SOFTWARE COMMUNICATIONS ARCHITECTURE TRADEOFF CONSIDERATIONS FOR MULTIPLE DEPLOYMENT PLATFORMS

Vincent J. Kovarik Jr.
(Harris Corporation, Melbourne, FL, 32902, United States, vkovarik@harris.com)

ABSTRACT

A specification for a common software architecture for software defined radios has been Proposed by the Joint Tactical Radio System (JTRS) Software Communication Architecture (SCA). While a core set of control and configuration interfaces provides a good starting point for interoperability, the SCA does not prescribe any specific approaches regarding the implementation architecture. This has resulted in issues regarding the application of the SCA to different target platforms. In response to some of these platform issues, exploratory initiatives, such as the Core Framework (CF) Lite, have begun to focus on framework implementations that remain SCA compliant but can be hosted on smaller, resource limited platforms. While the CF-Lite is focused on the specific problems introduced by the small platform, it does not address the range of deployment platform permutations that will eventually host the SCA. This paper explores implementation architectures that may be applied to a range of radio systems.

1. INTRODUCTION

Few technologies have been developed that have had as significant an impact on an industry as the Software Defined Radio [1,2] (SDR). The Software Communications Architecture (SCA) has accelerated this impact on radio systems for the U.S. government by rapidly becoming the standard architecture required in military radio system procurements.

Developed under the Joint Tactical Radio System (JTRS) program requirements [3], the SCA defines a common software infrastructure that prescribes how a radio system shall be implemented. The objective of the SCA is to reduce the long-term cost of military communications systems by enabling radios to host new capabilities through software upgrades rather than requiring expensive hardware upgrades or, worse, entirely new hardware.

While the intent of the SCA is to provide a common framework that insulates the waveform applications from the underlying hardware implementation. There are, nevertheless, critical interdependencies between the implementation of the waveform application and the underlying radio hardware. Figure 1 illustrates the conceptual implementation space of a software-defined radio.

At the highest level is the definition of the waveform application. This definition may be specified as a MATLAB simulation, a set of mathematical specifications, or some other functional specification method. This specification.

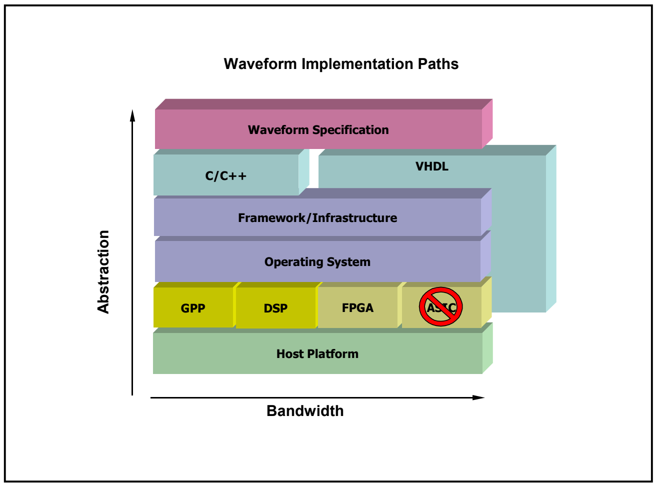

Figure 1: Software Radio Abstraction Layers

represents the highest level of abstraction of the waveform

Depending upon the bandwidth and throughput requirements of the waveform application, the implementation approach typically follows one of two paths. The waveform may be implemented in a high-level programming language such as C or C++, if targeted for deployment on a General Purpose Processor (GPP) or Digital Signal Processor (DSP). If the waveform performance is sufficiently demanding then it will push the specification towards VHDL. This path leads to an implementation using a Field Programmable Gate Array (FPGA) or an Application Specific Integrated Circuit (ASIC). For the most part, ASIC implementations are not considered as part of the solution space for software-defined radios because of their immutability after deployment.

Between the hardware and the waveform application sits the operating system for the radio set and the radio infrastructure, known in the SCA as the Core Framework. The Core Framework (CF) provides the common view and interfaces into the radio set resources.

