

# 1

## Introduction

The fundamental objective of the SCA is to provide a common software infrastructure for managing radio systems. Although software comprises a significant part of most recent radios – thus enabling new capabilities and functions to be added to the radio at some future times – the software is loaded and controlled through proprietary mechanisms and each radio manufacturer typically employs a unique infrastructure or architecture. A software defined radio, as interpreted here, refers to a class of radios, the capabilities of which are not simply provided by software but utilize an infrastructure that supports interchangeable components as well as functionality.

This chapter provide background information regarding the SCA. The SCA specification describes a collection of components, the configuration of the components, and the assembly of the components into a functional waveform application on a radio system. Taken together, these form an infrastructure for defining and constructing a software defined radio system.

### 1.1 Software Radios

Figure 1.1 illustrates the abstraction space of bandwidth versus waveform abstraction. At the lowest level is a set of hardware that provides the actual processing of the waveform and support software. The processing is provided by one of four options, General Purpose Processor (GPP), Digital Signal Processor (DSP), Field Programmable Gate Array (FPGA), and Application Specific Integrated Circuit (ASIC). The ASIC is typically not considered part of the solution set within a software radio because, once programmed, it cannot be modified after deployment – one of the fundamental tenets of a software radio.<sup>1</sup>

The aim of Figure 1.1 is to illustrate the two orthogonal perspectives of software radio design. The waveform design starts as a set of requirements, simulation, mathematical

---

<sup>1</sup> *This does not necessarily imply that an ASIC cannot be applied within a software defined radio. It is the authors' opinion that, given certain circumstances and architectural approaches, it is feasible to integrate ASICs within the design of a software defined radio. Such a design would need to support the ability to interchange flexibly functional components of the waveform processing implemented in ASICs. This would require some mechanism such as the ability to call algorithms implemented on an ASIC as though it were a function call.*

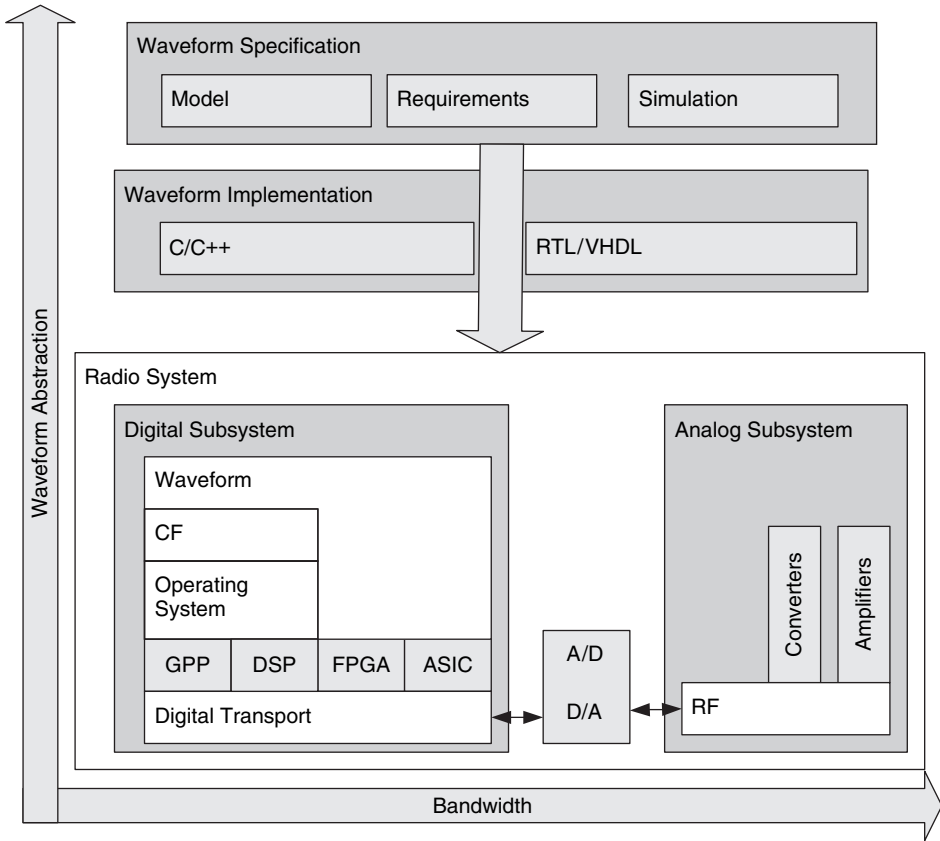


Figure 1.1. Waveform abstraction relative to bandwidth

model, or some other conceptual representation. As the waveform progresses from design to implementation, the capabilities of the waveform, in terms of throughput and capacity, typically drive the implementation to a high-level language for deployment on a GPP or DSP. Higher throughput demands drive the deployment towards and FPGA or an ASIC.

The GPP processor typically provides the management and control services for the system. Overlaid on top of the processor is an operating system and, integrated with the operating system, is a collection of software that provides the run-time infrastructure for the radio set. The infrastructure, in SCA terms, is called the Core Framework. On top of the Core Framework sits the waveform and other applications.

