STRATEGIC ADAPTATION OF SCA FOR STRS

Todd Quinn[1] (ZIN Technologies/NASA GRC, Brook Park, Ohio, USA;
todd.quinn@zin-tech.com);
Thomas Kacpura[1] (ZIN Technologies/NASA GRC, Brook Park, Ohio, USA;
thomas.kacpura@zin-tech.com

ABSTRACT

The Space Telecommunication Radio System (STRS) architecture is being developed to provide a standard framework for future NASA space radios with greater degrees of interoperability and flexibility to meet new mission requirements. The space environment imposes unique operational requirements with restrictive size, weight, and power constraints that are significantly smaller than terrestrial-based military communication systems. With the harsh radiation environment of space, the computing and processing resources are typically one or two generations behind current terrestrial technologies. Despite these differences, there are elements of the SCA that can be adapted to facilitate the design and implementation of the STRS architecture.

1. INTRODUCTION

Space Telecommunication Radio Systems (STRS) is an open architecture specification based on software defined reconfigurable technologies and is being developed by NASA for its future space-radio communications and navigation systems. Software defined radios (SDRs) offer advanced operational capabilities which will result in reduced mission life cycle costs for space platforms. The objective of the open architecture for NASA space SDRs is to provide a consistent and extensible environment on which to develop, manage and operate the increasingly complex software radios used in NASA space missions. The open STRS architecture provides a framework for leveraging earlier efforts by reusing various architecture compliant system components developed previously in NASA programs.

The United States Department of Defense (DoD) has developed an open architecture for their next generation of military radio communication systems. With the participation of a large number of companies, the government has spent a considerable amount of effort, time, and expense on the development of the Software Communications Architecture (SCA). The STRS architecture and the SCA share many of the same goals; however, the constraints of space-based systems currently prevent full utilization of the SCA by NASA.

To leverage the work accomplished by the DoD, this paper examines aspects of the SCA that can be applied to facilitate the design and implementation of the STRS architecture. STRS compatibility with the SCA would allow NASA to utilize commercial development and testing tools, share waveform components which may reduce programmatic costs of maintaining an architecture. Highly effective commercial software development tools are reducing the time and cost of developing SCA compliant waveforms and platforms. STRS adoption of these commercial tools would provide a consistent set of standards and practices possibly lowering the costs of platform and waveform development.

2. SOFTWARE COMMUNICATIONS ARCHITECTURE

The DoD Joint Tactical Radio System (JTRS) program created the SCA to provide a specific framework on which to build their next generation of military radios. Mandating compliance to the SCA, JTRS is ensuring that components of military communication systems developed by various manufacturers will seamlessly operate together. This interoperability also promotes hardware/software component reuse, faster technology insertion, and expands participation of companies with new and innovative ideas. The SCA is the foundation for cost effective and flexible communication systems that can adapt and evolve over time as military operational capabilities change and expand to meet future DoD requirements.

The SCA stipulates how various hardware and software components form the structure of a radio communication system. The specification of the SCA constrains software development at component interfaces and does not limit the functional implementation within the components. Innovative intellectual property can still be protected while

[1] This work is performed under NASA contract NAS3-99155

Proceeding of the SDR 06 Technical Conference and Product Exposition. Copyright © 2006 SDR Forum. All Rights Reserved

achieving benefits of software re-use on other platform implementations of the SCA.

The SCA was developed with an objected-oriented approach and is graphically represented with the Unified Modeling Language (UML). Details and complete technical specification of the architecture can be found in the Software Communications Architecture Specification document [1].