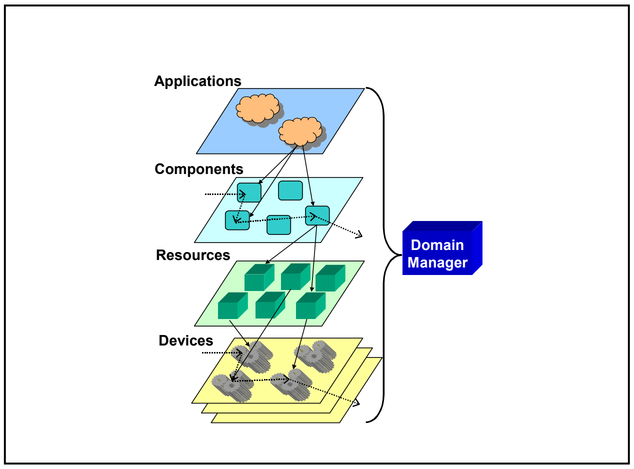

Figure 2: SCA Abstraction Space

How the SCA decomposes the waveform implementation and maps it onto a set of hardware devices is illustrated in figure 2. At the highest level, the application is the functional entity that the radio system executes. The waveform application is described as a collection of components that provide discrete functions and are logically tied together through the use of the SCA Port abstraction. The next lower level defines a collection of resources that describe the capabilities of the underlying hardware devices. Finally, at the lowest level, the hardware devices form the physical platform providing the processing engine required to execute the waveform implementation.

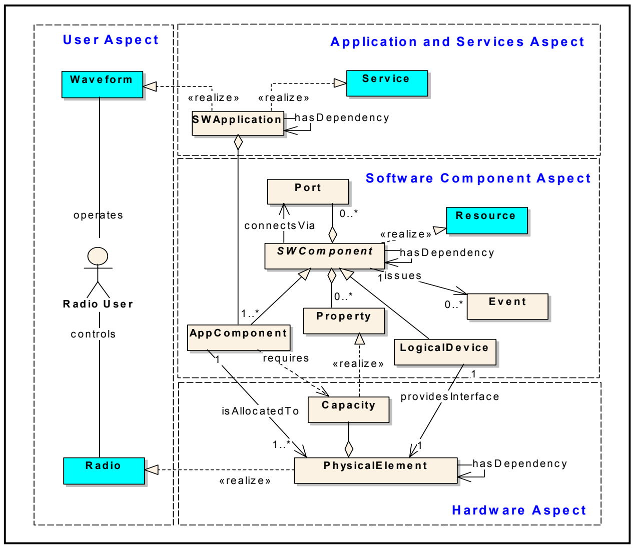

Figure 3: Aspects of an SCA Radio Set

An alternative view of the SCA abstraction space is to look at the aspects of the software radio. Aspect-oriented analysis or programming has evolved over the past several years as a mechanism for providing a larger encapsulation of the system facets. These aspects are illustrated in Figure 3.

At the lowest level, as with the SCA abstraction view of Figure 2, is the hardware aspect. This level defines the set of hardware components, their capabilities, and interconnections. The next level up is the Software Component aspect. At this level the physical hardware is abstracted into a Logical Device. This Logical Device implements the SCA Device interface thereby providing a consistent function protocol into the hardware platform. The Application aspect provides a consistent encapsulation of the waveform application and other software that provides a service running on the system. Finally, the User aspect reflects the user's view of the software radio. From the user's perspective, the interactions are either controlling the physical radio elements or, through the radio interface, controlling the waveform as a logical entity.

2. THE SCA ARCHITECTURE

The SCA architecture is a derivation of the CORBA Components Model (CCM). The CCM defines a framework for the specification and assembly of software components in a dynamic fashion. The key concept is the ability to connect different, pre-built components in a dynamic fashion rather than building a monolithic and inflexible system.

2.1 SCA Components

The SCA Core Framework components consist of a Domain Manager (which typically includes an implementation of the Application Factory and Application as well as an instance of a File Manager), one or more Device Manager instances (which includes a File System as part of the Device Manager), one or more Device instances, and one or more waveforms application instances. Note that these instances provide the underlying waveform instantiation and control logic in the form of an Assembly Controller to which the Application provided in the Domain Manager delegates all operations.