### 1.1.1 Software Radio Aspects

A software radio system can be viewed through one of four perspectives or aspects. Each aspect forms a functional grouping of objects and services provided by the radio system. Illustrated in Figure 1.2, these aspects are:

## cd Software Radio Aspects

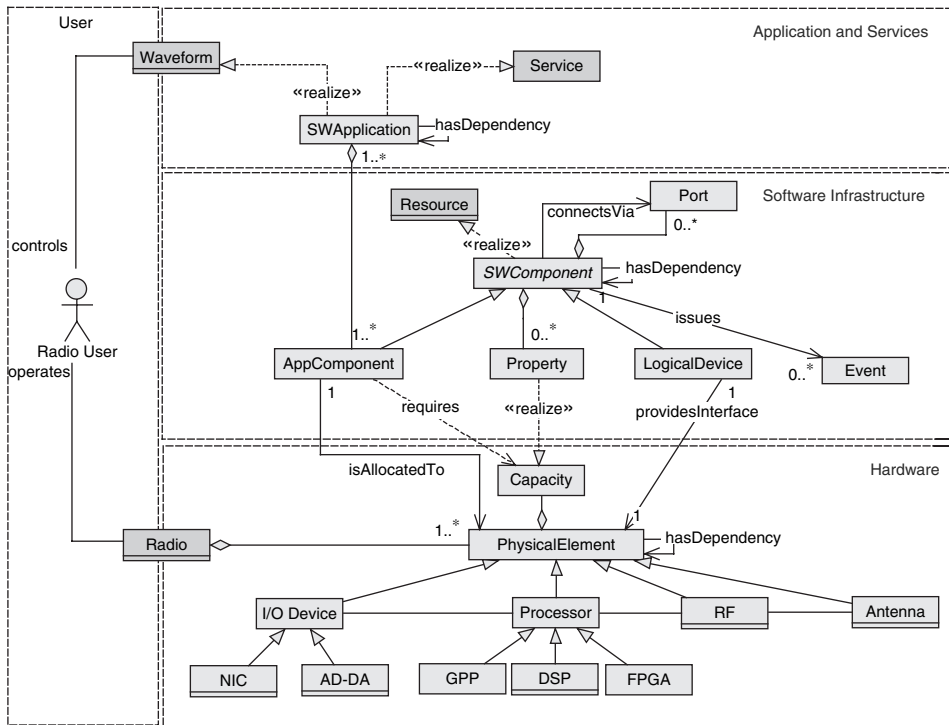


Figure 1.2. Software defined radio aspects

- **Hardware** – This aspect describes the physical set of devices and components that comprise the radio set.
- **Software** – This aspect defines the set of services and interfaces through which all waveform applications must interface to the underlying hardware.
- **Application** – This aspect defines the application and service layer. All waveforms and common services execute in this aspect.
- **User** – This aspect is the view through which the user interacts with the radio set. There are two basic modes of interaction within this aspect. The user is either performing radio control operations, e.g. setting system parameters, or performing application control and data transfer, e.g. setting the gain parameter for a specific waveform instance.

The SCA can be viewed as one realization of the Software Infrastructure aspect with some parts within the Applications and Services aspect. It defines a logical infrastructure for management and abstraction of physical hardware components, a standard set of abstractions for software components that form the digital processing portion of a waveform implementation, general services available for use by the system, and a set of common interfaces for managing, deploying, and configuring waveform applications within the system.

## 1.2 The Software Communications Architecture

Any new concept or technology has a learning curve associated with it and the SCA is no exception. The SCA defines a software infrastructure for the management, control, and configuration of a software defined radio. It does not mandate any specific architecture, design, or implementation for the radio system hardware or waveform application. Before launching into the detailed discussion of the SCA, it is advisable to spend a short bit of time providing some background data and explanation on what the SCA is, and is not, the history of its evolution, and the reasons why you would (or would not) want to apply the SCA to your system.

The SCA is based on several related technologies: Object-Oriented (OO) techniques in software engineering, the Common Object Request Broker Architecture (CORBA), and the CORBA Components Model (CCM). Object-oriented languages have been around for a number of years from Simula in the late 1960s, Smalltalk and Flavors in the early 1980s, to current object-oriented languages such as C++, Python, Ruby, and Java, to name a few.

As systems evolved towards distributed architectures and a client-server model, CORBA evolved as an industry standard for describing the interfaces provided or used by two components using a pseudo-code called an Interface Definition Language (IDL). IDL provided the means for specifying the available interfaces and, through the IDL 'compiler', generated source code that is compiled into each of the applications. The code generated includes the support routines necessary to support remote procedure calls between processes on the same computer and between computers, i.e. in a distributed environment. Thus, the developer was freed from the drudgery of writing low-level, inter-process communications code and, more importantly, CORBA code built by one individual could interoperate with code built by another individual, the only requirement being that both the author of the client application and the server application use the same IDL. This was an important step forward in the ability to develop modular software while encapsulating the internal logic and requiring only that each of the developers agree on a set of IDL.

Although the CORBA technology provided several important advances, it became apparent that the mechanism by which systems were deployed was still dependent on manual configuration. The CCM evolved to address the need for specifying the requirements for deploying a set of application software by describing what resources were required to deploy the system successfully on a set of hardware. The method for describing the components of a system and the related deployment requirements is through a set of eXtensible Markup Language (XML) files. XML is a text-based language that utilizes tags to define items, their attributes, and values. This CCM XML was the genesis of the SCA Domain Profile XML.

With this brief summary of background information and foundation technology as a backdrop, the next sections provide a summary of what the SCA is, is not, why you would (or would not) want to use it, and a brief history of its evolution.

### 1.2.1 The Evolution of the SCA

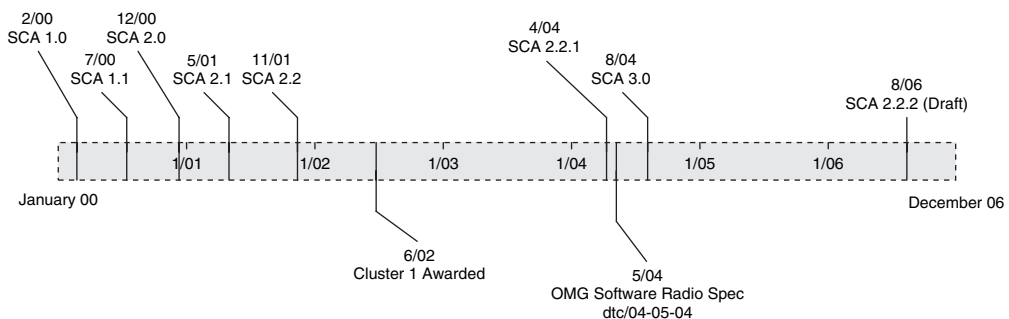
The United States military was (and is) facing an increasingly critical need to support communications for multiple missions, rapid deployment, diverse mission scenarios and objectives, increased interoperability, and to reduce the cost of operations. One of the

primary obstacles to meeting these challenges was that the bulk of the radio systems were predominantly hardware-based, limited to those waveforms that were designed into the system, and incapable of being upgraded or adding new waveforms without significant cost due to hardware re-design.

