A Survey of Frameworks & Open Environments Applied To Cognitive Radio Design

F. Le Roy¹, R. Lababidi², A. Mansour³

Lab-STICC, ENSTA Bretagne
2 Rue François Verny, 29200 Brest, France

Abstract—In modern societies, spectrum bands become very expensive and highly requested resources. Many applications based on wireless communications require smart collaboration and wise sharing of available resources. To achieve this goal, concepts of cognitive radio have been recently formulated. Even though, these systems are newly proposed, many researchers are becoming enthusiastically involved in the development of such concepts and implanting them in communication devices. In addition, several recent IEEE wireless communication standards are proposed. Major software environment, framework and cognitive radio concepts are discussed in this manuscript.

Keywords—Software and Cognitive Radio; Software Radio Environment; Framework, Middleware.

I. INTRODUCTION

In the last two decades, radio spectrum resources have become increasingly scarce. However, most of spectrum bands are used sporadically which demonstrates a poor spectral efficiency. To access temporary released spectrum bands (or weakly occupied), new opportunistic access paradigms have recently been introduced. According to these paradigms, cognitive terminals can access unused licensed sub-bands. Hereinafter, primary users represent the users who are holding the license of a spectrum band, and secondary users are other users which are trying to use available licensed spectrum bands.

The concept of the cognitive radio was firstly introduced in the pioneer works of J Mitola III [1]. According to Mitola, these communication systems can automatically and dynamically adjust their radio operator parameters to adapt to their transmission environment. Mitola [2] showed that the cognitive radio operator devices should implement a six-phase cognitive cycle as follows:

- The observation phase: During this phase, the system should analyze various physical environmental variables related to electromagnetic wave propagation parameters, geographical positioning, meteorological measurements, etc.
- The orientation phase: It evaluates the importance of the observation measurements according to a database and possible stimulations of the environment.
- The planning phase: It returns all possible solutions by taking into account equipment specificities and a priori constraints
- The decision phase: As its name indicates, it makes operational decisions by informing or choosing among predetermined applied scenarios.
- The action phase: In this phase, the device will be actively reacting with its environment.
- The final phase is a training phase: It can affect the next decision phase according to the outcomes of new observed results.

For many military as well as civilian applications, cognitive radios are considered as advanced solutions to take advantage of available unused spectrum bands. To achieve this goal, new standards have been recently emerged, such as 802.22 [3] along with 802.11h [4] (which is used in uplinks of weather satellites). However, the 802.11h standard is only based on cognitive function such as DFS (Dynamic Frequency Selection) of the standard 802.22. Therefore, it can avoid in European countries the interferences among 802.11h terminals and radiolocation radars by monitoring transmitted power mechanisms. For example, as it was described by Ian F. Akylidiz et al. in [5], spectrum in cognitive networks can be share between primary users (licensed) and secondary users (unlicensed). When a radio channel is unused by a primary user, a secondary one can then use spectrum resource opportunistically.

This paper is organized as follows: Section II introduces cognitive architecture concept and it underlines on the impact of cognitive cycle on software architecture. In the next section, we present a rapid description of four of the most software radio environments discussed in literature. Section IV presents a comparison of these environments at different levels of abstraction. Section V concludes this work.

II. COGNITIVE ARCHITECTURE

Cognitive radio systems are based on software radio and software defined radio concepts Error! Reference source not found.. The major ideas of these concepts consist in reducing the dedicated hardware parts of the terminal and replacing them by software codes to add flexibility to heterogeneous hardware/software platforms.
Software-defined radio (SDR) architecture is a composition of hardware resources associated to a software environment. These compositions form a communicating nodes that implements on one or more flexible network stacks. On these devices, software environment maps necessary processing resources on available nodes of the target platform.

As shown in Figure 1, the cognitive layer added to the SDR platform monitor a spectrum database where the transmission can be associated to a frequency band. This allocation method is controlled by the “behavior policy” and “user requirements”. The behavior policy ensures the cognitive transmissions with some regulations rules. At the end of the loop analysis process, cognitive terminals deploy the better waveform that can be executed on an SDR platform under the radio channel constraints.