Tying these components together is a set of services including a CORBA Name Service, a Log Service (Note that there now exists an Object Management Group (OMG) approved Lightweight Log Service that the SCA will likely reference in a future release of the specification and drop the SCA-defined Log Service), and a CORBA event Service.

Thus, in a fully SCA-compliant radio system, the above components and services must be provided. In the current SCA operational scenario, there is a "boot node" which provides the initial system boot sequence. This node powers up, starts the Device Manager for that node and then initiates additional processes including a File System, a Log Service, Event Service, and the Domain Manager. In addition, any devices that are managed by the Device Manager for the node are started and initialized.

After the primary node is booted and the components are started and initialized, additional nodes may be brought up as well. Each of these nodes may include a Device Manager and multiples Devices and additional software components.

The number of process can vary significantly, based on the specific implementation approach. Table 1 illustrates a typical set, based on the SCA specification.

Table 1: SCA Components and Processes

Component	Process(es)
Device Manager	1 per Device Manager instance
Device	1 per device. May be implemented as an instance within the Device Manager and accessed via a thread.
Log Service	1 per system. Multiple log services may co-exist on a radio set.
File System	1 per Device Manager instance. May be instantiated within a Device Manager and accessed via a thread.
Domain Manager	1 per radio system
Event Service	1 per radio system
Name Service	1 per radio system
Application	1 per instance. Depending on how components of the waveform application are implemented, this could result in many processes.

As can be seen from Table 1, depending on the implementation approach and number of devices, the number of processes required to boot up an SCA-compliant system can grow to more than a dozen. Depending on the capabilities of the general-purpose processor, the resources consumed for basic framework operations can be significant.

3. THE DEPLOYMENT SPECTRUM

There is a wide range of deployment platforms and options based on the number and type of waveforms that the radio system is required to support. This need must be tempered with the physical constraints introduced by the radio's targeted deployment environment.

3.1 SCA Deployment Drivers

There are several key deployment drivers that impinge on the implementation strategy for an SCA compliant software radio. The first is the overall bandwidth or throughput requirements. Depending on the data rate and the complexity of the waveform, conventional programming language implementations and the processor type may not be sufficiently powerful to support the waveform. While there are certain issues that must be addressed to implement an SCA-compliant FPGA processing device, it is realizable. Some approaches for SCA-compliant FPGA-based implementations are presented in [4].

The throughput and bandwidth of the waveforms to be hosted on the software radio must be balanced with the physical constraints imposed by the target user and environment. The system must live within the size, weight and power constraint imposed by the operational environment, usage scenarios and form factor requirements.

3.2 Deployment Environments

As noted in the introduction, current implementations typically fall into resource rich or resource-limited systems. At the upper-end of the deployment range are systems that have, for all practical consideration, virtually unlimited power and are not constrained by form factor issues. Such deployed systems typically are realized in the form of a chassis such as VME or compact PCI. These systems can have multiple single board computers, special purpose signal processing components, and other radio system components that are realized as a PC board within the chassis or rack.

The SCA implementation for systems such as these are not constrained in terms of how they must manage the system resources beyond the requirements levied by the SCA in terms of allocating resources to components of a radio application. JTRS cluster 1, 3, and 4 in the JTRS procurement process may be generally placed in this category, although Clusters 1 and 4 do have some unique form factor constraints that will limit available resources and drive several implementation decisions.

At the lower-end of the deployment spectrum is the man-pack and hand held radio systems. These systems are battery powered and carried by personnel. Any reduction in weight for these systems yields a direct benefit to the radio user in terms of lessening the total amount of required weight. Of course, the battery is the primary culprit that adds weight to the overall unit. Thus, the battery is the typically driving factor in the engineering of the radio system. Approaches for SCA implementation on these platforms have been proposed by [5] and [6].