Concurrently, over the past two decades, the capabilities of processors have increased dramatically, special purpose processors such as DSPs and FPGAs have become commonly available, and the speed and resolution of Analog to Digital and Digital to Analog circuits have steadily increased. The result is that more of the waveform signal processing that once was exclusively the preserve of the analog domain was migrating into the digital domain implemented in software. Early experiments in software-based radios such as SpeakEasy showed that there were significant benefits to be gained by moving towards a software-based architecture. Many of the radio manufacturers had already started down the path of implementing core signal processing components in software. Early multi-channel radio systems developed in the 1990s, such as the Joint Combat Information Terminal (JCIT) and the Digital Modular Radio (DMR), provided a software infrastructure for the management of radio resources.

With the need to enhance reconfigurability, support multiple missions, and reduce long-term operations and maintenance costs as a background, the Joint Tactical Radio System (JTRS) Joint Program Office (JPO) was formed to develop a new family of software-based, reconfigurable, radio systems. One of the first activities was to define a common software infrastructure that would be applied to this new family of radio systems. Thus, the SCA was born.

The timeline in Figure 1.3 illustrates several key milestones in the evolution of the SCA specification. There were several preliminary versions but 1.0 was the first version of the specification used to develop an initial implementation of the SCA. After an incremental release with version 1.1, significant portions of the specification were re-worked resulting in version 2.0. Version 2.0 had a number of issues and required some additional details and specifications to address all the aspects required of a software infrastructure. Nonetheless, there were several 2.0 implementations that provided valuable feedback to the specification development process. Again, an incremental version was released in mid-2001, version 2.1,



**Figure 1.3.** SCA specification timeline

followed by 2.2 in November of the same year. Version 2.2 was generally considered to be complete enough to implement and apply to a fielded software radio system.<sup>2</sup>

In June 2002, the first major program to apply the SCA was awarded to Boeing by the Communications and Electronics Command (CECOM). The Cluster 1, later renamed the Ground Mobile Radio (GMR) program, was the inaugural project using version 2.2 of the SCA. Other JTRS Cluster programs were awarded, then in April 2004, almost three years after the 2.2 specification, version 2.2.1 was released. This version cleaned up many of the errors in 2.2 and incorporated several clarifications and enhancements. One of the significant changes was that with version 2.2.1, the Log Service was removed from the SCA specification and the OMG Lightweight Log specification was referenced instead. In mid-2004, the OMG released its Software Radio Specification. The OMG specification was initiated by a number of the individuals who had contributed to the SCA development. The original objective was to evolve the SCA into an industry standard rather than a military-only specification. However, as the specification evolved in the OMG, it took on a life of its own with the resultant OMG specification being significantly different from the SCA specification.

At the same time, issues with waveform portability were being raised through the on-going JTRS Cluster programs. The basic problem was that the code developed for a GPP was reasonably portable between platforms. However, the code developed for a DSP and FPGA generally remained specific to the particular processor and architecture of the radio.<sup>3</sup>

This portability issue came to a head in late 2004, resulting in several special workshops called by the JPO to address the DSP and FPGA portability issue. The result of these workshops was the SCA 3.0 specification. This version of the SCA changed little of the core requirements describing the SCA. It did, however, define additional constraints on DSP software related to what system calls could be used by DSP code, defined a proposed set of waveform components, proposed a high-level data transport design (called HAL-C), and had an Antenna API section. The general reaction in the community was that the specification required additional work and, although the concepts and approaches were potentially useful, more detail and analysis was required in order to achieve a set of descriptions that could be implemented efficiently.

From late 2005 to early 2006, the JPO was re-organized to address resolution of problems with the Cluster programs more effectively and move forward. The program office was moved to San Diego, CA, from Washington, D.C. and is now administered by the Navy SPAWAR office. In mid-2006, version 2.2.2 was released. As the number implies, this is an incremental version from the 2.2.1 version of the SCA. Furthermore, the 3.0 version is shown on the JTRS website as 'not supported.' Thus, at the time of this publication, version 2.2.2 is the latest release supported by the program office.

---

<sup>2</sup> Although in principle this was true, there remained a number of important clarifications, corrections, and additional requirements that were necessary to transform the 2.2 release into a reasonably solid specification. For example, the IDL provided in appendix C of the SCA specifications would not compile as defined due to name conflicts with POSIX named values.

<sup>3</sup> Waveform portability became a significant issue within the SCA community. Although the portability of waveforms across multiple platforms is certainly a desirable objective, the interpretation of portability evolved into reuse without modification. This was never a fundamental requirement of the SCA. Waveform portability can be viewed as a cost function with the objective to minimize the cost as much as possible. Due to differences in physical hardware, processor, and transport architectures between different board manufacturers, simply moving a waveform from one system to another without modification is unrealistic.

### 1.2.2 *What is the SCA?*

The main purpose of the SCA specification is to define the Operating Environment (OE) software, also commonly referred to as the Core Framework, which implements the core management, deployment, configuration, and control of the radio system and the applications that run on the radio platform. In order to provide a common reference for describing what the SCA is and isn't, it is useful to refer back to the introduction provided with the SCA specification. The quote below is an excerpt from the Introduction.

The Software Communication Architecture (SCA) specification is published by the Joint Tactical Radio System (JTRS) Joint Program Office (JPO). This program office was established to pursue the development of future communication systems, capturing the benefits of the technology advances of recent years, which are expected to greatly enhance interoperability of communication systems and reduce development and deployment costs. The goals set for the JTRS program are:

- greatly increased operational flexibility and interoperability of globally deployed systems;
- reduced supportability costs;
- upgradeability in terms of easy technology insertion and capability upgrades; and
- reduced system acquisition and operation cost.