The transmission link quality can be evaluated using statistics generated at each stack layer (bit and frame error rate, bandwidth, etc.). This QoS (Quality of Service) metrics provide feedback to the cognitive parts of the platform. The deployment of component on specific nodes should take into consideration in real time the platform capacity.

Cognitive extension of an SDR platform adds a big constraint on open environment flexibility. For example, during a communication procedure, embedded software architecture must be able to freeze any component without losing the data during the reconfiguration phase. In addition, when this reconfiguration occurs, data synchronization between the transmitter and receiver must be fully insured by the overall system architecture. Many solutions to this problem exist: Adding information at different layer of the network stack.

Or using some specific algorithms (modulation, channel coding, etc.) for which synchronization elements can be hidden (e.g.: rotating constellation). This last solution is very attractive but the impact on the cognitive systems is still under development is out of the scope of this manuscript.

### III. SOFTWARE ENVIRONMENT OVERVIEW

There are different open environments and framework dedicated to software defined and cognitive radio (cf. [7][8]). The major four ones are discussed hereinafter.

#### A. GNU Radio

The “GNU Radio” project [9] has been designed to allow rapid prototyping of software radio application. It is an open source framework under a GNU Public License (GPL). Software applications or waveform are composed of C++ components promoted by signal processing libraries. This composition is implemented using Python. The SWIG compiler [10] is used by “GNU Radio” installation script. It constructs Python "wrapper" for C++ signal processing libraries. Signal processing block instanced by application Python script are configured and interconnected throw a data flow graph. Vertices of this graph are associated to processing element as edges to data flows. The SDF (Synchronous Data Flow) [11] execution model can be used to model the execution of GNU Radio flow graphs. Data communication between processing blocks interfaces is provided by circular buffers mechanism. Access to the scheduler of the execution graph is done by inheritance of the top signal processing block to a specific class (gr.top.block). By default, this instance calls a scheduler for whom a thread is executed at each vertices of the graph. Although this software framework is particularly well suited to run on a processor GPP, but studies have shown that execution could also use GPU [12], DSP [13] or FPGA [14] resources embedded on the SDR platform.

Unfortunately, “GNU Radio” framework has been designed for data stream applications. Then dynamical reconfiguration of software components has not been taken into account. An FPGA experiment [14] shows that dynamic reconfiguration functions of the cognitive layer can then be mixed with processing blocks. Thus, block size exchanged between processing blocks is not easily reconfigurable. So, packets driven layers of OSI (Open Systems Interconnection) model (with variable size) are very difficult to manage. However, in the context of cognitive radio this type of situation frequently occurs in different parts of the OSI model.
ADROIT project (Adaptive Dynamic Radio Open-source Intelligent Team) [15] had produced a specification of the concept of “m-block” [16] but nobody implement it in “GNR Radio” framework. In the networking context, layers above physical ones are mainly oriented to packet exchange. Consequently, the “GNU Radio” software framework is poorly suited. However, the ADROIT [15] and Hydra projects [17] showed that the association of GNU Radio (for physical layer) with Click Modular Router [18] (for layer media access) can make a MAC-PHY layer functional.

A final drawback of this framework is lack of tools for estimating and measuring real-time processing capabilities of the execution platform. In fact, every block performs its processing task without being aware of application and platform timing constraints. Then functionality test of an application can just be made at a very late time in the design process.

B. OSSIE

OSSIE (Open Source SCA Implementation Embedded) [21] is an operating environment (OE) that implements an SCA (Software Communication Architecture) core framework (CF) v.2.2 [19]. This OE provides all necessary services for execution and deployment of SCA waveform (component configuration, lifecycle control, etc.) on a given platform.

In the SCA OE, between POSIX operating system and the CF, an implementation of CORBA [20] middleware must be inserted. This one manages the interoperability of components distribution over heterogeneous platforms. The operating system provides multitasking for the deployment and execution of applicative components (or resource) on platform component (or device).

