APPROVED FOR PUBLIC RELEASE

ESSOR Architecture – Motivation and Overview

LEAD AUTHOR

Christian SERRA
a4ESSOR S.A.S. – France
christian.serra@fr.thalesgroup.com

CO-AUTHORS

ELEKTROBIT – Finland
Pekka HEIKKINEN – pekka.heikkinen@elektrobit.com

INDRA Sistemas – Spain
Rafael AGUADO – ramunoz@indra.es

RADMOR S.A. – Poland
Rafal KOSIUK – rafal.kosiuk@radmor.com.pl

Saab AB – Sweden
Bo GRANBOM – bo.granbom@saabgroup.com

SELEX Communications S.p.A. – Italy
Fabio CASALINO – fabio.casalino@selex-comms.com
Stefano SCALA – stefano.scala@selex-comms.com

THALES Communications S.A. – France
Bruno COUNIL – bruno.counil@fr.thalesgroup.com
Eric NICOLLET – eric.nicollet@fr.thalesgroup.com

Wireless Innovation Forum Technical Conference – December 2010

ABSTRACT

The purpose of this paper is for ESSOR Industries to present the motivations and results of the European Secure SOftware defined Radio (ESSOR) program concerning the definition of the ESSOR Architecture.

The ESSOR Architecture is a Software Defined Radio (SDR) Architecture relying on the already published Joint Tactical Radio System (JTRS) Software Communications Architecture (SCA) and Application Programming Interfaces (APIs). The ESSOR Architecture is a complete and consistent Secure SDR Architecture addressing the European military radio-communications market, and fostering on Waveform portability amongst heterogeneous SDR Platforms.

In the scope of these efforts, this paper provides insight details on the ESSOR Architecture definition, on the following areas: definition of Operating Environments (OE) for DSP & FPGA processing elements, providing scalable architectural approaches between Modem Hardware Abstraction Layer (MHAL) and Common Object Request Broker Architecture (CORBA) based solutions, definition of extensions and additions to the already published JTRS Radio Devices (RD) and Radio Services (RS) Application Programming Interfaces (API).

1. Introduction

The goals of developing a Software Defined Radio (SDR) architecture is to facilitate waveform portability between heterogeneous Software Defined Radio hardware platforms and to foster radio reconfigurability in front of a large set of waveforms.

In front of these goals, the Joint Tactical Radio System (JTRS) [1] has initiated the path, publishing the Software Communications Architecture (SCA) [2], recognized as a de-facto standard by the SDR community.

This paper presents the motivations and results of the European Secure SOftware defined Radio (ESSOR) programme concerning the definition of the ESSOR Architecture, an SDR architecture which extends the Software Communications Architecture on the following areas: definition of the Operating Environments (OE) for DSP & FPGA processors providing scalable architectural approaches between Modem Hardware Abstraction Layer (MHAL) and Common Object Request Broker Architecture (CORBA) based solutions, definition of extensions and additions to the already published JTRS Radio Devices (RD) and Radio Services (RS) Application Programming Interfaces (API).

The ESSOR programme has been established under the umbrella of the European Defence Agency (EDA), sponsored by the governments of Finland, France, Italy, Poland, Spain and Sweden, and awarded by the Organisation Conjointe de Coopération en matière d'ARmement (OCCAR) to the dedicated joint venture Alliance for ESSOR (a4ESSOR S.A.S.) in charge of managing the industrial consortium composed of the following respective National Champions: Elektrobit, Indra, RADMOR, Saab AB, SELEX Communications and THALES Communications.

This document aims to provide an overview of the ESSOR Architecture, and is organised as follow:

- **§ 2** highlights the scope and motivation of the ESSOR Architecture, introducing the concepts of Waveform Application, SDR Platform and APIs,
- **§ 3** explains the concept of Operating Environment, focusing on the complements identified by ESSOR for the DSP and FPGA processing elements,

- **§ 4** identifies the ESSOR extensions for Radio Devices and Radio Services APIs, providing a dedicated focus on the ESSOR Transceiver Subsystem API which addresses the abstraction of a critical functional block of an SDR Platform,

- **§ 5** brings conclusions and perspectives in front of the already achieved milestones into the ESSOR Program,

- **§ 6** summarizes the abbreviations used in this paper,

- **§ 7** identifies the used references,

- **§ 8, as Appendix,** provides a summary of the ESSOR Program and explains how the definition of the ESSOR Architecture fits in it.

2. Scope of the ESSOR Architecture

The ESSOR Architecture is an SDR architectural framework, aiming to establish a "Reference Architecture" scalable to different SDR platform classes and suitable for different implementations.

The SDR Platform is defined as the aggregation of Software and Hardware (HW) Platform, where the Software (SW) Platform is a particular implementation of the ESSOR Architecture, scaled for specific needs, that relies on the HW platform.

HW Platforms are typically characterised by different constraints on size, weight and power (SWAP), processing capacity, RF front-end capability (e.g. simplex/duplex), mainly determined by the operational usage of the Radio Set. Depending on these characteristics, SDR Platforms can be categorised in classes (typically: handheld, manpack, vehicular, naval/fixed, airborne…).

Figure 1 below shows an overview of the composition and the structure of an SDR set, in which the SDR Platform (PTF) provides capabilities to the instantiated waveforms by means of APIs.

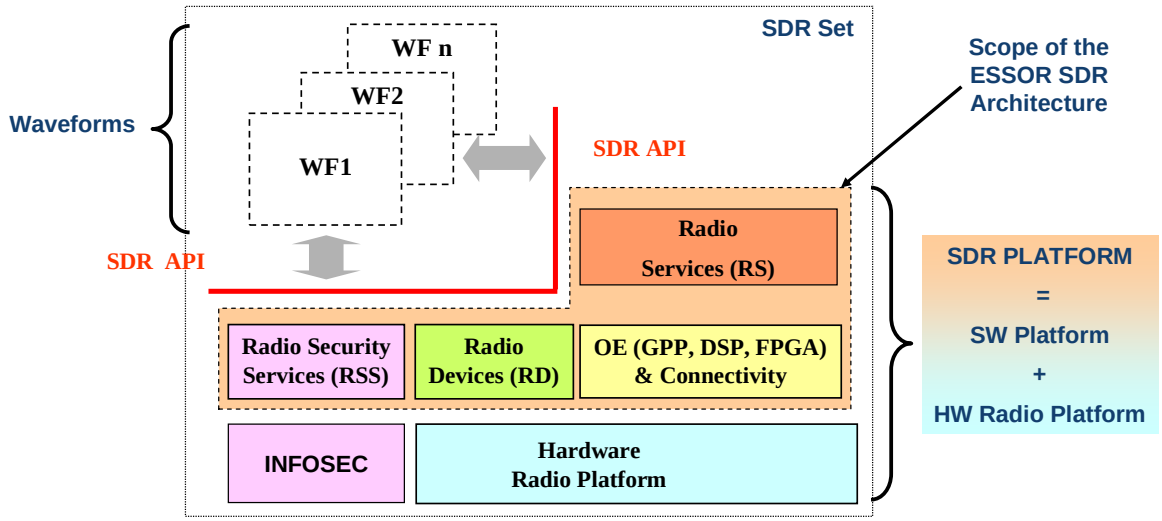

Figure 1 - SDR composition and structure

The definition of the ESSOR Architecture answers to the following motivations and objectives:

- The identification and definition of a full set of capabilities required for the ESSOR Architecture, that are accessible through the defined set of APIs,
- The facilitation of the porting of new waveforms (as the ESSOR HDR waveform) and identified legacy waveforms, on different classes of SDR platforms,
- To be compatible with the SCA 2.2.2 specification,
- To maintain compatibility with JTRS SDR architecture at the maximum level possible.

The ESSOR Architecture is composed of the following areas:

- **Core Framework (CF):** entities implementing JTRS SCA 2.2.2 CF interfaces, to which ESSOR Architecture is compliant, with the addition of some minor modifications, clarifications and extensions.

- **Operating Environment (OE) for GPP, DSP and FPGA:** entities related to Execution Environments (for code loading and execution) and Connectivity for GPP, DSP and FPGA, they are important foundations of the ESSOR Architecture. Two main categories of OE exist, depending on the connectivity solution adopted between the two specified by the ESSOR Architecture: CORBA and ESSOR MHAL. In some cases, the same component is applicable to both execution environments. For the identified OE components, ESSOR Architecture provides rationale, recommendations, specifications and API definition, when applicable.

- **Radio Devices (RD):** entities that provide an abstraction of the HW modules of an SDR. Radio Devices offer a high-level SW interface (API) to the other components of an SDR (e.g. WF application or Radio Services) that need to access HW modules. For RD, ESSOR Architecture provides detailed API definition, relying on published JTRS API specifications when possible, with some modifications/clarifications, and extending them with ESSOR additional functionalities, such as the ESSOR Transceiver API.