In order to achieve these goals, the SCA has been structured to

- provide for portability of applications software between different SCA implementations;
- leverage commercial standards to reduce development cost;
- reduce development time of new waveforms through the ability to reuse design modules; and
- build on evolving commercial frameworks and architectures.

SCA V2.2, November 17, 2001, p. vii

As the above quote states, the key objectives are to increase flexibility and interoperability, reduce support costs, provide upgradeability through technology insertion, and reduce system acquisition costs. None of the stated objectives are related to any technical design, development, or waveform aspect of the radio system. Thus, the first fundamental objective of the SCA is to improve the business case for evolving and enhancing communications systems and their procurement.

In order to achieve this objective, the SCA structure is intended to 'provide' for portability of applications software, leverage commercial standards, support the reuse of waveform design modules, and build on evolving commercial frameworks. As with the objectives noted in the previous paragraph, the SCA structure focuses on design and development processes to reuse design modules and leverage commercial software and standards.

### 1.2.3 *Common SCA Perceptions*

The previous section provided a brief description of the SCA. It is a truism that any technology is often received and perceived differently by each individual: Some of the

perceptions are based in fact and some are based on an incomplete understanding of the technology. The following paragraphs discuss some of the commonly cited misconceptions about the SCA.

### **1.2.3.1 The SCA defines a technical architecture for a software radio**

There is nothing in the SCA specification that provides technical data or guidance on the design and implementation of a software radio. The SCA, based on the CORBA Components Model, defines an architecture for the deployment of applications. In the case of a software radio, those applications tend to be waveforms.

### **1.2.3.2 The SCA enables reusable components**

The SCA enhances reusability from two perspectives. First, the SCA specification defines a common set of interfaces for basic deployment configuration, and control of applications. So, from the perspective of user interfaces and external control of the system, the same interface calls that are used to load, start, and stop a SINCGARS waveform are identical to the FM3TR waveform. Second, the Application Programmer Interface (API) appendix to the specification is intended to promote reusability of waveform software components through common waveform interfaces. This continues to be an area of on-going discussion because all radio system developers have different perspectives as to what the interfaces should be for a specific waveform.

### **1.2.3.3 A waveform can be moved from one SCA platform to another without modification**

Many individuals have interpreted the portability objective of the SCA as reusability without modification. The SCA specification defines common, high-level interfaces for deploying, configuring, controlling, and monitoring the hardware and software applications within an SCA-based radio system. This simplifies the effort required to port applications because the interfaces do not change. However, deploying waveforms across multiple radio systems without modification was never a stated requirement.

### **1.2.3.4 The SCA results in a waveform performance impact on my system**

The simple fact is that, once the SCA deploys the waveform on the radio system, the SCA Core Framework goes into a quiescent state and does not utilize significant processor cycles. Also, for waveforms implemented largely in FPGA or DSP processors, there is typically no impact due to the SCA on functioning waveforms in those processors. There are some impacts in terms of the memory footprint required to support an SCA framework. However, the SDR Forum, NASA, and other groups are looking into reduced footprint architectures. Where the framework is running on the same GPP being used for waveform processing, some performance impact may be encountered. In this case, standard systems analysis to evaluate the load margins is necessary.

### **1.2.3.5 I must use CORBA for the data transport**

CORBA should be the starting point but is not mandatory if performance reasons prohibit it. Also, individuals often confuse the latency impacts of the underlying transport mechanism, which typically defaults to TCP/IP, as being synonymous with CORBA. In reality, CORBA is a protocol layer, much like Hypertext Transfer Protocol (HTTP), that rides on top of the data transport mechanism. Most modern ORBs support plugable transports allowing customization and optimization of the actual data transport.

### **1.2.3.6 The SCA is only applicable for small radios**

The SCA is not specifically targeted for any one type or class of radio system. Small form-factor, resource-limited radio systems have a more significant set of issues to overcome when building an SCA-compliant handheld or manpack radio, due to their Size Weight, and Power (SWaP) constraints. This is usually due to the fact that the GPP on the small radio is already used extensively for waveform code, and processing impacts due to adding the framework can be significant.

### **1.2.3.7 The SCA and/or CORBA is not suitable for large, complex systems**

The origins of the SCA are based on the JTRS program which focused on tactical radio systems. These systems ranged from small handhelds to rack-mount systems in vehicles, ships, and aircraft. Although these systems do not have the complexity of large terminal systems, it is possible to apply the SCA to larger systems. More thought must go into the architecture of the system, however. It may be the case that the SCA manages the core set of radio equipment and waveform deployment under the direction of a higher-level system or network management operation. The key aspect is that the SCA is targeted towards the management of the hardware and software that implement and support the end-to-end waveform application.

As for the applicability of CORBA to large scale systems, it can be said that it is in wide use throughout industry. Large, distributed Java-based applications are, in fact, using CORBA and Java remote procedure calls are using the CORBA protocol. Also, the Iridium satellite Command and Control segment integrated a COTS-based system, OS/COMET, within a comprehensive CORBA framework. The resultant system ran on over 50 computers and was comprised of several hundred processes.

### **1.2.3.8 The SCA is not suited to systems above 2 GHz**

This reason is typically rooted in the fact that the SCA originated in the JTRS program, which was focused on the tactical radio spectrum under 2 GHz. The simple fact of the matter is that there is nothing in the SCA specification, either explicitly or implicitly, that limits the usefulness or applicability of the SCA to systems above 2 GHz.

