Radio Management Across Multiple Standards: A Microkernel Approach - D R A F T -

Vincent J. Kovarik, Jr.
Harris Corporation,
Melbourne, FL
vkovarik@harris.com

Abstract

Current trends in software radio infrastructures design have evolved over the past several years and resulted in several standards. One of the primary infrastructure architectures is the Software Communications Architecture (SCA) developed in conjunction with and in support of the Joint Tactical Radio System (JTRS) project. One of the basic premises of the SCA and other similar standards, is the specification of a set of common, open interfaces and behavioral requirements. The objective being to define a set of common interfaces to which multiple vendors and suppliers can provide components, develop software, new waveforms, and have a common method for control. While these are valuable capabilities, a common criticism of the SCA is that it incurs too much overhead for small-form factor radio deployment. This paper asserts that one of the factors contributing to the size of the implementation has been the approach to design from the interface rather than designing to the interface.

1. Introduction

With the advent of digital signal processing, increasingly more of the signal processing within the radio system has migrated into the digital domain and is performed through software. The increased use of software enables a degree of configurability and flexibility heretofore not seen in communications systems.

In order to maximize the capabilities of the processing resources and leverage the flexibility of digital processing, there must be an overarching control or management infrastructure. Management of a software radio has multiple aspects that need to be addressed, including the management of physical or hardware resources and the logical or software resources of the radio.

2. Radio Infrastructure Standards

Several standards have been proposed and developed over the past decade. The most familiar are the Software Communications Architecture (SCA), PIM and PSM for Software Radio Components (SWRADIO), and Space Telecommunications Radio Systems (STRS) specifications. Although each the standards have the common objective of managing a software radio, each has different perspectives in terms of target deployment environments and interfaces.

Development of the SCA was originally sponsored through the Joint Tactical Radio System (JTRS) Joint Program Office (JPO). The SWRADIO specification was developed through the Object Management Group (OMG), an industry standards organization, and STRS was initiated and largely developed by National Aeronautics and Space Administration (NASA).

The first of these standards was the SCA which was an adaptation of an existing standard, the CORBA Component Model (CCM). The CCM

standard, developed by the OMG, forms the basis of the SCA. The remainder of this section will discuss each of the above standards from the context of the rationale and objectives that resulted in the initiation of the standard. The three will be discussed in the general order in which each standard was introduced, beginning with the CCM.

2.1. CORBA Component Model

The Common Object Request Broker Architecture (CORBA) standard initially gained widespread acceptance in the early 90's. One of the primary benefits provided by CORBA was an implementation-neutral approach to specifying and developing distributed applications. Targeted for a General Purpose Processor (GPP) environment, CORBA allowed components of an application to be distributed across a network of computer resources without requiring developers to write software specific to the network interfaces, data format and transport protocols.

Although CORBA simplified the development of distributed applications, the deployment of these applications were still dependent on the specific host target capabilities. So, while the application components could locate and and connect to other components of the application, the actual deployment of the application remained a manual process.

To address this limitation, the OMG initiated development of the CORBA Components Model (CCM)[1]. The intent of this specification was to describe a distributed application as a set of components using a set of eXtensible Markup Language (XML) files. The XML files described each of the components in the application, the deployment constraints of each of the components in terms of operating system, processor, etc. and the connection dependencies of each components which identified the order and precedence of the flow of control and data through the distributed application. It was this distributed application deployment approach that was used as a foundation for the SCA.

2.2. Software Communications Architecture

In the mid-1990's, the benefits of software radio were recognized as having substantial benefits to the military radio domain as a means by which the military could develop a software-based radio system that would be capable of evolving to new waveforms and missions through the update and addition of software. This had the potential of having a very significant cost benefit to the military because it would break the dependency on radio hardware to implement new waveforms and capabilities, which was very expensive. However, it was also recognized that, in order to realize the potential benefits, a standardized management infrastructure was necessary to in order to ensure commonality of interfaces and behaviors.