- **Radio Services (RS):** entities that provide software functionalities useful for the waveform application. RS are related to the operations of the waveform, its control and monitoring and the download of its files and the properties configuration. Similarly to Radio Devices, ESSOR Architecture provides detailed RS API definition, relying on published JTRS API specifications when possible and extending them with ESSOR additional functionalities.

- **Radio Security Services (RSS):** entities that provide security functionalities in conformance with the security objectives of ESSOR. Presentation of this aspect of the ESSOR Architecture is not in the scope of this document.

The ESSOR Architecture is providing the following benefits:

- **Flexibility:** provides the capability to adapt the implementation of the ESSOR Architecture to different types of hardware architectures, since even for different platforms of the same class, underlying hardware can be very heterogeneous.

- **Scalability:** provides the capacity to select APIs and features adapted to the class of the platform which implements the ESSOR Architecture (tactical, naval, …).

- **WF portability:** provides the capability to minimize the effort required for porting an ESSOR-compliant WF application from an ESSOR-compliant PTF to another one.

3. ESSOR Operating Environments

3.1 Presentation

The notion of Operating Environment (OE) considered in ESSOR Architecture is related to what is needed in a Processing Element (PE) to support waveform components execution[1] and to allow such components to interact with each other.

ESSOR Architecture defines Operating Environments for the following categories of Processing Elements: GPP, DSP and FPGA.

An ESSOR Operating Environment is composed of:

- **An Execution Environment,** in charge of:
  - Deployment Control (code loading and execution) of Waveform Components on a PE
  - Application Environment Profile (AEP) for operating system usage (GPP and DSP operating environments only).
- **A Connectivity,** providing the logical interconnection for deployed waveform components, wherever PE located (for mutual interaction purposes).

Figure 2 below shows two waveform components C1 and C2 running in the execution environment of a Processing Element, allowed to interact with each other (internal connectivity) and with other entities external to the PE (external connectivity).

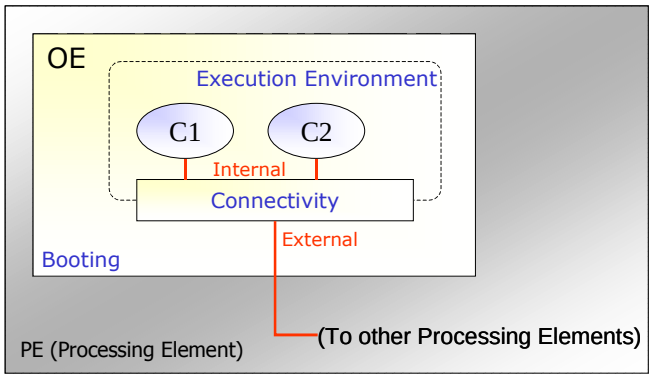

Figure 2 - Operating Environment for GPP, DSP and FPGA Processing Elements

The selected connectivity is of paramount importance for characterization of the OE, since it influences the way all external interactions of executed components are handled: inter-component interfaces, interfaces with radio devices and radio services, and interface with the Core Framework.

In order to cover those interactions, the ESSOR Architecture considers two alternatives:

- Usage of CORBA connectivity for all processing elements (GPP, DSP and FPGA),

[1] Although it is improper the term "execution" applied to FPGA, it is used to refer to the processing or control activity performed by the hardware structures dynamically defined by the FPGA configuration file loaded on this processing element.

- Usage of CORBA connectivity only for GPP and usage of ESSOR MHAL connectivity on DSP and FPGA.

In both cases the GPP operating environments are based on JTRS SCA 2.2.2, as presented in § 3.2.

The first case, also known as CORBA everywhere approach, extends JTRS SCA 2.2.2 towards DSP and FPGA in defining "CORBA Operating Environments for DSP and FPGA", with a driving principle consisting in generalizing the usage of CORBA technology with proper recommendations on IDL and CORBA profiles. Those operating environments are presented in § 3.3.

The second case, also known as ESSOR MHAL approach, extends JTRS SCA 2.2.2 towards DSP and FPGA in defining "MHAL Operating Environments for DSP and FPGA", with a driving principle consisting in the usage of ESSOR MHAL connectivity on DSP and FPGA OE. Those operating environments are presented in § 3.4.

The DSP and FPGA Operating Environments are based on common elements that are presented in § 3.5.

3.2 GPP Operating Environments

In case of GPP Operating Environment, both aspects considered as constituents of OE, Execution Environment and Connectivity, are consistently defined in the SCA 2.2.2 specification, addressing the Core Framework (CF), the Operating System (OS) and the CORBA middleware used as a Connectivity.

As the ESSOR Architecture inherits from the SCA 2.2.2 specification, the ESSOR Architecture identifies only minor modifications, clarifications and extensions related to the CF. A short summary of these points is provided below:

- Optional filling Application interface's attributes (componentNamingContexts, componentProcessIds, componentDevices, componentImplemenations attributes),
- Limitation of pending connections concept defined by the SCA specification to platform components and not for waveform components,
- File system size interpretation clarified as the total capacity of the file system (bytes per sector multiplied by total number of sectors),
- Possibility to use different names for naming context and object name binding during DomainManager's registration in CORBA Naming Service,
- Performing of start() and stop() operations for Devices and Services by dedicated platform entity.

3.3 CORBA Operating Environments for DSP and FPGA

3.3.1 Presentation

A CORBA Operating Environment for a DSP and FPGA is defined as an Operating Environment where the Connectivity used amongst GPP, DSP and FPGA Processing Elements is implemented using CORBA, as illustrated in Figure 3 below. Such environment takes advantage of COTS solutions for DSP and FPGA Real-Time ORB available nowadays, and is resulting in "CORBA-everywhere" solutions.

Topics related to **CORBA Connectivity** are described in § 3.3.2. Other features related to CORBA

Operating Environments are:

- **DSP AEP** (cf. § 3.5.2): provides CORBA DSP OE with a set of operations that WF components have to use to access to Operating System functionalities on DSP, similarly to SCA AEP,
- **Deployment Control** (cf. § 3.3.3): provides the capability to deploy WF components on CORBA DSP and FPGA Operating Environments.

Figure 3 provides an overview of the ESSOR CORBA Operating Environments for DSP and FPGA.

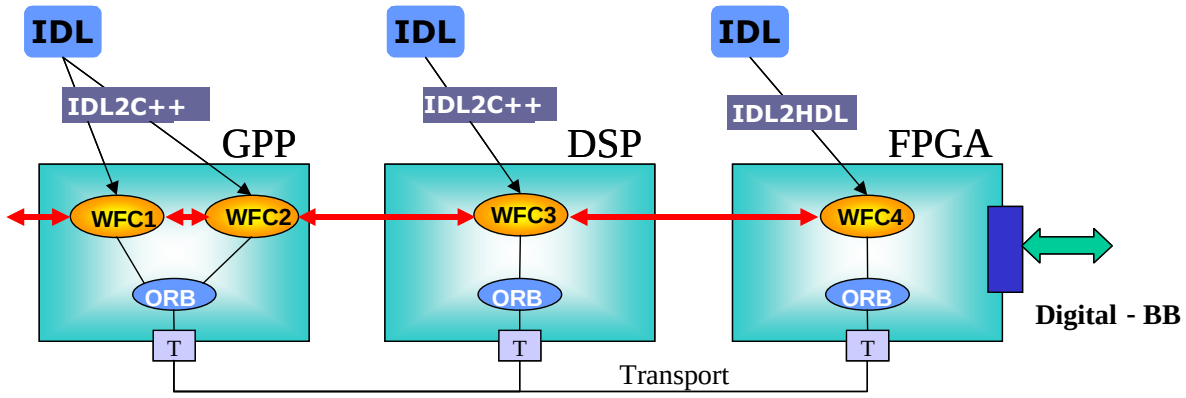

Figure 3 - Overview of CORBA Operating Environments for DSP and FPGA

3.3.2 CORBA Connectivity

The founding principle of CORBA middleware, i.e. the remote operation invocation, allows in particular abstraction of physical locations of interacting objects: the ORB (Object Request Broker) is the mediator of such remote interactions.

The adoption of CORBA as connectivity solution lightens the specification of CORBA Operating Environments, since no additional APIs are required to connect components.

A recommendation is given regarding the adoption of an optimized, highly performing transport protocol under the CORBA middleware, whose choice is platform dependent and is left to the platform developer.

One important topic is related to the question of which CORBA features a DSP or FPGA-based environment should support. Due to the specific processing and data flow throughput regarding WF components usually allocated to these classes of PE, recommendations related to the CORBA profiles to be used on such Processing Elements are given.

The ESSOR specification is referencing for DSP the minimumCORBA profile, in order to maximize backward compatibility with SCA 2.2.2.

For FPGA environments, specific recommendations are as well provided on the usage of a HW-ORB and the features it should support.