OSSIE is a GPL project from the “Wireless” laboratory of Virginia Tech University. It is based on omniORB [22], a GPL implementation of a CORBA object request broker. OSSIE consists of a set of tools to build, execute and debug SCA waveforms. One of the main tools of this chain is the Eclipse plug-in OEF (OSSIE Eclipse Feature) that can generate SCA components wrapper, combine component to form waveforms and deploy this one on platform nodes. After that, XML files of the waveform have been generated and can be interpreted by the executable “NodeBoo ter” to start all managers (DeviceManager, DomainManager, etc.) require by waveform to make it work.

OSSIE has the great advantage to be based on a standard definitely suitable for deployment of distributed applications on heterogeneous architectures. Unfortunately, studies [23][24] showed that CORBA’s introduce a high latency in the execution of SCA waveform execution. Indeed, number of components and interfaces dramatically increases this latency. Since publication of these works, SCA Next [25] evolution had also made CORBA optional. The objective of CORBA was to ensure the interoperability of software components on heterogeneous platforms. However, as we indicated in [26] : code generators, ORB settings, multithreading libraries and non-standard bus can make code un-portable to another platform. As Figure 2 shown, “FlexFabric” bus use makes the waveform specific to “Spectrum Signal” platforms.

This kind of interconnection limits drastically the portability of the waveform. But, not using it, often deport timing constraints on non-CORBA component. To ensure code portability, platform components must meet API standard governing the exchange of data between application components and platform components. For example, modems are often implemented on non-CORBA component (DSP or FPGA device). The platform component responsible for performing this kind of function must then be in accordance with standard API like MHAL (Modem Hardware Abstraction Layer) [27].

A. Cormier’s works [28] have shown that dynamic reconfiguration can be added to the OSSE’s runtime and debugging environment. XML files are then regenerated on the fly, then application is stopped, old configuration is unloaded and reloaded before restarting the new application. The latency introduced during the reconfiguration phase transition does not consider the time reaction of the cognitive loop!

Furthermore in waveform design process, no computation model is defined. So, this advantage facilitates integration of tool chains adapted to this applicative field. However, this advantage also means that no performance management mechanism is proposed by this software architecture. For this reason, Q-SCA, described in [29], is an implementation SCA in which the computation model is fixed. Then, allocation and scheduling algorithms allow latency and throughput optimization.

C. ALOE

The ALOE (Abstraction Layer and Operating Environment) [30], formerly PHAL-OE (Platform and Hardware Abstraction Layer Operating Environment) is a software radio environment developed under the GNU license by Polytechnic University of Catalonia (UPC).
It is based on an abstraction layer that allows running and deploying software radio applications on heterogeneous platforms. ALOE has been designed to take into account cognitive radio’s constrains. For this reason, this environment is aware on real-time computations running on platforms and is able to optimize real-time sharing processing resources. As GNU Radio, the “Synchronous Data Flow” computation model is used. Its architecture is based on software components written in C++. Proof of concept has been carried out for x86, ARM5, ARM7, and GPP processors, for 67xx and 64XX TI DSP and for XILINX FPGA (Virtex family).

“Surfer” environment [31], from the University of Indiana, seems to be closer to ALOE. Unfortunately, very few information is available from it and sources are not available under a free license.

D. Iris

“Iris” is a software development environment developed within the Irish research project [32]. Unlike the previous three environments, “Iris” has definitely been designed with the aim of software nodes design for cognitive networks. As a result, particular care was focused on model of computation (MoC) provided by this environment. Indeed, depending on waveforms, abstraction level of components, flexibility and performance requirement are not identical. Then, three runtime engines come with this environment:

- The first one: sPHY (scheduled PHY Engine) is suitable for relatively static component composition (with low reconfigurability needs) with high rate processing constraint. For this kind of waveform design, SDF is the most suitable MoC to use. Here, data-passing mechanism is used with the help of buffer techniques at each component interface.
- The second one: fPHY (flexible PHY engine) which is suitable for flexible component design uses the PN (Process Networks) [33] MoC. The data-passing mechanism is also achieved by a buffers mechanism.
- The third one: NET (Stack Engine NETwork) is used for the layers above physical layer. This engine uses the same MoC as fPHY engine. Unlike previous the two engines, data communication is message passing. To optimize footprint this engine uses zero copy message to exchange packet between processing component.