## *1.2.4 Why Use the SCA?*

Given that the SCA is not a technical reference architecture for a software radio, why should you use it? Part of the answer lies in the fact that, as discussed earlier, the objective of

the SCA is to improve the business case for reduced cost for enhancements, upgrades, and logistics. These are benefits that are primarily realized by the customer or recipient of the SCA system. Thus, more often than not, the reason cited for using the SCA is that it is a requirement levied on the project. In order to achieve longevity and acceptance, there must be business reasons for the developers of software radios to use the SCA. Some of the business reasons to consider using the SCA architecture are:

#### **1.2.4.1 The SCA provides a common infrastructure for distributed application deployment**

Although the SCA was founded and focused on the radio domain, the basic infrastructure for deployment of the application components is applicable across virtually any domain. This includes radio-related areas, such as signal processing systems, as well as unrelated domains, e.g. device and software management across processors within a vehicle.

#### **1.2.4.2 The SCA allows quicker integration of external applications**

Because of the small set of interfaces defined within the SCA, external applications can easily be integrated using the IDL specified for loading, starting, stopping, and controlling applications within an SCA system. One such example is the integration an SCA system within a network of systems. Overall management of the radio nodes is often performed by a network management or network control operation. These management systems often utilize Simple Network Management Protocol (SNMP) or Java, which includes native support for CORBA. So, when the network control software decides that it needs to load a particular waveform or communications software on an SCA radio, it simply issues a Java Remote procedure Call (RPC) to the SCA radio to load the desired waveform. If SNMP is used, then an SNMP proxy can be implemented that provides the SNMP interface to the network system and issues Java calls to the SCA radio.

#### **1.2.4.3 The SCA enables easier insertion of new technology**

Because the SCA specifies the interfaces for deployment, configuration, control, and monitoring of the hardware and software with an SCA system, new technology can be inserted with less cost and impact. As an example, part of the description of the application in the SCA identifies a software component that provides a certain function. Within that description, there may be several different implementations, e.g. one on a DSP, one for a Intel GPP running VxWorks, and so on. As technology capabilities progress, an implementation that could once only be realized on a DSP may now be realizable on a GPP.<sup>4</sup>

Because the SCA allows multiple implementations, the new implementation can be deployed on an existing system that has the GPP resource but not the DSP without changing other components. This concept also applies to new hardware.

---

<sup>4</sup> This does not imply that no effort is required to port a component from one system to another. It does mean that, if the implementation suitable for a particular system is now available, it can be deployed and configured on the system without rebuilding the entire application.

#### **1.2.4.4 The SCA provides an infrastructure for extensibility and integration**

The SCA is a foundation for the design and development of a comprehensive radio system. It defines the essential interfaces and behaviors provided in the infrastructure. Applications may be developed and built that provide higher level capabilities and features. Technologies such as cognitive radio may be integrated with an SCA system providing higher-level control and management of the radio resources. For example, a set of SCA radio systems that have a cognitive map of their surroundings and temporal logic can negotiate configurations between them to enhance reliability of the communications. This may take the form of changing parameters of the existing waveforms within the radio or loading new waveforms based on the collaboration with other radio systems.

#### **1.2.4.5 The SCA reduces the development of Non-Recurring Engineering (NRE) for system development**

Standardizing on a common software infrastructure reduces the effort required for future systems because a common set of control routines and implementations are developed. This can have a positive impact on the development cost and schedule. The key is to develop a standard infrastructure that is applied across a variety of systems that supports the SCA capabilities. This last item is central to how an organization approaches the development and use of the SCA.

Now that we've covered several of the positive and negative viewpoints of the SCA, let's continue with a brief overview of the technical aspects of the SCA. The remainder of this chapter will introduce the SCA Operating Environment and specification structure as a foundation for the subsequent chapters.

### **1.3 The Operating Environment**

The JTRS program Software Communications Architecture specifies the requirements for a common software radio Operating Environment (OE). The OE consists of the Core Framework (CF), the CORBA ORB, and the operating system. The operating environment specifies the interfaces, rules, constraints, and procedures that must be adhered to in order to implement an SCA-compliant radio system. The Core Framework provides

- a collection of common services used by the waveform and other applications;
- software enabling the installation, configuration, management, and control of waveforms;
- a federated file system enabling common file system operations and access across multiple processing platforms;
- device interfaces that provide a common abstraction of the underlying physical hardware.

As illustrated in Figure 1.4, the SCA Core Framework is shown along the vertical axis and can be thought of as the management plane of the SCA system. The waveform is illustrated as a set of components that are assembled across the horizontal axis forming the application plane.