One essential recommendation applicable to CORBA Connectivity available for CORBA OE is related to the support of the Common IDL Profile for DSP and FPGA as defined in § 3.5.1 for the definition of waveform-specific interfaces among waveform components. It has to be noted that the SCA CF::Resource

Interface of the waveform component (used for component management purposes) is not constrained by this profile, when required by system design considerations.

3.3.3 Deployment Control for CORBA Operating Environments

The common Deployment Control features defined in § 3.5.3 are applicable to CORBA Operating Environments.

Deployment Control enables waveform components to be deployed in DSP and FPGA compliant with CORBA OE. Implementation of this capability is distributed among the following elements:

- DSP Device and FPGA Device located on GPP OE and reachable by CF,
- DSP_DeploymentControl and FPGA_DeploymentControl, located respectively on DSP and FPGA Processing Elements, that support waveform application components loading and activation.

Specifications for the Deployment Control are defined in the CORBA case for both Static and Dynamic Resource deployment.

Figure 4 below shows the functionalities related to component deployment on different categories of Processing Elements (GPP[2], DSP and FPGA) through the usage of the elements mentioned beforehand.

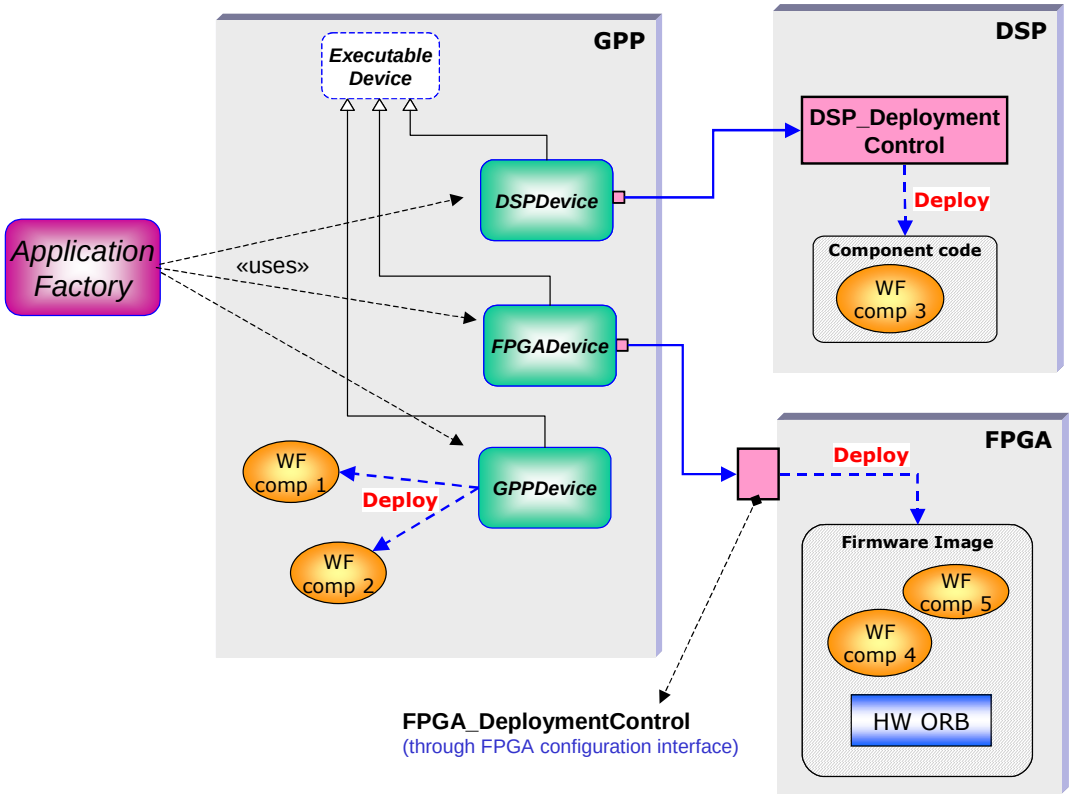

Figure 4 - Overview of Deployment Control for CORBA OE for DSP and FPGA

[2] For completeness, also the case of deployment on GPP through an executable GPPDevice is shown

3.4 MHAL Operating Environments for DSP and FPGA

3.4.1 Presentation

The ESSOR Architecture defines, for DSP and FPGA Processing Elements, MHAL Operating Environments as Operating Environments where ESSOR MHAL Connectivity is used.

It enables implementation of DSP and FPGA Operating Environments in DSP and FPGA processing elements without the support of the CORBA technology, replacing CORBA Connectivity with the ESSOR Modem Hardware Abstraction Layer (MHAL) Connectivity.

The ESSOR Architecture defines the waveform components executing in MHAL Operating Environment as MHAL Resources. All the features necessary to provide MHAL Resources with a similar level of service as with CORBA OE are specified.

The **ESSOR MHAL Connectivity**, compliant with JTRS MHAL, is described in § 3.4.2. Other features of the MHAL OE are:

- **DSP AEP** (cf. § 3.5.2): provides MHAL DSP OE with functionalities similar to SCA AEP. DSP AEP defines a set of operations that MHAL Resources have to use to access to Operating System functionalities on DSP,
- **Deployment Control** (cf. § 3.4.3): provides the capability to deploy MHAL Resources on MHAL DSP and FPGA OE,
- **Resource Support** (cf. § 3.4.4): enables the MHAL Resource to be controlled by Core Framework or other management entities through the SCA CF::Resource interface as if the MHAL Resource was a standard CORBA SCA Resource,
- **RD/RS Access** (cf. § 3.4.5): enables the MHAL Resource to access to Radio Devices and Radio Services which are present in the architecture.

Figure 5 provides an overview of the ESSOR MHAL Operating Environments for DSP and FPGA.

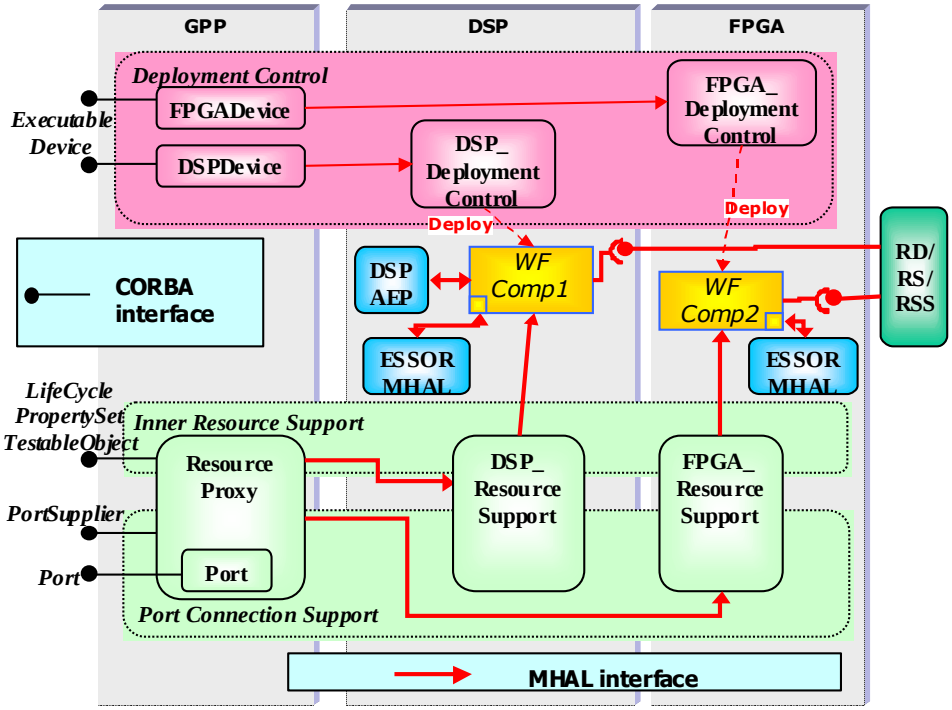

Figure 5 - Overview of ESSOR MHAL Operating Environments for DSP and FPGA

The features Resource Support and RD/RS Access are not specifically addressed for CORBA OE because implementation of SCA Resource interface and connection to Radio Devices and Radio Service is implicit.

3.4.2 ESSOR MHAL Connectivity

3.4.2.1 Presentation

The ESSOR MHAL Connectivity has been defined with the aim to suit the connectivity needs of highly constrained DSP and FPGA processing environments. It provides, together with CORBA, the second connectivity option specified by ESSOR Architecture.

The ESSOR MHAL Connectivity provides an abstract framework for the communication between the software components of the waveform application and the SDR platform. This communication framework makes no assumption on the underlying hardware, being extensible to DSP or FPGA processing elements. The model defines a set of rules in the way the different components of the system have to be defined and connected.

The ESSOR MHAL Connectivity respects the JTRS MHAL Connectivity, as defined in §§ A.2-A.6 of [3], that defines the notions of MHAL Communication Service and associated GPP, DSP and FPGA API extensions. It extends the JTRS specification with the optional Channel API, that is based on the Communications Sequential Process (CSP) model [4], focusing on scalability and WF components synchronization.