Unfortunately, few studies provide quantitative results on throughput, latency performances.

The other interesting technology introduced in [34] is virtual machine. This one offers attractive possibilities in software-defined and cognitive radio. Indeed, waveform is no longer compiled but interpreted.

Then waveforms running on one device can naturally be ported to another. In addition, the underlying dynamic evolution of physical environment in cognitive context is taken into account natively by software architecture. However, this option defers system’s complexity on virtual machines, OS module and API (transceiver and modem) design. Unfortunately, there are very few studies in literature on latency, throughput and jitter of this kind of hardware and software combination.

IV. SOFTWARE ENVIRONMENT OVERVIEW

A software radio waveform is composed by an assembly of processing resources deployed on software-defined radio device targeted. A comparison of software radio environments can study at different points of view as described hereinafter.

A. The application designer point of vue

TABLE I. shows that OSSIE uses only standardized interface to design waveforms with better portability. In fact, nothing prevents an SCA OE (Open Environment) to use its customs interfaces while remaining standardized. Waveform can be no longer portable if these custom interfaces are not open source!

<table>
<thead>
<tr>
<th>Application Design</th>
<th>Framework or Open Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>GNU Radio</td>
</tr>
<tr>
<td>Application language</td>
<td>Python</td>
</tr>
<tr>
<td>Component language</td>
<td>C++</td>
</tr>
<tr>
<td>Component interface</td>
<td>custom</td>
</tr>
<tr>
<td>Model of computation</td>
<td>SDF</td>
</tr>
</tbody>
</table>

The SDF MoC used by most of these environments is very well adapted to the modeling of physical layer. Indeed, in this kind of application, generation process of data is globally synchronous like a modem design. In contrast, for higher abstraction layers, inter-component communications becomes more asynchronous. Then communication model became globally asynchronous but locally synchronous. The MoC suited to this kind of model is the PN MoC proposed by Iris.
TABLE II.
THE HARDWARE ABSTRACTION LAYER DESIGNER POINT OF VUE

<table>
<thead>
<tr>
<th>Platform Design</th>
<th>Framework or Open Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Device</td>
<td>GNU Radio</td>
</tr>
<tr>
<td>OS</td>
<td>Linux</td>
</tr>
<tr>
<td>Middleware</td>
<td>no</td>
</tr>
<tr>
<td>GPP</td>
<td>yes</td>
</tr>
<tr>
<td>GPU</td>
<td>Nvidia</td>
</tr>
<tr>
<td>DSP</td>
<td>Omap</td>
</tr>
<tr>
<td>FPGA</td>
<td>partial</td>
</tr>
</tbody>
</table>

a: Cell architecture is a close cousin of GPU’s one.

C. Operating waveform point of vue

From the operating waveform point of view (see TABLE III.) software environment described above do not offer the same features.

TABLE III.
OPERATING WAVEFORM POINT OF VUE

<table>
<thead>
<tr>
<th>Waveform Deployment</th>
<th>Framework or Open Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Deployment Language</td>
<td>GNU Radio</td>
</tr>
<tr>
<td>none</td>
<td>XML</td>
</tr>
<tr>
<td>Deployment Manager</td>
<td>none</td>
</tr>
<tr>
<td>Property config.</td>
<td>dyn. a</td>
</tr>
<tr>
<td>Dynamical config.</td>
<td>partial</td>
</tr>
<tr>
<td>Multi-waveform</td>
<td>no</td>
</tr>
</tbody>
</table>

a : dynamic

For example, GNU Radio hasn’t got deployment manager. Deployment and mapping process of processing resource on devices is just possible through application and dedicated libraries (on library to each devices embedded by the platform). Dynamic reconfiguration of the application can also be made through the application.