In the late 1990's work was begun to develop a standard software radio management infrastructure through a group called the Modular Software Radio Consortium (MSRC). The MSRC was an industry/government consortium that began developing a radio management infrastructure specification in the late 1990's that would eventually evolve into the SCA [2].

2.3. PIM and PSM for Software Radio

One of the objectives of the JTRS JPO was to promote the evolution of the SCA into an industry standard. The rationale was that, an industry standard would gain wider acceptance and thereby promote continued refinement, improvements and extension to the specification through industry participation. To achieve this objective, collaboration with OMG was initiated through the companies and individuals who developed the SCA specification to socialize and promote the specification and evolve it into an OMG specification document.

As work progressed within the OMG it became apparent that, although the SCA provided an initial starting point for the specification, the standards process would result in the evolution of the specification away from the specific form of the SCA specification. The OMG had developed a process called Model Driven Architecture (MDA) which described a methodology for developing an archi-

tecture through a series of models that reflected different perspectives of the specification. At the time, the two key perspectives were the Platform Independent Model (PIM) and Platform Specific Model (PSM). The PIM, as the name implies, defines a model that describes the system without specifying the implementation technology. While the PSM extends the PIM by identifying specific implementation technologies that *realize* the PIM for a specific target platform.

Because the SCA identified specific implementation technologies, e.g. CORBA, the SCA was viewed as a PSM. Consequently, the bulk of the standards work performed in development of the PIM which is an abstraction of the SCA, resulting in the SWRADIO specification [3].

While the OMG work provided additional value through the PIM and did involve the participation and contributions from a variety of industry representatives, it is debatable whether the specification had any positive effect on the acceptance and adoption of the SCA by industry or government outside of the U.S. military.

2.4. Space Telecommunications Radio Standard

A common view of the SCA has been that it is not suitable for deployment in systems with stringent Size Weight and Power (SWaP) constraints. This view was primarily based on the perception that the memory and processor requirements for hosting SCA were greater than could be supported in small form factor platforms.

Although some implementations of the SCA did require substantial resources, this perception was not entirely valid [4] and there is no inherent shortcoming that prevents use of the SCA in a small form factor radio. Nonetheless, a parallel effort to develop a software radio infrastructure suitable for space deployment was initiated by NASA. The primary, driving requirement of the effort was the sever SWaP constraints of a space-deployed communications system. The results was the STRS specification which defined an initial draft specification for a space-deployable radio infrastructure.

3. Framework Comparison

Each of the specifications were driven by organizations with different perspectives and agendas such as target deployment environment, SWaP constraints and the types of applications to be run on the radio platform. While each of the specifications appear to be different on the surface, upon closer inspection, similarities in the interfaces and behaviors quickly emerge.

3.1. Interfaces

One of the fundamental areas of similarity between each of the specifications is the interface definitions and behaviors. A short cross-section of a subset of the interfaces for each of the specifications was inspected to identify the similarities and differences between the specifications.

Briones et al [5], provide an overview of the relationship between the OMG SWRADIO specification and the STRS specifications. Table 1 provides a simple table of the interfaces for a subset of the functionality across each of the standards.

As can be observed in Table 1, there are substantial similarities in the names and behaviors of the interfaces across the different specifications between the subset. This similarity of name and function raises the question as to what are the fundamental differences, if any, that require the development of a specification for a different domain.

3.2. Fundamental Questions

Based on the history of software radio infrastructure development and the brief comparison of selected interfaces in table 1, there are several fundamental questions that are raised.

1. Are there fundamental differences between the above that indicates a real necessity for multiple software radio infrastructure specifications?

2. Are there commonalities across the above specifications that indicate a common set of interfaces and behaviors, regardless of the implementation or deployment?