Figure 6 below presents how the Channel API complements JTRS MHAL Connectivity, forming the

whole ESSOR MHAL Connectivity specification.

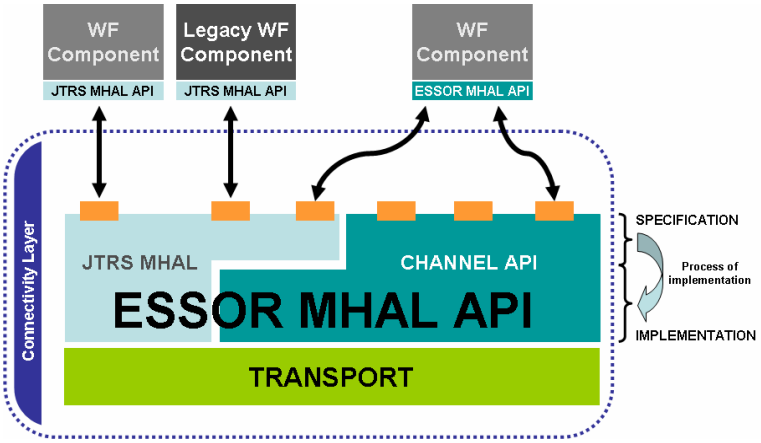

Figure 6 - MHAL Extension abstraction level

The usage of ESSOR MHAL connectivity in order to implement waveform-specific interfaces can follow two approaches (depending on waveform application implementation choices and capabilities of the OE):

- Specify the waveform-specific interfaces according to implementation language engineering practices (C/C++ header files for DSP or signals with chronograms for FPGA) and defining the associated MHAL messages,

- Usage of an IDL compiler and a broker to automatically perform the aforementioned activities (mainly seen as a viable perspective for DSP OE). In that case the compliance with the Common IDL Profile specified in § 3.5.1 is requested.

3.4.2.2 Overall connectivity model

Components from GPP have to be capable of easy connection with MHAL Resources allocated in DSP or FPGA and vice versa. ESSOR MHAL connectivity specification, using the new Channel API connectivity model, enables the management of this interconnection allowing the developer to concentrate on the waveform application. Figure 7 below shows how the communication among MHAL Resources and SCA Resources located in different processors is made onto the communication channels.

The communication channel is defined as the mean by which the source and sink interfaces are connected one to each other. The sink interface is already defined with a LD, therefore the channel identifier is defined at build time in a configuration file where the couple (sourceId, LD) is mapped to a channel identifier, as represented in Figure 7 below.

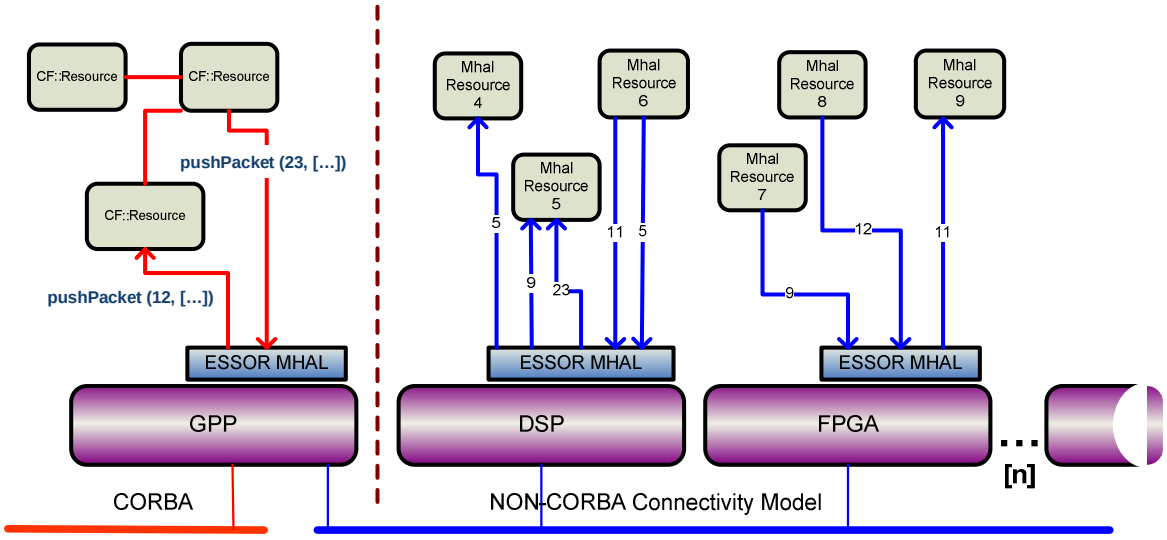

Figure 7 - Overall Connectivity model

3.4.2.3 Features by category of Processing Elements

GPP Processing Elements

The current JTRS MHAL specification defines the MHAL GPP as a Radio Device. In order to improve the performance of the connections between different processors, the ESSOR MHAL specification extends the JTRS MHAL solution, defining two possible approaches when a GPP waveform component needs to exchange messages with a DSP/FPGA waveform component not directly accessible by the ORB:

- **MHAL Device:** the use of MHAL Device is intended as described in the JTRS MHAL specification. In this case, the MHAL proxy located on GPP is modelled as a Radio Device in order to exchange messages with waveform components located in DSP and FPGA.

- **MHAL Access Resource:** the idea behind the inclusion of the MHAL Access Resource into the ESSOR Architecture is to limit the timing overhead due to the presence of Marshalling/Demarshalling and other Mux/Demux activities performed by ORBs, when WF CORBA components not located in the same address space of the MHAL Device want to communicate together. This situation causes that MHAL messages are encapsulated in CORBA messages. This solution implies the redefinition of MHAL Device component as a dedicated Resource component which has to be implemented in each GPP processing element.

As it was shown each solution has its advantages and its disadvantages, therefore the choice of using a MHAL Device or a MHAL Access Resource is an implementation decision that is out of the scope of this document.

DSP Processing Elements

Following the principles defined previously, a waveform application is a collection of one or more concurrently executed MHAL Resources connected by channels. These resources can be allocated both in DSP and FPGA providing, in this way, a unified methodology and model for both kinds of processors.

The MHAL Resources deployed in the DSP processor must match with the API which communicates with the underlying hardware. Each MHAL Resource has a set of sink functions and a set of source functions

that are used to connect these resources together. A MHAL Resource is a black box, communicating with the outside only via its ports. The source functions, therefore, can be connected to sink functions using channels.

MHAL Resources can be treated as atomic building blocks for parallel systems. They can be joined together, rather like electronic components, by connecting their sink/sources with channels. The connection may be implemented either directly in shared memory or across an inter-processor link.

FPGA Processing Elements

As well as the JTRS MHAL FPGA API, ESSOR MHAL FPGA API consists of a collection of transmit and receive node user signal and timing descriptions that provide services to route MHAL communications. MHAL Resources in the FPGA communicate, through ESSOR MHAL component, with MHAL Resources co-located into the FPGA or either in any other remote processing element.

Sink and source functions in FPGA communicate with Rx and Tx nodes respectively. Connectivity based on channels can also be modelled with nodes; therefore one FPGA MHAL Resource may have any number of input channels and output channels. Each channel can be constructed from two buses going in opposite directions, even though data transfer on channels is unidirectional.

The Tx Node defined in the ESSOR MHAL FPGA API provides new capabilities to allow more re-configurability and flexibility. Channels can be modelled with this interface and buses can be used more efficiently since high data, low data or both can be managed. While current definition of JTRS MHAL FPGA is constrained to 16-bit busses, the ESSOR Tx node is able to extend the size bus up to 32-bit. This may allow for certain platforms a more efficient use of busses and also an increase in data working frequency. This bus extension can be configured by user thus keeping backwards compatibility with current JTRS MHAL API FPGA specification.

3.4.3 Deployment Control for MHAL Operating Environments

The common Deployment Control features defined in § 3.5.3 are applicable to MHAL Operating Environments.

Deployment Control enables waveform components to be deployed in DSP and FPGA compliant with MHAL OE. Implementation of this capability is distributed among the following elements:

- DSP Device and FPGA Device, located on GPP OE and reachable by CF,
- DSP_DeploymentControl and FPGA_DeploymentControl, that supports the *MHAL Resource* loading and activation.

On DSP, The MHAL Resource implements an entry function (rscEntryPoint) and MHALResourceInterface interface, in order to be deployed and released within the MHAL OE.

The Deployment Control feature implies the creation of Resource Proxy on GPP and registration of the MHAL Resource to the DSP_ResourceSupport to support the feature described in § 3.4.4.

In the case of Dynamic Resource Deployment as defined in § 3.5.3, the executable code of MHAL Resources is instantiated depending on the desired capabilities, while the executable code relative to the platform is considered as resident in the DSP and FPGA processing elements.