2.1 SCA Operating Environment

Figure 1 shows the SCA separation of application software from the underlying hardware. The SCA operating environment consists of a :

- Core Framework,
- CORBA Middleware
- POSIX Real-Time Operating System

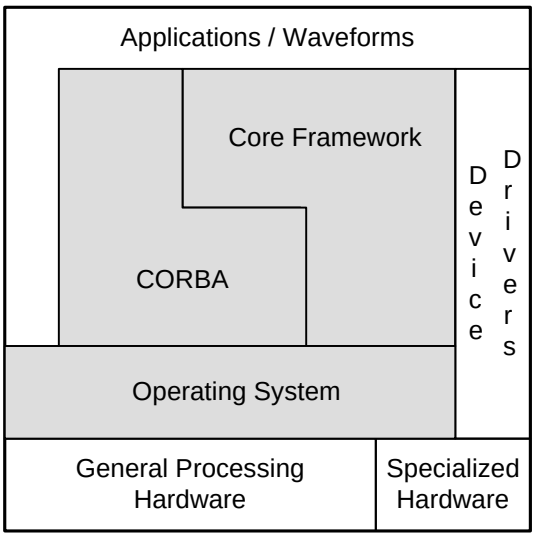

Figure 1. SCA Application / Hardware Separation

The core framework provides a set of interfaces between the application layer (SCA compliant waveform) and the operating system associated with the underlying processing hardware. These interfaces provide access to the various application building blocks, application control mechanisms, and core framework services. Core framework services are common functionality used by a majority of waveform application such as file management, logging, and timing.

Middleware is a data transport layer that provides communications among software components. CORBA (Common Object Request Broker Architecture) [2] is a multi-platform distributed communication system developed by the Object Management Group. CORBA is independent of programming languages, processing hardware, and operating systems. This contributes to easier software re-use, better software portability and increased productivity. With the use of CORBA, data transportation and remote procedure calls do not have to be created from scratch and are standardized from system to system. The software developer does not have to know about the communication details of the underlying hardware on all possible platforms and can concentrate on the higher level issues of platform and application development.

Another aspect of the SCA is the requirement for a real time operating system (RTOS). Radio systems have real time requirements; frequently data has to be moved from one point to another or converted from one format to another in a certain fixed amount of time. The SCA satisfies real time requirements by using an RTOS that provides tight control over software execution timing. The choice of an RTOS for an SCA compliant platform is up to the hardware platform provider. However to support portability, the chosen RTOS is required to be compliant with a POSIX (Portable Operating System Interface) standard.

2.2 Software Components and Interfaces in the SCA

A software component is an organized unit of program instructions that provide a small set of related functional capabilities normally called the component's behavior. The behavior of a component can only be accessed through its open public interfaces. Each instance of a software component maintains a set of private internal variables commonly referred to as the component's properties. The SCA defines a variety of components and their associated public interfaces. The rest of this section briefly describes the major components defined in the SCA that may be leveraged by STRS. Detailed descriptions of all SCA components can be found in the SCA specification.

At the application layer, an SCA compliant waveform is comprised of one or more Resource components which have specific interfaces. The Resource interfaces provide common control and configuration functions for waveform components; these interfaces are used by the SCA core framework. The waveform developer can extend Resource by adding new behaviors and interfaces to create specialized Resource components. As an example, the Device component in the SCA core framework is an extension of Resource and acts as a software proxy for actual hardware.

System control is accomplished through the core framework components defined as DomainManager, ApplicationFactory, Application, Device, and DeviceManager. The DomainManager provides control and configuration of the overall system domain. The DomainManager interfaces are grouped into three categories: human computer interaction, registration, and core framework administration. The ApplicationFactory, also part of domain management, provides an interface to request the creation of a specific type of application within

Proceeding of the SDR 06 Technical Conference and Product Exposition. Copyright © 2006 SDR Forum. All Rights Reserved

the domain. For each instantiated application, an Application component is created by the ApplicationFactory. The Application component controls, configures, and returns status of the instantiated application. A Device component is an extension of Resource and has additional behaviors for abstracting the control and functionality of physical hardware elements. A DeviceManager manages a set of logical Devices.

2.3 SCA for Space

It is important to focus on minimizing the required resources of the system (e.g. size, power, and mass) for the constrained space environment. Processors and other electronic devices used in space require radiation hardening. These components lag at least a generation or two behind the processing capabilities of their terrestrial-based equivalents. Due to slower processors and limited memory footprint, these reduced capabilities constrain the operating environments of space radios compared to radios using commercial components.