Table 1. Comparison of interfaces

STRS	OMG	SCA	Description
STRS_ControllableComponent • WF_Start(); • WF_Stop();	**ControllableComponent** • start(); • stop();	**Resource** • start(); • stop();	The start and stop interfaces provide the function calls to start and stop processing of a hardware or software component of the system.
STRS_Lifecycle • WF_Initialize(); • WF_ReleaseObject();	**Lifecycle** • initialize(); • releaseObject();	**Lifecycle** • initialize(); • releaseObject();	The Lifecycle calls provide the essential interfaces for the proper initialization and finalization of the hardware and software resources within the radio system.
STRS_PropertySet • WF_Configure(); • WF_Query();	**PropertySet** • configure(); • query();	**PropertySet** • configure(); • query();	The PropertySet interfaces support the definition and use of properties associated with an object in the radio framework. Each property is a name/value pair with a string mnemonic for the property name and any value for the property.
STRS_TestableObject • WF_RunTest(); • WF_GroundTest();	**TestableObject** • runText();	**TestableObject** • runText();	The TestableObject interfaces provide the ability to initiate Built in Test (BIT) on any resource within the system.

3. If there is a need for different or additional interfaces to address specific target deployment environment or requirements, can a common implementation be realized that supports deployment-specific extensions rather than have entirely new specifications?

The above questions lead to the hypothesis that a common architecture and implementation that provides the core functional elements required to manage and configure a radio system. This implementation would be consistent across multiple radio system platforms with additional extensions of interfaces supported for particular environments.

4. A Microkernel Approach

Early operating systems were monolithic with each operating system developed to a specific hardware architecture providing all of the capabilities necessary to manage and run applications, interface to devices and manage tasks and processes.

As processor architectures proliferated, it quickly became cost prohibitive to develop an operating system from the ground up for each operating system that would be run on a processor. This led to the microkernel approach to operating system design.

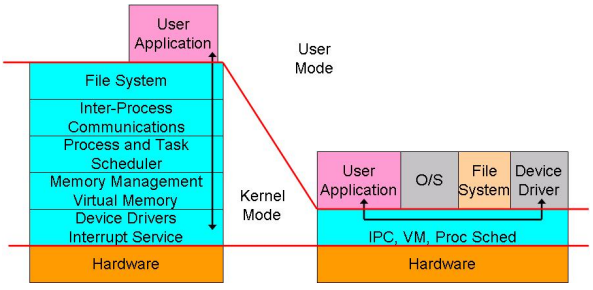

Figure 1. Microkernel operating system architecture

Figure 1 illustrates the basic concept of a

microkernel-based operating system (OS). The microkernel OS provides the core management elements required to support an operating system implementation, e.g. task scheduling, memory management, basic I/O drivers.

The underlying question is whether there are similarities in form and function between operating systems and software radio infrastructures. More importantly, if such similarities exist, can the same approach be applied to software radio infrastructure design and implementation.

4.1. Essential Management Requirements

Given the proliferation of radio management frameworks and the similarity of interfaces and functions, as illustrated previously in table 1, it would seem that the same approach use to reduce the complexity and cost of operating systems development would be applicable to software radio framework development.

The first step to realizing a microkernel Software Defined Radio (SDR) Operating Environment (OE) architecture is to identify the common, cross platform functional capabilities.

Device Control: The start and stop interfaces provide the function calls to start and stop processing of a hardware or software component of the system.

Software Deployment: Since functional capabilities are increasingly performed by software, the deployment of software (including Digital Signal Processor (DSP) and Field Programmable Gate Array (FPGA) images) across the set of processing elements in a radio must be supported.

Resource Management: Basic operations require some level of management control over the resource available in the system. In an operating system this may be memory or processor cycles. In a radio system it may these elements as well as Radio Frequency (RF) electronics, amplifiers, and modems.

Command Interface: The deployment of waveform software, the start of processing, getting device status and other operations are typically initiated via some operator command or interface. While the specific visual or tactile method used by the operator to enter the command may vary, there is nonetheless a command processor that accepts the command from an operator and transforms that into the appropriate system level command or call.

Based on the above common functional capabilities, it appears that the realization of microkernel implementation of a radio management framework would be both logical and cost effective.