In a hand held system, the capabilities of the radio set are driven by the power resulting in very different design tradeoffs. Issues such as partial operation, being able to turn off components when not in use, and still retain content for fast system start-up, need to be addressed. Clusters 2 and 5 of the JTRS procurement process generally fall into this area.

3.3 Implementation Constraints

On the surface it would appear that a simple decomposition of the SCA implementation into two categories would be adequate and, indeed, that is primarily the approach to date. However, upon closer inspection of the radio sets and the finer-grained variations between them, it becomes apparent that what is present is not a binary world where each radio set can be simply categorized as large or small. Instead, the implementation architecture is a rich set of constraints that are interrelated.

To understand how deployment constraints are organized in a more fine-grained ontology, they must be mapped to the underlying SCA architecture. The following section discusses specific areas of functional SCA requirements for which the implementation architecture choices have a direct bearing on the radio system architecture.

4. SCA COMPLIANCE

The baseline SCA specification identifies a number of functional requirements, system components, and radio services that must be present in order to be considered compliant with the specification and to receive certification. In analyzing the specification, several key components come to the fore. These components are discussed below.

4.1 Name Service

The implementation of a Name Service provides a global mechanism for locating services within a software system. In the context of a distributed system, the Name Service provides a valuable capability in that it eliminates the necessity to encode the location and/or path to an application. Services need only to register with the Name Service.

Typically, the name service runs as a process in a networked system. In a resource-rich system with multiple processors, providing a name service as a process on one of the GPP nodes doesn't present an issue. In a resource-limited environment, however, the use of resources to start and manage a process that, essentially, simply provides a lookup service for radio components to locate and connect to each other is impractical.

4.2 XML Processing

Waveform applications are described in the SCA using eXtensible Markup Language (XML) files. These files describe the set of application components, the resources required for the components, and the connectivity. The set of XML files describe the Domain Profile for a waveform. While XML parsing is not a CPU intensive process, is does entail a significant amount of string processing and construction of an internal tree structure representing the XML elements, their attribute, and values. This XML tree is then traversed and a more efficient internal data structure is constructed that represents the Domain Profile.

For example, the implementation of the Domain Profile in dmTK consists of more than forty C++ classes. The classes are instantiated based on the content of the XML specification of the XML files. Once instantiated, it is more efficient to traverse the semantic relationships of the Domain Profile following data structure pointers rather than traversing the XML tree structure because the semantic interpretation of the dependencies and relationships that cannot be expressed in the XML files are built into the instantiated object model.

The processing of the XML files, however, is not limited to the definition of waveform applications. XML files also define the properties of the SCA Devices and Device Managers. So processing XML files is a common aspect of an SCA implementation.

In a resource-limited environment, however, the processing of textual data as the primary mechanism for defining the waveform application components and organization is not a resource efficient approach. In some cases the XML processing is performed on startup of the hardware, e.g. Devices and Device Managers, and others it is performed upon installation of a new waveform application.

4.3 Waveform Data Transport

Perhaps the most significant performance aspect of a software-defined radio and an SCA-compliant radio system specifically is the mechanism used to move waveform data along the processing chain. In an abstract sense, processing a waveform can be viewed as a data flow machine.

As the data is processed in one functional component, it is moved along to the next. This abstract model holds for any type of waveform data from baseband samples to InPhase and Quadrature (I&Q) data.

As illustrated in Figure 4, the Port merely connects to endpoints of a communications link. There is no implication within the connection specification regarding the communications transport protocol.

If one takes the SCA requirements literally, CORBA is the data transport mechanism that must be used in an SCA radio. However, without too much analysis effort, it is easy to ascertain that such an absolute interpretation is not only impractical but unrealistic. Data transport for high-bandwidth waveform applications require alternative transports if the performance requirements are to be met.