Figure 8 provides an overview of deployment control for MHAL OE.

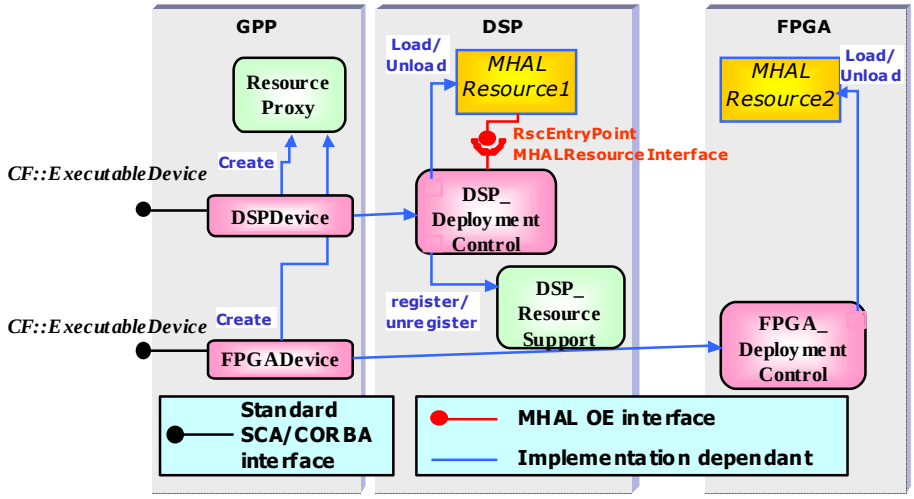

Figure 8 - Overview of Deployment Control for MHAL OE for DSP and FPGA

3.4.4 Resource Support

The Resource Support feature enables the MHAL Resource to be controlled by CF or other management entities through the SCA CF::Resource interface.

This feature is composed of two sub-features:

- **Inner Resource Support:** this sub-feature offers the capacity to forward requests emitted by management entities (CF, platform control, other Resources) relative to the SCA CF::Resource interface to the MHAL Resources located in DSP or FPGA processing elements; this is performed through a Resource Proxy located on GPP, then forwarded by MHAL OE.
- **Port Connection Support:** this sub-feature enables to establish the connectivity provided by the ESSOR HAL Connectivity between the different MHAL Resources and other Resources of the waveform application.

The Port Connection Support sub-feature can be implemented according to different alternatives which enable to locate or to fix MHAL Logical Destinations (LD) at build time or during deployment of the waveform application.

In MHAL DSP OE, relative interfaces are defined in C and C++ language. In MHAL FPGA OE, interfaces are defined as Register Transfer Language (RTL) interfaces.

3.4.5 Radio Devices / Radio Services Access

The Radio Devices /Radio Services access features enable an MHAL Resource to access to RD/RS which are present in the platform which implements the ESSOR Architecture.

Two alternatives for a MHAL Resource to access a given RD/RS are allowed, in order to integrate the target waveform components:

- **MHAL Access:** the MHAL Resource sends and receives MHAL messages to and from RD/RS. This alternative is similar to the solution provided by JTRS for the MHAL RF Chain Coordinator feature.
- **Direct Access:** the MHAL Resource uses a C, C++ (in DSP) or RTL (in FPGA) interface which is the

result of the mapping of RD/RS IDL interfaces defined for each RD/RS.

MHAL Access is corresponding to more separation in the way waveform application components and the RD/RS are encapsulated, while Direct Access first targets performance. The set of possibilities for a given target waveform integration are a platform design decision, and the choice (when allowed) for a given waveform porting is a design decision to occur during porting effort.

3.5 Common elements for CORBA and MHAL Operating Environments

This chapter presents elements that are common to the definition of the CORBA and MHAL Operating Environments for DSP and FPGA respectively presented in § 3.3 and § 3.4.

3.5.1 Common IDL Profile for DSP and FPGA

In order to ease waveform portability, a lightweight common IDL profile has been defined to support definition of waveform-specific interfaces. The usage of this profile is reported for CORBA OE in § 3.3 and for MHAL OE in § 3.4.

The principle of this profile is to narrow down, from the complete set of IDL features, the syntactical and behavioural features possibly used for a given interface to a minimum set. This approach avoids the OE to implement support of features not strictly needed by the waveform components typically running on DSP and FPGA Processing Elements.

The improvements brought in homogeneity of description of interfaces are benefiting to portability of complex waveform applications where components can be implemented, depending on the target platform structure and porting decisions, in DSP or in FPGA. The limitation imposed is not over constraining the design of the waveform application since the limited set of features is well-suited to a large set of real-time and embedded applications, in particular PHY layer processing.

Application of the profile is not preventing portability of a compliant component to GPP Operating Environments that support a larger set of IDL features, since the profile is a strict subset of the IDL defined for usage in GPP OE. The reciprocal is only possible if a component has been defined, in a voluntary manner, in conformance with the lightweight common IDL profile, even though executed into a more capable GPP OE.

In order to ease waveform portability, the adoption of the Common IDL profile for the definition of waveform-specific interfaces among waveform components is one recommended approach for both CORBA and MHAL Operating Environments.

3.5.2 Common DSP Application Environment Profile (AEP)

An Application Environment Profile (AEP) has been defined for usage in CORBA and MHAL Operating Environments for DSP. Similarly to the lightweight Common IDL Profile, the principle has been to define the minimal subset of operating system features that still serves the needs of targeted applications. Implementation of the profile is typically intended to be realized by RTOS (Real-Time Operating Systems), but other OS may comply.

The profile is requiring basic technical services such as Scheduling Services, Memory Management Services and Time Management Services, complying with the “POSIX” definition approach used in the SCA specification.

3.5.3 Common Deployment Control features

The Deployment Control feature provides the capability to deploy WF components on DSP and FPGA

Operating Environments.

From GPP OE perspectives, issues related to components deployment (loading, unloading, execution and termination) have analogous solution in the CORBA and MHAL cases.

The Deployment Control follows the standard SCA rules of interactions between an ApplicationFactory CF entity and a ResourceFactory or an ExecutableDevice, for deploying WF components on DSP/FPGA. Two Executable Devices are defined for this scope:

- **DSP Device:** for deployment on DSP,
- **FPGA Device:** for deployment on FPGA.

Two alternatives are defined by the ESSOR Architecture for the implementation of this feature:

- **Static Resource Deployment:** the entire software running in the processing element (including the Operating Environment software) is deployed, at waveform deploy time, along with the WF components,
- **Dynamic Resource Deployment:** the code of the waveform components is deployed, at waveform deployment time, on top of the Operating Environment software, that is considered as resident in the DSP and FPGA processing elements (started at platform boot).

Usage of one option or another is having no significant influence on the achieved waveform portability. Dynamic Resource Deployment is an advanced option that should be implemented to answer duly identified operational requirements.

4. ESSOR Radio Devices and Services

4.1 Concept of Radio Devices and Services

The concept of Radio Device and Radio Service in SDR architecture is derived from logical devices and services defined by the JTRS SCA for the radio domain. From a waveform portability perspective, the interest of RD and RS is to provide the waveforms with the capacity to access the specific features of a platform independently from the implementation of these features. The ESSOR Architecture defines a set of API for RD and RS on which waveforms can rely on, and that each ESSOR compliant platform implements specifically depending on its own characteristics.

Distinctions between RD and RS features are as follow:

- **Radio Device API:** access to features which are typically implemented by hardware,
- **Radio Service API:** access to features which are typically implemented by software.

The RD and RS have all a similar structure. They implement a main interface defined by the SCA 2.2.2 specification (CF::Resource, CF::Device, CF::LoadableDevice or CF::ExecutableDevice) depending on the type of the component. This interface is usually used by Core Framework or dedicated platform components for general control (start, stop, management of configuration, testing). The specific features are implemented by ports. A port is associated with an interface (defined as an IDL interface according to the SCA) and the set of interfaces associated with ports of a RD or RS defines the RD or RS API. In the ESSOR Architecture, the definition of port is similar to those used into the SCA 2.2.2 specification.

Although the concept of logical devices and services defined by the SCA provides a good level of support to waveform portability, ESSOR Architecture identifies areas of portability enhancements:

- **Performance criteria:** the purpose of performance criteria is to define a performance capacity that each implementation of the ESSOR Architecture, which will implement the radio device or radio service on which the criteria apply, will have to document.

  Criteria may have different types. It can be the capacity of some radio devices to support a specific configuration (as list of supported baud rates for the SerialPortDevice), it can be the capacity of the hardware below the implementation of the architecture (delay for up and down conversions of base-band samples for example) or the delay necessary to perform a software processing (delay necessary to route an IP packet for example). All these performances are correlated, from waveform point of view, to a radio device and radio service or API.