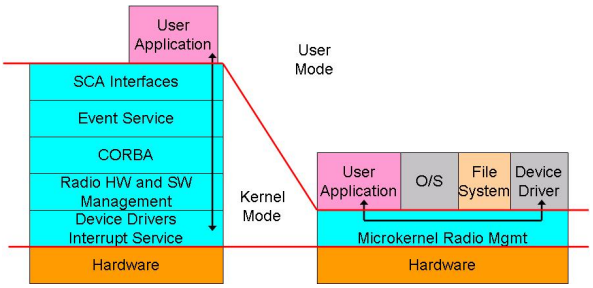

Figure 2. Microkernel SDR operating environment

Figure 2 illustrates the conceptual view of a microkernel approach for implementing an SDR core framework.

4.2. Deployment-Driven Functionality

Assuming the availability of a microkernel SDR management framework, the question then becomes how to address the different requirements of specific SDR systems. While there are some differences between the SCA, SWRADIO and STRS specifications, most of the differences are relatively minor, i.e. differences in function signatures. The fundamental purpose and behavior of the functions are essentially isomorphic.

The key drivers in radio framework implementation then appears to be:

- the key functionality required for a given radio system,

- the SWaP constraints of the radio architecture and
- impact of the deployment environment.

When analyzed objectively, the above items are the underlying drivers that promulgated the initiation of the STRS specification over use of the SCA. For example, the use of CORBA within the SCA adds overhead that was deemed to exceed the resources available on a space-deployed radio system.

The SWRADIO specification alleviated some of the issues associated with specific implementation technology through its use of a PIM to define the essential architecture and interfaces without committing to an implementation approach or technology. In fact, elements of the SWRADIO specification were incorporated into the second iteration of the STRS specification. However, the essential functional interfaces across each of the specifications remain essentially identical.

5. Summary

This paper has presented a conceptual approach to the development of software radio management infrastructure that focuses on the essential functions required to support radio management. As presented in the paper, there is a substantial intersection of functionality and interfaces between each of the three specifications. This indicates that the underlying control and management functionality is essentially the same, differing only in minor aspects.

By using a microkernel approach to developing a radio management infrastructure, it is feasible to implement a core set of management functionality and behavior that is common across each of the specifications. This implies the core functionality is similar. Thus, the driver appears to be related to the deployment constraints of the target environment.

The microkernel approach will be investigated in future work to investigate the feasibility of having a common management kernel that can be adapted to different specification through a layered interface library.

Acronyms

BIT	Built in Test
CCM	CORBA Component Model
CORBA	Common Object Request Broker Architecture
DSP	Digital Signal Processor
FPGA	Field Programmable Gate Array
GPP	General Purpose Processor
JTRS	Joint Tactical Radio System
JPO	Joint Program Office
MDA	Model Driven Architecture
MSRC	Modular Software Radio Consortium
NASA	National Aeronautics and Space Administration
OE	Operating Environment
OMG	Object Management Group
OS	operating system
PIM	Platform Independent Model
PSM	Platform Specific Model
RF	Radio Frequency
SCA	Software Communications Architecture
SDR	Software Defined Radio
STRS	Space Telecommunications Radio Systems
SWaP	Size Weight and Power
SWRADIO	PIM and PSM for Software Radio Components
XML	eXtensible Markup Language

References

[1] Object Management Group. *CORBA Component Model (CCM) Specification*, 2006.

[2] Joint Tactical Radio System (JTRS) Joint Program Office (JPO), Arlington, VA. *Software Communications Architecture (SCA) Specification*, version 2.2 edition, November 2001.

[3] Object Management Group. *PIM and PSM for Software Radio Components*, dtc/2005-09-05 edition, 2005.

[4] J. Bard and V. Kovarik. The software communications architecture. In Walter Tuttlebee, editor, *Software Defined Radio*. Wiley & Sons, 2007.

[5] J. Briones, L. Handler, S. Hall, R. Reinhart, and T. Kacpura. Case study: Using the omg swradio profile and sdr forum input for nasa's space telecommunications radio system. Technical Report NASA/TM-2009-215478, NASA, January 2009.