6], is studying pre-standardization issues for APIs towards a Transceiver Software-defined Subsystem.

7.2 Standardization perspectives

7.2.1 Functional Description Language (FDL)

Functional Description Language (FDL) is a promising technology to formalize Functional Configuration Descriptions through structured data constructs.
Such descriptions can typically characterize Configuration States expected to be implemented by the RRE. From the early studies presented in [7], the question is now opened to progress on the definition of a formal standard.
A strong correlation is potentially existing with the modeling profiles subject to support Platform-Independent Modeling of waveform applications, since the meta-data associated to a UML or SysML PIM of a waveform application are containing exactly the same sort of information than a FDL file.

7.2.2 Cognition Support Data

Cognition Enablers (cf. § 5.1), are largely used in many research and pre-development efforts. Emergence of standard structuring concepts for Policies and Profiles would facilitate the emergence of associated tools and help to avoid redevelopment of basic handling services of such support Data.

7.2.3 Embedded Component Models

Standard component models for very constrained embedded and real-time Execution Environments (e.g. DSP and FPGA) are not yet a reality.
Definition of formal standards providing solutions for efficient DSP and FPGA component models would facilitate technological adoption of SDR into such constrained environments. The work on DSP micro-Framework [8] is typically progressing towards future standardization of such efficient implementation solutions.
Conversely, several approaches based on maturation of CORBA technologies are preparing CORBA-based component models for such kind of environments.

8. ACKNOWLEDGEMENT

This work was performed in project E^{2}R II [1] which has received research funding from the European Community's 6th Framework program. This paper reflects only the authors' views and the Community is not liable for any use that may be made of the information contained therein. The contributions of colleagues from E^{2}R II consortium are hereby acknowledged.

9. CONCLUSION

This paper has presented in details the first structuring concepts of an Architecture Framework for characterization of future SDR, CR and CSR products. A formal UML profile to support modeling is defined. Standardization cases are discussed from the Architecture Framework perspective, enabling to foresee how the proposed set of concepts can facilitate expression of recommendations.
Usage of this Framework will hopefully help convergences and facilitate synergies in the SDR and CR research and industrial community.
In further steps, the Framework will be enriched in order to characterize richer items subject to facilitate technological readiness of SDR and CR. The European Community's 7th Framework E3 project, which contract elaboration is being finalized, is foreseen to be one key contributor to the Architecture Framework development.

10. REFERENCES

[1] IST project E²R II, https://e2r2.motlabs.com/
[2] Software Communications Architecture (SCA), http://sca.jpeojtrs.mil/
[3] OMG PIM & PSM for Software Radio Components http://www.omg.org/technology/documents/formal/swradio.htm
[4] OMA Device Management download page: http://www.openmobilealliance.org/release_program/dm_v1_2A.html
[5] P1900 http://grouper.ieee.org/groups/emc/emc/1900/4/index.htm
[6] The Transceiver Facility Platform Independent Model (PIM): Definition, Content and Usage, *Eric Nicollet*, SDR Forum Technical Conference, Orlando FL, November 2006.
[7] Performance Evaluation of the Functional Description Language in a SDR Environment, *Shi Zhong, Craig Dolwin, Klaus Strohmenger, Bernd Steinke*, SDR Forum Technical Conference, Denver CO, November 2007.
[8] A DSP Micro-Framework (DSP µF) for OMG SWRadio Specification Extension, *Frédéric Lafaye, Eric Nicollet*, SDR Forum Technical Conference, Denver CO, November 2007.
[9] IEEE P1900.B: Coexistence Support for Reconfigurable, Heterogeneous Air Interfaces, *Markus Muck, Soodesh Buljore et al.*, IEEE DySPAN 2007, Dublin, April 2007