  The ESSOR Architecture does not provide values for performance criteria, but the supported waveforms provide the list of required performance criteria, and the associated values necessary to support them. Based on that information it is possible to compare the waveforms required performances with the performances provided by an implementation of the ESSOR Architecture, and to check if these performances enable the identified waveforms to be ported on this implementation.
- **Guidelines to design Real Time interfaces:** the purpose is to provide guidelines relative to the definition of IDL interfaces for radio devices and radio services so that they limit time overheads for the components that implement them. These guidelines are decomposed in a set of rules which address different aspects of the definition of an IDL interface.
- **Definition of allocation properties for WF deployment and RD/RS usage:** these allocation properties are used to make association (dependency) between waveform components (Resources)

and platform components (Devices and Services). Each platform component with the aid of allocation properties defines its own allocation capabilities. On the other hand each waveform resource, using reference to allocation properties defined for each Device or Service, exposes its own needs which have to be fulfilled by platform components. In order to increase portability of waveform components, the ESSOR Architecture defines two kinds of association between waveform and platform components:

- **Deployment related:** in order to deploy each of the waveform components, ApplicationFactory has to find suitable Loadable/Executable Device (based on provided allocation properties) which fulfils all resource requirements. Those requirements are defined in SCA Software Package Descriptor (SPD) file (section implementation) for each of the waveform components.

- **Usage related:** in order to use some Radio Devices and/or Radio Services by waveform component, ApplicationFactory has to find suitable platform components (based on provided allocation properties) which fulfils all resource requirements. Those requirements could be defined in SPD file (sections usesdevice) for each of the waveform components.

4.2 ESSOR Radio Devices

Radio Devices are software components that provide an abstraction of the HW modules of an SDR. Radio Devices offer a high-level SW interface (API) to the other entities of an SDR (e.g. Waveform or Radio Services) that need to access Hardware modules.

RD APIs are considered as part of the most important interfaces in SDR architecture, providing the possibility to maximise the waveform portability. In addition, ESSOR Radio Devices definition tries to maximise backward compatibility with JTRS APIs, extends the JTRS APIs and creates new APIs in order to support fourteen different military waveforms, covering HF / VHF / UHF Frequency bands, addressing National and Coalition needs and operating in Narrow Band and Wide Band channels.

This has been done by taking the published JTRS APIs as a starting point, then identifying the shortfalls and missing functionalities of these current APIs in front of the agreed set of legacy waveforms and the new ESSOR HDR waveform.

Table 1 below depicts the Radio Devices considered in ESSOR definition.

ESSOR RD API	**Reference JTRS API**	**Considered WInnF API**
Audio Port Device	AudioPortDevice API [5]	
Platform Discrete Device	-	
Serial Port Device	SerialPortDevice API [6]	
Ethernet Device	EthernetDevice API [7]	
GNSS Device	GpsDevice API [8]	
Transceiver Subsystem Device		Transceiver Facility Specification

Table 1 - List of Radio Devices (RD) of the ESSOR Architecture

From the previous table:

- The **Audio Port Device API** provides access to the Audio Port HW. This API is used in close co-operation with **Vocoder Service API**,
- The **Platform Discrete API** provides interface for SDR components that need software access to a predefined set of hardware platform discrete signals,
- The **Serial Port Device API** provides access to the Serial Port Hardware,
- The **Ethernet Device API** provides access to the Ethernet Hardware,
- The **GNSS Device API** provides access to GNSS Receiver Hardware,
- The **Transceiver Subsystem Device API** (described in § 4.3).

For a Platform implementation, the adoption of a Radio Device or a Radio Service API is considered a Platform-dependent choice since, for the scalability concept, it depends on the applicability of such API to the PTF; for this reason no API can be considered strictly mandatory in a PTF.

The definition of APIs uses the "Extension approach" (derived from the JTRS specification), based on the fact that an API is composed of some basic interfaces (and ports), providing the main functionalities, and extensions (where needed) consisting of supplementary interfaces (and ports), that provide additional functionalities. By this way the API developer implements at least basic functionalities and then implements extension interfaces according to the needs.

When an API is adopted in a Platform implementation:

- All basic interfaces/ports shall be implemented, except some supported capabilities that are optional (not mandatory), e.g. some ports for internal connections between PTF components.
- Extensions are generally not mandatory, and can be implemented if requested and applicable to the specific PTF, even if in some cases the adoption of at least one extension is strictly necessary.

UML 2.0 modelling language has been adopted for the description of the components, with a usage of Class diagrams, State diagrams, Sequence diagrams, Context diagrams and Ports diagrams.

Any of these APIs are defined with state of the art approach and exhaustive specification information, like context, class, state and sequence diagrams plus behaviour description and, obviously, the IDL definition for each specific component's interface.

4.3 ESSOR Transceiver Subsystem API

The ESSOR Transceiver Subsystem Device API addresses the complex topic of definition of the adequate API relative to the subsystem of the radio set in charge of acquisition (in reception) and radiation (in transmission) of the radio signal, denoted Transceiver as contraction of the terms "transmitter" and "receiver", as represented in Figure 9 below.

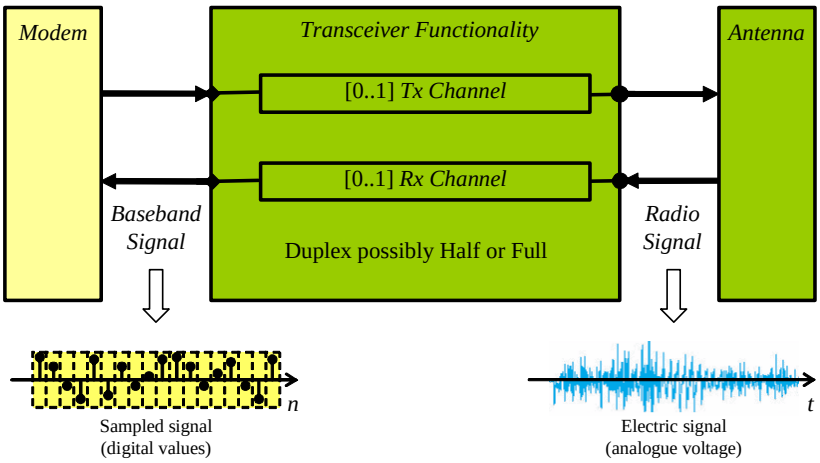

Figure 9 - Role of Transceiver

This API of the ESSOR Architecture has been defined considering the technical specification Transceiver Facility V1 [9] of the Wireless Innovation Forum (WInnF – former SDR Forum [10]). A certain number of improvement modifications, and, furthermore, an important number of additions have been brought by ESSOR in defining the ESSOR Transceiver API. The Transceiver Subsystem Device API is defined in accordance with the principles that drive the definition of the other RD API of ESSOR. As such, compliance with SCA Device interface, definition of IDL and associated typical set of ports is achieved.

4.3.1 Key functional Features

One first particular aspect is related to the structure of the Transceiver Subsystem API, that is uniquely distributed among a reduced set of six mandatory interfaces, which provide the fundamental control and data exchange, and 3 sets of extensions, which provide optional interfaces (one set for Rx, one set for Tx, and the third one, *Get Boundaries*, for Transceiver reconfiguration management). Such degree of flexibility is justified by the fact that Transceiver Subsystem API needs to embrace the needs of the high variety in waveform applications potentially implemented by a given Transceiver Subsystem.

The real-time control mechanism implemented by mandatory interfaces TxControl and RxControl is relying upon the notion of burst, which represents the elementary operational period of a Transceiver. Effective usage of a Transceiver is therefore consisting in activating a succession of discontinuous bursts.

The control of bursts has a very high flexibility thanks to the key notions of Timing Profile ("when and for how long to operate a burst") and Tuning Profile ("how to transform the signal when operating a burst") as described in Figure 10 below.

- The Start Time of the Timing Profile is expressed using an Absolute or an Event-based (= relative) referencing convention.

- The Tuning Profile is expressed using explicit arguments values (e.g. for Carrier Frequency, Transmit Power, Receive Baseband Packet Size) plus one particular integer index, that references, among a waveform-specific predefined set (= Tuning Presets), the behaviour that the Transceiver shall apply in the signal transform.

Last, data is exchanged amongst Modem and Transceiver by packets of baseband samples a given burst

being possible composed of a single packet or several ones.