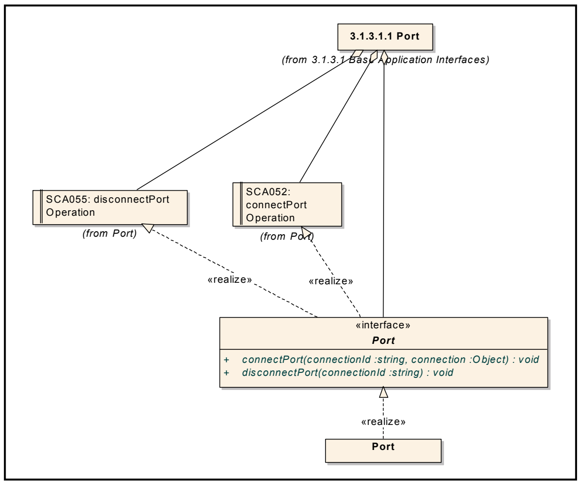
3.1.3.1.1 Port
(from 3.1.3.1 Basic Application Interfaces)
SCA055: disconnectPort Operation
(from Port)
SCA052: connectPort Operation
(from Port)
«realize»
«realize»
«interface»
Port
+ connectPort(connectionId :string, connection :Object) : void
+ disconnectPort(connectionId :string) : void
«realize»
Port

Figure 4: Port Interface and Requirements

In the following section alternative interpretations and approaches to the above components of the SCA specification are presented.

4. IMPLEMENTATION ADAPTATIONS

This section presents some different perspectives to the issues identified above and proposes several implementation alternatives. The objective is to maintain a common and compliant architecture at the highest level possible while providing realistic alternatives for different deployment configurations. These implementation options are intended to maintain compatibility to the greatest extent possible which still addressing the platform specific constraints and limitations of the radio system.

4.1 Name Service

Fundamentally, the Name Service can be viewed of as a registry of services. Each service and application makes its capabilities available to other components by registering with the Name Service. The Name Service then accepts requests from clients wishing to use a particular service, looks up the service using the name provided, and provides a connection endpoint through which the client can connect and issues requests to the service provided.

In a multi-process, distributed environment, the Name Service typically runs as a stand-alone process within the distributed system. The Name Service runs in a "well known" location or, as in some ORBs, is located through an IP-multicast call. Once located, the applications can register and request services.

The service name is simply a string name provided within a hierarchical naming context. So, for example, a GPS time service might register as `/dmTK/GPSTime`, where `/dmTK` represents the top level context of the name service and `/GPSTime` is the string name for the service.

In a resource-limited environment, it may not be desirable, or even possible to run the Name Service as a separate process. However, the same functionally could be provided if the Name Service ran as a thread within a master process or, in a more minimalist implementation, it can be implemented as a service call within the operating system.

The key aspect is that, in all implementation scenarios, the function signature used to make the request to register or locate a service is equivalent. The only difference is the process model, e.g. a full process, a thread within a master process, or a service call within the kernel process space.

Thus, looking at the kernel function call in a bit more detail, the Name Service can be implemented as an object module that extends the capabilities of the operating system. This is much the same as object modules are used to add device-specific network card interface.

4.2 XML Processing

The processing of XML has several alternative approaches, again dependent on the resources available. As noted in the previous section, the SCA specification states that the XML files form the Domain Profile and are processed during waveform installation. Some individuals may also interpret the specification to mean that the XML files are also processed during waveform instantiation, although most individuals experienced with the SCA subscribe to the former interpretation.

While not computationally intensive, from a processing standpoint, the processing of text XML files can be resource intensive from the perspective of significant I/O to open and read a multitude of XML files and the requisite Document Type Descriptor (DTD) files that define the legal syntax of the XML files.

A common approach is to process the XML files during waveform installation and construct an internal representation of the domain profile that is more efficient to manage and process during waveform instantiation.

There are two fundamental approaches to improving the efficiency of XML file processing. One is to perform the initial processing, using the validating XML parser, build an internal representation, and then to utilize the internal representation within the fielded radio set. This frees the fielded radio from having to host an XML parser and maintain the XML source files internally on a small form factor or battery powered unit.

In this scenario, the SCA compliant radio set consists of the small, hand held unit and a laptop or other support processor networked together. The laptop hosts the XML processor and other support components required for set configuration and waveform installation but is not part of the deployed radio set. The deployed set maintains the internal domain profile representation resulting from the parsing process.