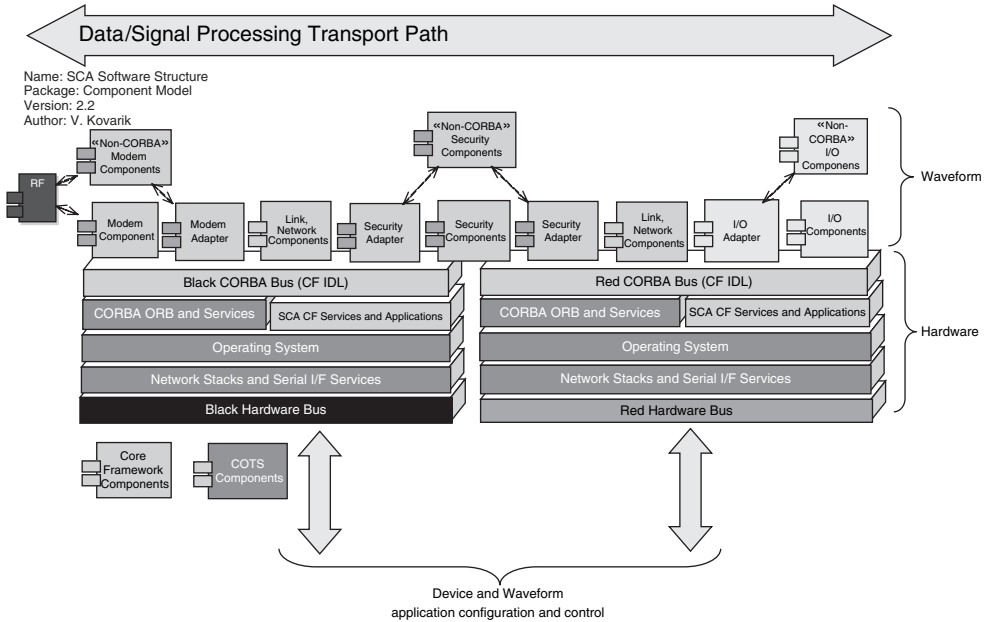


Figure 1.4. The SCA waveform component organization

1.3.1 Conceptual Organization

Conceptually, a SCA radio has three segments: i) The Waveform Deployment; ii) the Core Framework; and iii) the Domain Profile. These three segments are each divided into physical and logical views. In the Waveform Deployment segment, the radio hardware is the physical view of the radio system. However, the waveforms are realized through software that is loaded on the physical radio elements. There are two layers in the logical view of the radio system. The first consists of the set of components that form a waveform application or other service on the system. The second is the application that provides the top-level interface and control for the set of components.

The Core Framework segment includes all software required to manage the radio system and deploy applications. It has a physical view and a logical view as well. The physical view of the Core Framework provides high-level management of the physical devices in the radio system. The logical view provides the same for the waveform applications and other services.

The Domain Profile segment consists of the set of XML files that describe the hardware resources within the radio system, the waveform application structure, and dependencies between waveform components, connections between components, and dependencies on hardware resources. This is illustrated in Figure 1.5.

1.3.2 OE Interface Constraints

Figure 1.6 illustrates the relationships between the primary components of an SCA-compliant system. At the base level are the services provided as POSIX interfaces by the operating system that form the AEP.

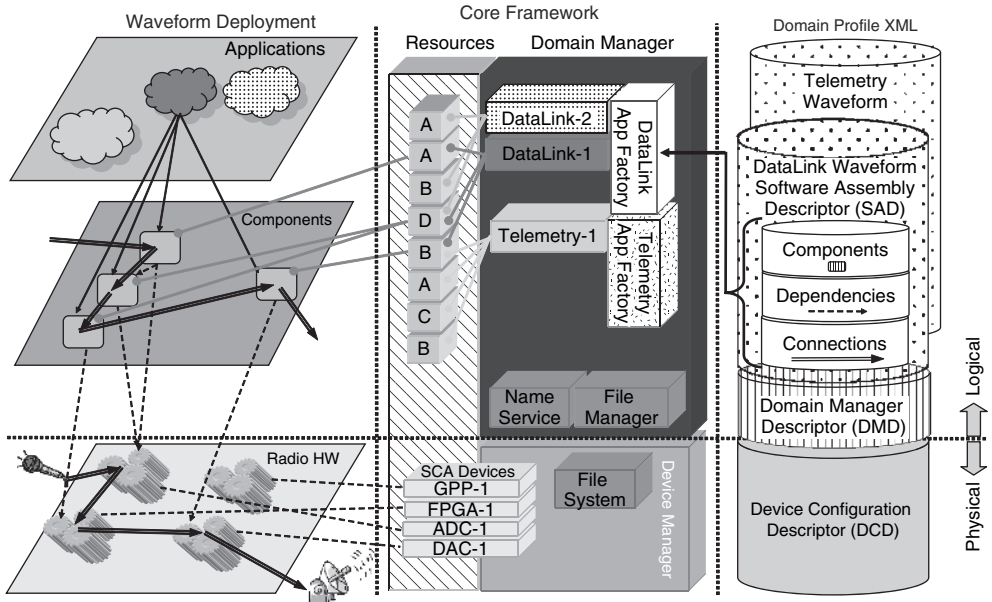


Figure 1.5. Abstraction layers in an SCA system

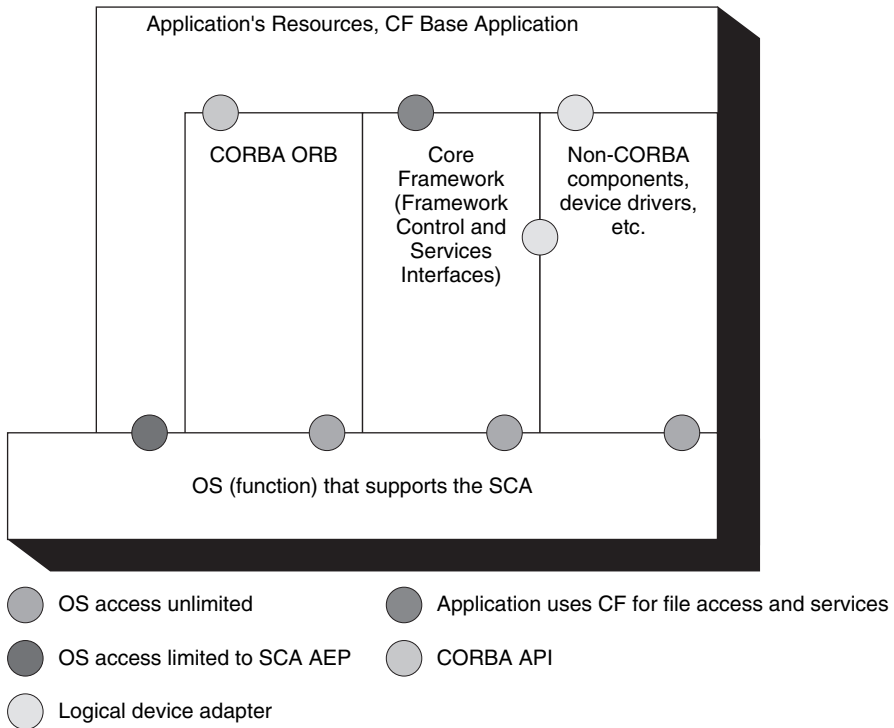


Figure 1.6. The SCA interface constraints

The objective of the Application Environment Profile is to provide a constrained set of well-defined operating systems calls that minimize the impact to application code. This objective is only valid for general purpose processor components because DSP and FPGA processors do not have an operating system. However, some DSPs do support an operating system and CORBA.<sup>5</sup>

The CORBA ORB provides a common middleware for the system and has unlimited access to the underlying operating system. The core framework is an implementation of the SCA specification providing the essential infrastructure components and services for the radio system. The non-CORBA components and device drivers are comprised of low-level drivers such as those provided by a device manufacturer for a particular physical device for one or more operating systems. The application is composed of the set of application components and resources that form an operational waveform. Finally, the operating system provides the underlying set of platform specific services.