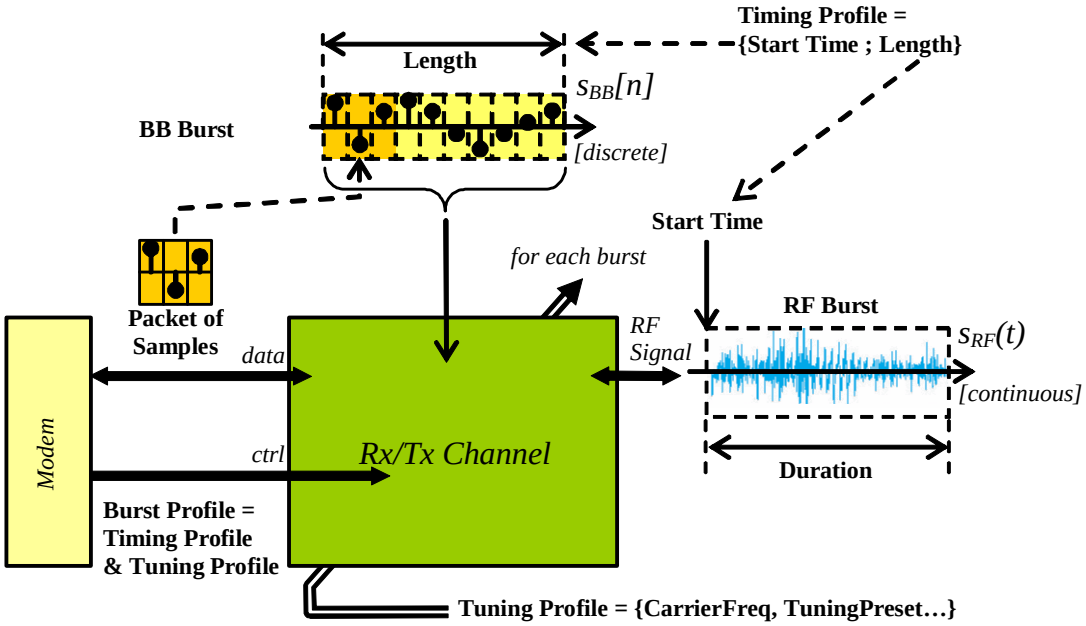

Figure 10 - Burst creation and data exchanges

Interfaces in Rx and Tx extension enable to deal with a lot of side capabilities such as Adaptive Gain Control (AGC), in-burst tuning, asynchronous Tx flow control.

4.3.2 Reconfiguration

In addition to the mechanisms defined for effective waveform usage, the ESSOR Transceiver Subsystem API clarifies the way to reconfigure the Transceiver, as highlighted in Figure 11 below:

- Definition of one simple interface that enables the platform control to select the desired waveform, using an integer index (the way a Transceiver reconfigures itself to comply with the needs of the requested waveform is left to implementation),

- Definition of a set of common formal parameters enabling to characterise the expected behaviours, called Transceiver Configuration Profiles (definition of the required Tuning Presets are one important part of those Configuration Profiles).

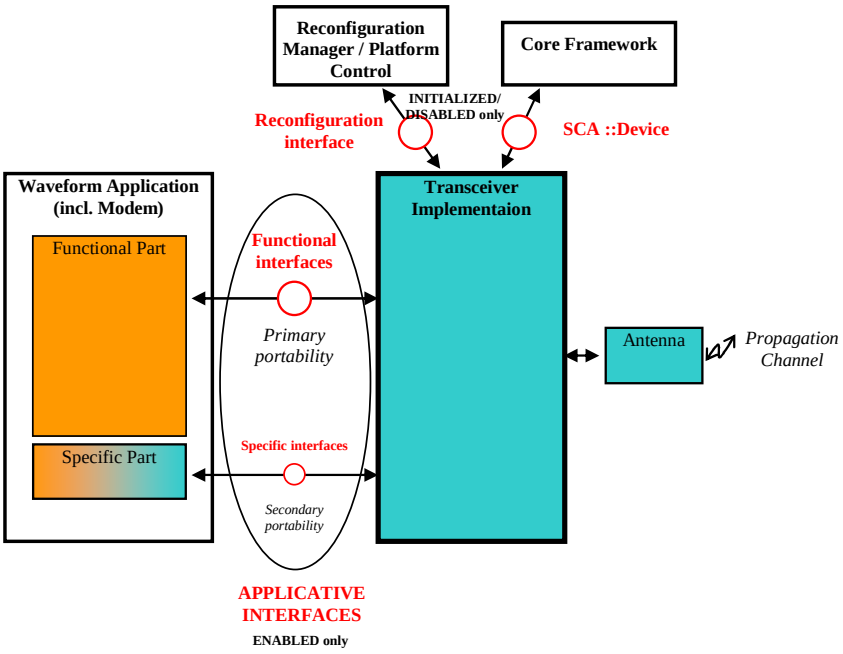

Figure 11 - Overview of Transceiver Interfaces

4.3.3 Additional characteristics of the ESSOR Transceiver API

In addition to the functional aspects presented in previous chapter, the fact that the Transceiver Subsystem API has to support implementation of Modem operations creates a need to complete the regular SCA-based specification approach for Radio Devices with additional aspects:

- Possibility to make *direct* connections between Modem and Transceiver, without the usage of a CORBA middleware or ESSOR MHAL, is granted by definition of language-specific interfaces suited for C, C++ and VHDL implementations.

- A complete part of the Transceiver Subsystem API is dedicated to formalization of real-time constraints to be used in system and software engineering phases, so that the joint operations of the waveform application and the transceiver subsystem meet the generally stringent real-time requirements applicable to modem.

- The Transceiver Subsystem is the typical example of RD that requires implementation in the form a Distributed Device, where API-compliant software interfaces to the Transceiver are located in different processing units, as highlighted in Figure 12 below.

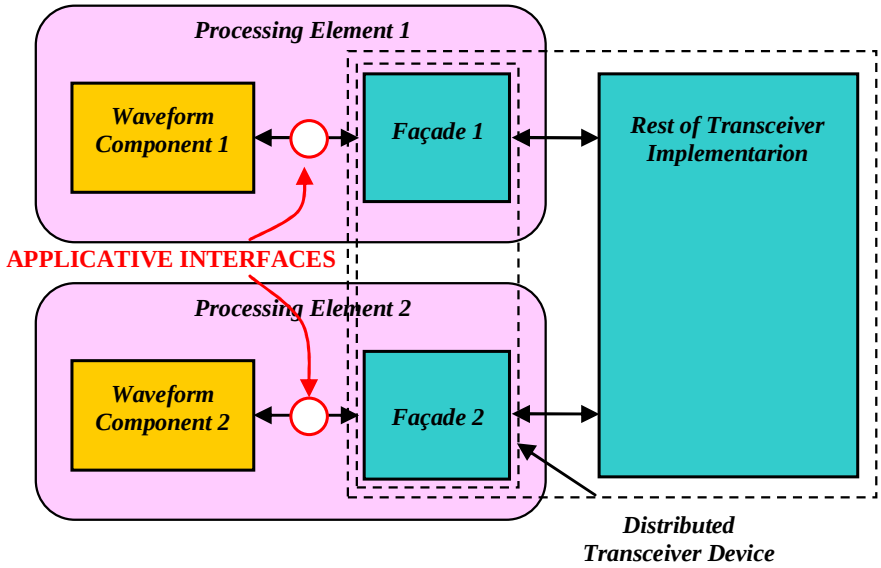

Figure 12 - Distributed Transceiver Device and Façades

- Last, from an implementation perspective, a *Transceiver Device* can be entirely based on proprietary design, or can use building blocks compliant with implementation standards. A typical example of these implementation standards is JTRS MHAL RF Chain Coordinator, as introduced in § A.7 of [3], which enables programming and real-time access to the Transceiver. Another possible example is usage of OBSAI (Open Base Station Architecture Initiative – www.obsai.org) or CPRI (Common Public Radio Interface – www.cpri.info) interfaces defined for cellular systems base stations.

4.4 ESSOR Radio Services

Radio Services are software components that provide functionalities useful for the waveform. These functionalities are related to the operations of the waveform (Timing Service, IP Service, Retransmission Service), its control and monitoring (HMI Service, SNMP Service, Fault Management), the download of its files and the properties configuration (Configuration Service).

As presented for the RD APIs in § 4.2, the adaptation of RS APIs is a Platform dependent choice.

The approach of the ESSOR Radio Services API specification is also to maximise the compatibility with the JTRS specification, where applicable. The API definition starts from the existing published JTRS API, highlighting the parts where clarifications or modifications are applied, and extending those interfaces and behaviours to cope with ESSOR additional functionalities.

When needed, additional RS API are defined. However, due to a limited number of JTRS published RS APIs, JTRS is influencing ESSOR Architecture lower in RS (only for Timing Service [11] and Vocoder Service [12]) than for RD API specifications.

The **Vocoder Service API** supports an extended set of vocoding algorithms compared to JTRS definition.

Table 2 below summarizes the list of Radio Services that are considered in ESSOR Service API definition.