OSSIE is based on the rich SCA management services of the platform. Thus, dynamic configuration of waveform is also possible through the surrounding its execution and debugging tool ALF. However, deployment of a waveform on other devices can be achieved by manually creating a second deployment model. Unfortunately, reconfiguration time of CORBA messages can’t be taken into account by the environment. It is almost impossible to dynamically load a component without breaking communication service.

Compared to GNU Radio and OSSIE, AOLE’s implementation is scalable at runtime according to processing load of the platform. However, compared to OSSIE, it is currently unable to handle simultaneous execution of several waveforms on the same platform.

Unfortunately, for this point of view, information is not available for “Iris” architecture.

V. CONCLUSION

In this manuscript it has been shown that architecture of cognitive terminal can rely on a software defined radio terminal. This one must then be able to execute a set of flexible services organized in abstract layers (see OSI Open Systems Interconnection, model). Cognitive layer should permit configuration services’ trigger and/or configuration of flexible layers of the network model. However, it must remain aware of the impact of its actions on the flexible stack and on platform load. Statistics must be accumulating at different abstraction level of SDR platform. Quality of transmission (on RSSI, BER, FER, ect.) and real-time scalability of the flexible stack must be took to allow the platform to be aware of its radio environment and platform capabilities. Unfortunately, the theoretical capacity displayed by a platform is often far from its actual capabilities. Indeed, the presence of operating system, middleware and environment or framework decreases the effective capacity of the platform. In addition, the impact of round trip time of dynamic reconfiguration on some networks layers (PHY, MAC) can be extremely difficult to take into account. Many efforts must then be worn on the development of tools for software and cognitive radio design. In our future work, we will focus on the flexibility of MAC / PHY layers for mobile wireless sensor networks mobile in uncertain environments radio channel.
International Journal of Emerging Technology and Advanced Engineering

REFERENCES


Frédéric LE ROY was born in Landerneau, France, in 1971. He received the Master and Ph.D. degrees in electronic from the University of Brest, France, in 1995 and 2000 respectively. He spends his Ph.D period at the actual PRACOM laboratory of Ecole Nationale Supérieure des Télécommunication de Bretagne (ENST Bretagne, France). His research was made in collaboration with the international company France Telecom R&D on High Level Synthesis methodologies for asynchronous circuits design.

In 2000-2001 he worked at Institut Supérieure de l’Électronique et du Numérique (ISEN Brest) for five where he was involved in industrial few projects of telecommunication and was the principal advisor for the bachelor grade.

Since 2006, he has been an associate Professor at Model Driven Engineering team of Ecole Nationale Institute des Techniques Avancées, Brittany (Ensta Bretagne), Brest, France, where he teaches computer science engineering and telecommunication. Actually, his research interests are in high performance circuits design and performance analysis for software and cognitive radio design. Dr Le Roy managed and participated in several Research and industrial projects

- COFORALOG
  “Conception de Forme d’Onde pour la RAdio Logicielle”. Industrial project on FM3TR waveform portability study on heterogeneous platform with Software Communication Architecture (SCA), ENSTA Bretagne, with DGA MI (French DoD).

- MOPCOM
  “Modélisation et spécialisationOn de Plates-formes et Composants MDA pour SOC/SOPC”, French research program (RNTL) on MDA for Embedded system, with THALES, THOMSON, Soius, ENSIETA, Supelec/IETR, INRIA/IRISA.

- RENAR.
  “Récepteur Numérique à Architecture Reprogrammable”. Industrial project on generics components design for IESS-308 waveform implementation on FPGA farms, with DGA MI (French DoD).

- IRIS+.
  “Réalisation d’un système complet de réception d’images en provenance du satellite MSG (Meteosat Second Generation)”.

Industrial project of a ground station design for the reception of French weather images, with INTA.

- RMIZER.
  “Spécification d’un processeur de contrôle adapté à la gestion du trafic circulant sur un réseau à haut débit”, French research program on Asynchronous Transfer Mode network controller design, with France Telecom R&D.