The CORBA ORB, Core Framework, and Device Drivers have unlimited access to the underlying operating system calls and services. However, as illustrated in Figure 1.6, the application is limited to a set of POSIX calls to the operating system. The rationale behind this limitation is that the application will be more easily ported to other platforms if it is constrained to a specific set of interfaces and the software radio platform is mandated to support the set of POSIX calls specified in order to be SCA-compliant.

Although this rationale has some merit, the porting of a waveform has a wide ranging set of complex issues and the POSIX constraints, in and of themselves, help but do not achieve, portability of waveforms between platforms. This becomes particularly evident when a waveform application is developed that uses a DSP or a FPGA as the processor for some portion of the waveform functional chain.

## 1.4 The SCA Specification Structure

The SCA specification consists of three major components:

- Software Communication Architecture Specification (JTRS-5000SCA)
- Application Program Interface Supplement (JTRS-5000API)
- Security Supplement (JTRS-5000SEC)

The SCA specification is the primary specification for building an SCA-compliant radio systems. The SCA specification defines the operational environment requirements and basic functional requirements. The contents of the SCA specification are the primary focus of this book.

The Application Program Interface (API) Supplement provides guidelines and requirements for building modular and portable application components. Certain aspects of the API Supplement will be referenced or discussed within this book. However, a thorough treatment of the API Supplement and the construction of a portable SCA application would require significantly more detail than can be reasonably included in this book.

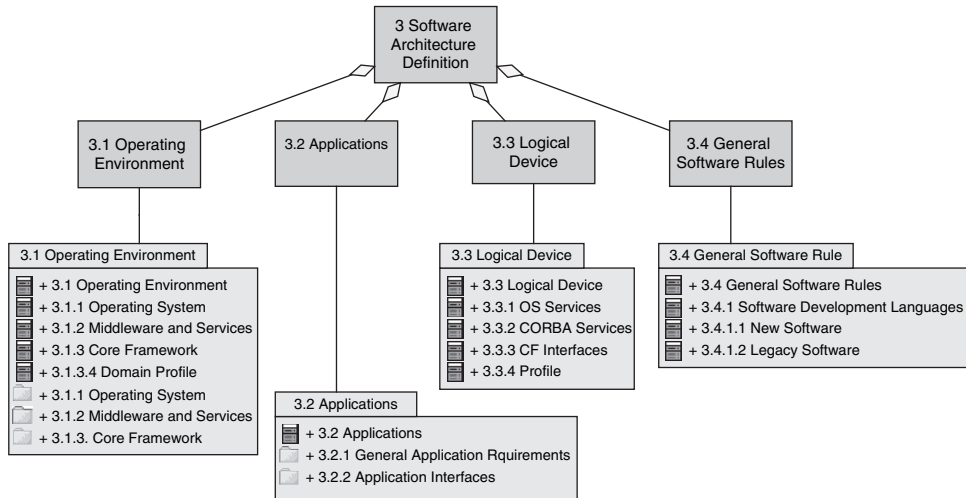
---

<sup>5</sup> The POSIX interface is required to be used by the application when accessing operating system services. All other components, e.g. the CORBA ORB, the Core Framework, non-CORBA components, and device drivers, may have unlimited access to the operating system.

The Security Supplement defines specific security requirements and APIs related to the design and construction of a Type 1 secure radio system. This book will reference some portions of the Security Supplement as it relates to the rest of the content. However, as with the API supplement, the level of detail required to address fully the security issues associated with the design and development of an SCA-compliant system are beyond the scope of this book.

Section 3 of the SCA specification identifies the requirements for the core software infrastructure architecture definition. The organization of Section 3 is shown in Figure 1.7. The subsection on the Operating Environment addresses the Core Framework and services. Section 3.2 focuses on the application interfaces and functional requirements. Section 3.3 provides the requirements for the SCA device interface and Section 3.4 identifies general requirements.

**cd 3.0 Software Architecture Definition**



**Figure 1.7.** The SCA specification organization

The Operating Environment, Section 3.1, of the SCA specification, defines the bulk of the requirements that must be met for a system to be SCA-compliant. It contains the common components of the Core Framework including the Domain Manager, the Log Service, CORBA requirements, and other components.

In Section 3.2, the requirements that must be met by the SCA application are defined. The application is comprised of the software that implements an end-to-end waveform. Thus, this section is met primarily by the waveform application developer. However, certain common aspects of the application are provided as part of the Core Framework components.

In order to manage, configure, and control the devices that make up a radio system, a Logical Device interface must be implemented. Thus, the device manufacturer and/or radio system integrator must provide implementations of the SCA Logical Device in order to integrate the hardware into the overall SCA system. These requirements are defined in Section 3.3. Sections 3.1 through 3.3 will be presented in more detail as each of the functional

requirements areas are discussed in later chapters. The general software requirements, Section 3.4, and other, non-functional requirements, are discussed in this section.

Figure 1.8 shows the relationships between the SCA specification and an implementation. The SCA specification provides the interface and high-level behavioral specification to be implemented by the Core Framework. Figure 1.8 also shows the standards developed by the Object Management Group (OMG) that are referenced by the SCA specification.