Radio Services	Functional Groups / Functionalities
Configuration Service	**SYSTEM CONFIGURATION** • WF deploy Persistent information Management • WF post-deploy Persistent Information Management **WF INSTALL** • WF Files storage • Installed WF record update **WF UNINSTALL** • Installed WF record removal
Fault Management Service	**FAULT MONITOR MANAGEMENT** • FAULT monitor • Fault events Management
HMI Service	**USER CONTROL INTERFACE** • WF HMI control
IP Service	**NETWORKING** • IP Traffic Management • IP packet Route
Retransmission Service	**RETRANSMISSION MANAGEMENT** • Retransmission channel Lifecycle • Retransmission channel Management
SNMP Service	**USER CONTROL INTERFACE** • WF SNMP control
Timing Service	**TIME MANAGEMENT** • System Time Management • 1PPS Management • External Synchronization
Vocoder Service	**USER INFORMATION INTERFACE** (AUDIO) • VOCODING algorithms • VOCODING streams • VOCODING Management

Table 2 - List of Radio Services (RS) of the ESSOR Architecture

Each of these APIs is defined with state of the art approach and exhaustive specification information, like context, class, state and sequence diagrams plus behaviour description and, obviously, the IDL definition for each specific component's interface.

5. Conclusion

The ESSOR Industries presented in this article an overview of the ESSOR Architecture, built from the main specifications published by JTRS for SDR standardisation: SCA 2.2.2 specification and JTRS API specification.

One main area of the extensions provided by the ESSOR Architecture relatively to SCA 2.2.2 concerns the definition of complete Operating Environments for DSP and FPGA Processing Elements, with two Connectivity options specified: CORBA and ESSOR MHAL. One other important area of extension concerned the definition of additional Radio Devices and Services API, including specification of a complete Transceiver API based from the Wireless Innovation Forum Transceiver Facility. The aspects related to Radio Security Services and security architecture were not covered in the document.

Implementation of the ESSOR Architecture will be realized on the 6 different platforms of the ESSOR participating states, and the ESSOR HDR WF will then be ported on those six platforms. This effort will bring decisive return of experience on the ESSOR Architecture.

6. Abbreviations

Abbreviation	Definition
AEP	Application Environment Profile
API	Application Program Interface
BB	Base Band
BSI	Base Software Item
CA	Common Activities
CF	Core Framework
CNC	Critical Non Common activities
CORBA	Common Object Request Broker Architecture
CPRI	Common Public Radio Interface
CSP	Communications Sequential Process
DSP	Digital Signal Processor
EDA	European Defence Agency
FPGA	Field Programmable Gate Array
GNSS	Global Navigation Satellite System
GPP	General Purpose Processor
HDR	High Data Rate
HMI	Human-Machine Interface
HW	Hardware
ID	Identification, Identifier
IDL	Interface Definition Language
IP	Internet Protocol
JTRS	Joint Tactical Radio System
LD	Logical Destination
MHAL	Modem Hardware Abstraction Layer
OBSAI	Open Base Station Architecture Initiative

OCCAR	Organisation Conjointe de Coopération en matière d'ARmement
OE	Operating Environment
ORB	Object Request Broker
OS	Operating System
POSIX®	Portable Operating System Interface
PTF	Platform
PPS	Pulse Per Second
RD	Radio Devices
RF	Radio Frequency
RS	Radio Services
RTL	Register Transfer Language
SCA	Software Communications Architecture
SDR	Software Defined Radio
SNMP	Simple Network Management Protocol
SPD	Software Package Descriptor
SW	Software
UML	Unified Modelling Language
VHDL	VHSIC Hardware Description Language
WF	Waveform
WInnF	Wireless Innovation Forum

7. References

[1] Joint Tactical Radio System (JTRS) - http://sca.jpeojtrs.mil/

[2] Software Communications Architecture Specification - V 2.2.2 May 15, 2006 - http://sca.jpeojtrs.mil/

[3] Joint Tactical Radio System (JTRS) Standard - Modem Hardware Abstraction Layer Application Program Interface (API - V_2.11.1 - 02 May 2007)

[4] Communicating Sequential Processes - Hoare, C. A. R. (1978)

[5] Joint Tactical Radio System (JTRS) Standard - Audio Port Device Application Program Interface (API) - V_1.3.2 - 02 April 2008

[6] Joint Tactical Radio System (JTRS) Standard - Serial Port Device Application Program Interface (API) - V_2.1.2 - 29 May 2009

[7] Joint Tactical Radio System (JTRS) Standard - Ethernet Device Application Program Interface (API) - V_1.2.2 - 31 March 2008

[8] Joint Tactical Radio System (JTRS) Standard - Global Positioning System - Application Program Interface (API) - V_2.1.2 - 30 May 2008

[9] Transceiver Facility V1 Wireless Innovation Forum- January 2009 http://groups.winnforum.org/d/do/1554

[10] Wireless Innovation Forum - http://www.wirelessinnovation.org

[11] Joint Tactical Radio System (JTRS) Standard Timing Service Application Program Interface (API) - V_1.4.2 - 30 May 2008

[12] Joint Tactical Radio System (JTRS) Standard Vocoder Service Application Program Interface (API) - V_1.1.1.1 - 02 August 2007

8. Appendix - Summary of the ESSOR Program

On behalf of Finnish, French, Italian, Polish, Spanish and Swedish Governments, the Organisation Conjointe de Coopération en matière d'ARmement – Executive Administration (OCCAR-EA) has signed on 19 December 2008 the European Secure Software defined Radio (ESSOR) contract with a joint venture composed of the industrial National Champions from Finland (Elektrobit), France (THALES Communications S.A.), Italy (Selex Communications S.p.A.), Poland (Radmor S.A.), Spain (Indra Sistemas) and Sweden (Saab AB). On governmental side, the ESSOR Program is managed by an OCCAR-EA Program Division based in Bonn (Germany). On Industrial side, the ESSOR Program is managed by the a4ESSOR S.A.S, a dedicated joint venture specifically set up by the 6 National Champions and based in Colombes (France).

The ESSOR Program addresses the following two Areas of Interest:

1. **A new High Data Rate Waveform (HDR WF):** The goals are to define, simulate, develop an interoperable secure high data rate mobile ad hoc networking waveform for Land military applications, to port, validate and demonstrate this waveform on different national SDR platforms.

2. **The ESSOR Architecture:** The goals are to define a secure SDR architecture for military purposes; to develop, implement and validate this architecture on different national SDR platforms.

The ESSOR Program activities are divided in two types:

1. **Common Activities (CA):** Activities collectively executed by the 6 National Champions.

2. **Critical Non-Common Activities (CNC):** National activities complementary to the Common Activities, individually executed by each National Champion.

Figure 13 below identifies the main milestones planned for the ESSOR Program (4.5 years long):

- **Jan 2009** – Kick-off of the Program,
- **Mid 2010** – Definition of the ESSOR Architecture (CA),
- **End 2010** – Definition of the HDR WF Specification supported by High Fidelity Simulations. (CA),
- **Mid 2011** – Upgrade to the ESSOR Architecture on the 6 different National Champion SDR Platforms, with related National validation (CNC),
- **End 2011** – Development of the HDR Base WF and validation into a Native Test Environment (CA),
- **End 2012** – Porting of the HDR Base WF on the 6 different National Champion SDR Platforms upgraded to ESSOR Architecture, with related National validation (CNC),
- **Mid 2013** – Multinational HDR WF interoperability demonstration amongst the 6 different National Champion SDR Platforms. (CA).

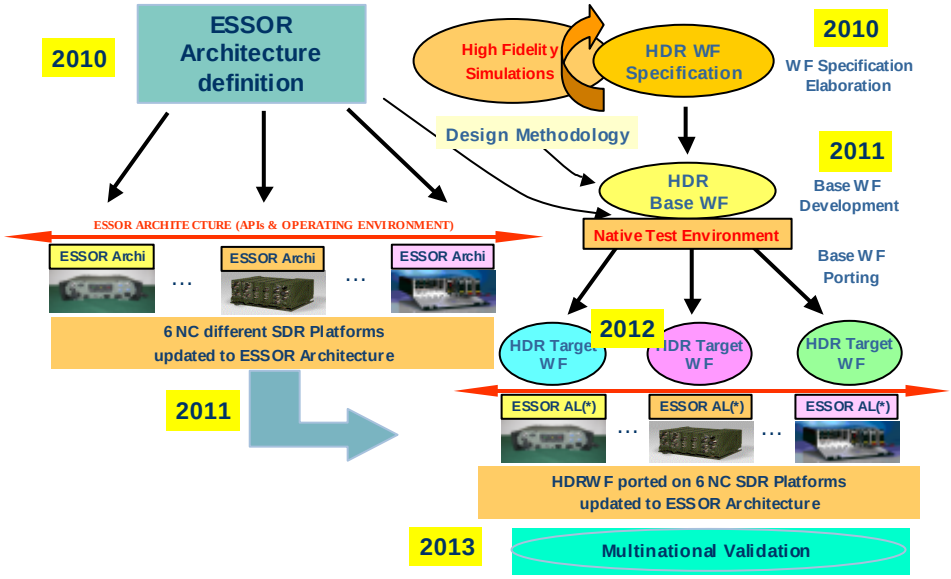

Figure 13 - Scope and general time frame of the ESSOR Program

END OF THE DOCUMENT