Raafat LABABIDI was born in Tripoli, Lebanon, in 1983. He received the Master and Ph.D. degrees in high-frequencies and optical telecommunications from the University of Limoges, France, in 2006 and 2010 respectively. During his doctoral studies, he was with the XLIM Research Institute C2S2 Department in collaboration with the international company Technicolor (formerly Thomson Multimedia) R&D Laboratory (Rennes-France), involved with the design of low cost tunable active filters for future Multistandard terminals.

In 2011, 2012 he was a postdoctoral researcher at the Institute of Materials, Microelectronics and Nanosciences of Provence (IM2NP) which is a common laboratory between University of Toulon and University of Aix-Marseille, France, working on the design of RF circuit for RFID and UWB applications. Since 2013, he has been an associate Professor at the department of Model Driven Engineering, National Institute of Advanced Technology, Brittany (Ensta Bretagne), Brest, France, where he teaches courses in RF and Microwave Engineering.

His research interests are in the pursuit of new concept to design fixed and reconfigurable frequency RF circuits for advanced communication systems. He holds 10 international patents and is a regular reviewer for the IEEE Transactions on Circuits and Systems II and for many international conferences in the field of RF and analog integrated circuits.

Ali MANSOUR received his M.S degree in the electronic electric engineering on September 1992 from the Lebanese University, his M.Sc. and Ph.D. degrees in Signal, Image and Speech Processing from the "Institut National Polytechnique de Grenoble - INPG (Grenoble, France) on July 1993 and January 1997, respectively, and his HDR degree (Habilitation a Diriger les Recherches. In the French system, this is the highest of the higher degrees) on November 2006 from the Universite de Bretagne Occidentale -UBO (Brest, France). His research interests are in the areas of blind separation of sources, high-order statistics, signal processing, passive acoustics, cognitive radio, robotics and telecommunication.
From January 1997 to July 1997, he held a POST-DOCTORAL position at Laboratoire de Traitement d'Images et Reconnaissance de Forme (INPG Grenoble, France). From August 1997 to September 2001, he was a RESEARCH SCIENTIST at the Bio-Mimetic Control Research Center (BMC) at the Institut of Physical and Chemical Research (RIKEN), Nagoya, Japan. From October 2001 to January 2008, he was holding a TEACHER-RESEARCHER position at the Ecole Nationale Supérieure des Ingénieurs des Etudes et Techniques d'Armement (ENSIETA), Brest, France. From February 2008 to August 2010, he was a SENIOR-LECTURER at the Department of Electrical and Computer Engineering at Curtin University of Technology (ECE-Curtin Uni.), Perth, Australia. During January 2009, he held an INVITED PROFESSOR position at the Université du Littoral Côte d’Opale, Calais, France. From September 2010 till June 2012, he was a PROFESSOR at University of Tabuk, Tabuk, KSA. He served as the ELECTRICAL DEPARTMENT HEAD at the University of Tabuk. Since September 2012, he has been a PROFESSOR at Ecole Nationale Supérieure de Techniques Avancées Bretagne (ENSTA Bretagne), Brest, France. He is the author and the co-author of three books:


He is the first author of several papers published in international journals, such as IEEE Trans. on Signal Processing, Signal Processing, IEEE Trans. on Wireless Communication, IEEE Trans. On Antennas & Propagation, IEEE Signal Processing Letters, NeuroComputing, EUROSP Journal on Advances in Signal Processing, IEEE and Artificial Life, Robotics, etc.. He is also the first author of many papers published in the proceedings of various international conferences.

Finally, Dr. Mansour was elected to the grade of IEEE Senior Member in February 2006. He has been a member of Technical Program Committees (TPC) in many international conferences. He was chair, co-chair and a scientific committee member in various international conferences. He is an active reviewer for a variety of international journals in different engineering fields. He was a Lead Guest Editor at "EURASIP Journal on Advances in Signal Processing - Special Issue on Signal Processing Methods for Diversity and Its Applications".