One of the common retorts to such an approach is that it does not meet the intent of the SCA radio architecture. While arguments can be offered for either interpretation, the underlying driver must be the full deployment environment and operational scenario. This approach allows a small hand-held to be fully re-configurable according to the SCA specification but limited to installed waveforms once configured for a specific mission.

Another, intermediate approach, that still provides the dynamic installation of the waveforms in the small form factor radio set without the overhead of a full XML validating parser, is to use a byte-code representation.

Similar to the approach used by programming languages such as Java, and others to provide platform transportability and independence, a byte-code specification of the XML element tags and attributes can be defined. Similar to the representation of an application in byte-code, the XML can be validated and parsed using a standard XML parser. Then the XML data tree can be post-processed to build a more compact byte-code assignment for the elements, attributes, and tags within the XML resulting in a more compact byte-code image.

The XML byte-code image is then resident on the deployed platform. An interpreter processes the byte-code image providing the essential information to the SCA Core Framework components thereby building the appropriate internal domain profile representation.

This approach yields a balance between the overhead of fielding a full XML parser and the stripped-down model of fielding only the domain profile built from the XML files. In this approach, the ability to install new waveforms is still provided but the textual XML files are pre-parsed, validated, and the byte-code image is fielded with the radio system.

4.3 Waveform Data Transport

Implementation of the waveform data transport has typically been approached using a CORBA-based implementation. There is a false perception that the underlying transport must utilize the traditional distributed CORBA implementation that resides on top of an IP stack when, in fact, the transport implementation is orthogonal to the interface specification.

The SCA currently uses the abstract concept of a Port to encapsulate the point-to-point connectivity. This approach provides a common mechanism for identifying data sources and sinks (The SCA names these *uses* and *provides* ports) and how those data sources and sinks are logically connected to form a data path.

Rather than force-fitting the communications between two connected ports to be a point-to-(multi)point CORBA connection, the transport mechanism could be abstracted away from the connection and the Port would simply provide the connection mechanism to an underlying data/message transport infrastructure. One could argue, in fact, that this abstraction is already provided by the Port abstraction in that the specification merely defines the minimal requirements to establish communications between two endpoints.

Following this approach would start to pave the way for a radio that is not merely a set of hardware loosely connected through a common software interface but be closer to the concept of a set of collaborative services that can be flexibly constructed in an ad hoc fashion realizing the specific functionality required at a given point in time.

5. CONCLUSIONS

This paper has identified several implementation tradeoffs in several key areas of the SCA. It has proposed several implementation variants that maintain the core intent of the SCA and compliance with the SCA specification while allowing a wider range of deployment implementations.

It is the author's belief that the SCA must continue to evolve such that the implementation architectures can be adapted depending on the deployment domains.

For this to occur, the definition of SCA compliance must evolve to encompass more than the current polarized interpretation for resource-rich and resource-limited environments. It must include the full spectrum of radio implementations.

6. REFERENCES

[1] Mitola III, J., *Software Radio Architecture: Object-Oriented Approaches to Wireless Systems Engineering*, 2000, John Wiley and Sons, Inc.

[2] Reed, J.H., Software Radio: A Modern Approach to Radio Engineering, 2002, Prentice Hall.

[3] *Joint Tactical Radio System (JTRS) Operation Requirements Document (ORD*), Version 32, April 2003, http://jtrs.army.mil/documents/JROC%20Approved%20ORD%20v3.2%209Apr03.pdf

[4] Kovarik, Jr., V.J., *Next Generation SDR: JTRS Myth, Reality, and Opportunity*, AIA Military Radio Conference, Washington, D.C., September 2003.

[5] Linn, C.A., *Designing Core Frameworks for Battery-Powered Platforms: 10 Techniques For Success*, Proceedings of the 2002 Software Defined Radio Technical Conference, November 2003.

[6] Cruz, J. W., Davis, T., Mario, M., Rolon, G., *An Approach to a Compact JTRS SCA Core Framework For Handheld Radios*, Proceedings of the 2002 Software Defined Radio Technical Conference, November 2003.