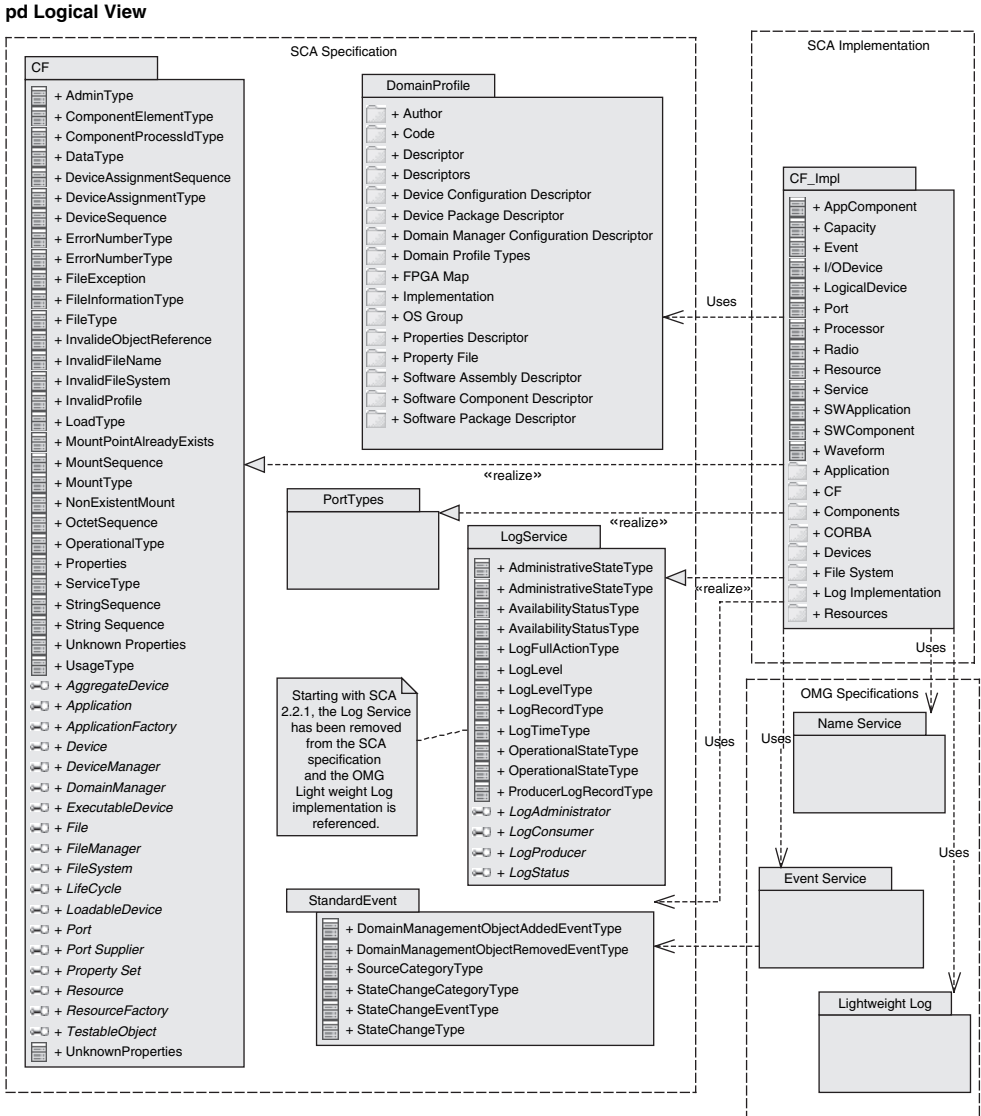


Figure 1.8. SCA specification versus implementation

The bulk of the Core Framework interfaces and behavioral specifications are contained within the Core Framework package. The IDL for the set of interfaces is defined as a single interface module. The Domain Profile specifies the XML files that are used to describe the underlying hardware and the software components and the interconnections that are required to deploy the waveform. There are some differing interpretations of what constitutes the Domain Profile. Some interpretations refer to the set of XML files as the Domain Profile, some view the internal parsed XML data structures as the Domain Profile, and others view the Domain Profile as a collection of internal data structures that form the internal domain knowledge of the SCA radio system.

This book takes the stance that the XML files represent a human readable version of the Domain Profile and that the internal data structures built as part of the processing of the XML profile form the internal Domain Profile used by the Core Framework in the process of instantiating a waveform. The hierarchical tree structure generated by the parser is viewed as an intermediate data structure that takes the text form of the XML files and parses them into a canonical form that allows easy extraction of the salient information required to deploy a waveform. It is true that the XML parse tree can be traversed and used directly as the internal representation of the Domain Profile. However, this form of the data is inefficient for the types of constraint enforcement required of the Application Factory during application instantiation.

Underlying all the above abstractions are a set of common services that provide critical capabilities for all the SCA components:

- **Name Service** – The Name Service enables a component within the radio system to locate a required service or application component and connect to that component.
- **Event Service** – The Event Service provides an asynchronous mechanism for components to publish events on an Event Channel and other components to register to receive those events.
- **Log Service** – The Log Service provides a basic logging capability within the SCA radio system that allows components to create a record describing some activity or state and have that record time-stamped and saved within a log for subsequent retrieval.
- **File Service** – Perhaps one of the most crucial services provided is the File Service which provides a common abstraction of a file system for all SCA radio components independent of the actual underlying operating system and file system implementation.

Part I of this book is organized along the high-level abstractions identified above. Chapter 2 addresses the common framework services to provide a frame of reference for components and references used in the description of other logical interfaces. After describing the common services, each of the subsequent chapters addresses each of the logical abstractions defined above starting with the Resource and ending with the Domain Manager.

## 1.5 Summary

The SCA specification provides an implementation-neutral framework for implementing and deploying applications. The general requirements of the SCA define a set of hardware and software constraints that the system must adhere to or provide. In the next chapter, a conceptual view of the operation of an SCA-compliant system is presented using the Use Case model approach of the Unified Modeling Language (UML).