Space waveform applications requiring digital signal processing have historically been executed in specialized Application Specific Integrated Circuits (ASICs). ASICs have the lowest power requirements and greatly satisfy radiation requirements for space, however they are not reprogrammable. Reconfigurable signal processing is slowly gaining in acceptance within NASA with the availability and use of space qualified DSPs and FPGAs. Critical applications sometimes use DSPs, FPGAs, and other specialized hardware during the design process, but for deployment in space the circuit is implemented in an anti-fuse FPGA or an ASIC. The use of DSPs and FPGAs are treated as special cases in SCA 2.2. The present focus of the SCA has not addressed specialized hardware abstraction, but is on-going as JTRS continues to develop radios for various military applications.

There are other challenges that are factors for an SCA that supports space radios. The SCA compliant core framework must fit on the space qualified platform in terms of resources, footprint, and features. If the SCA is used to provide an environment where radio capabilities can be reprogrammed, the necessary core framework will take up resources that would normally be dedicated directly to signal processing. Smaller amounts of already limited resources will be left for signal processing. The SCA in space also has to address concerns with the added software complexity and the affect on system reliability.

3. SPACE TELECOMMUNICATION RADIO SYSTEMS

The STRS architecture provides the foundation for a new generation of NASA communication systems with greater system interoperability, increased hardware/software component reuse, faster technology insertion and the ability to adapt and evolve to changing to changing requirements. Figure 2 shows the STRS architecture separation of waveform application software from the underlying hardware. The basic premise of STRS architecture is that the STRS infrastructure and waveform application execute on a combination of general purpose processing hardware and specialized processing hardware.

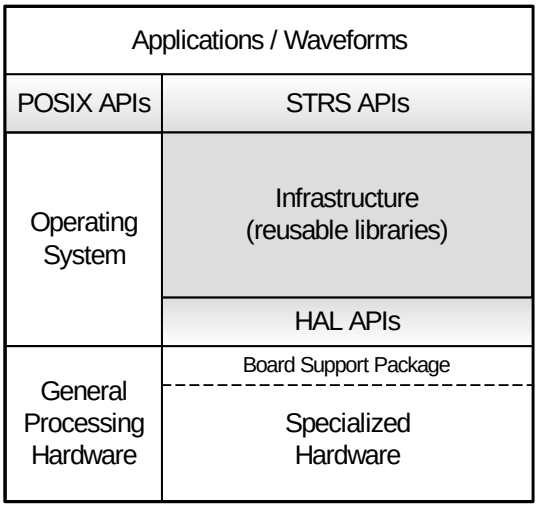

Figure 2. STRS Application Software / Hardware Separation

The STRS application programming interfaces (APIs) layer located between the waveform application and the rest of the radio system is a key concept to the STRS architecture. These APIs abstract the waveform application from the rest of the radio's operating environment and provide a set of interfaces for high portability and reuse of waveforms and their application components. The STRS APIs provides interfaces to manage the radio platform and waveforms, control logical devices and use available platform services.

The layers below the STRS APIs comprise the STRS operating environment consisting of three elements: 1) the infrastructure, 2) a real time operating system, and 3) a Hardware Abstraction Layer (HAL). The STRS infrastructure implements the STRS APIs which supports system management, device control and data transfer functions. The infrastructure interacts through the APIs with both the waveform application and the computing hardware elements. The infrastructure provides an abstracted path from the waveform application to the hardware, independent of the particular hardware elements.

STRS requires that the real time operating system for the radio platform be commercially available and also implement portions of POSIX (Portable Operating System Interface). POSIX provides industry standard functionality

Proceeding of the SDR 06 Technical Conference and Product Exposition. Copyright © 2006 SDR Forum. All Rights Reserved

across a variety operating systems to help increase the portability of applications.

The HAL is the third element of the operating environment providing interfaces for board support packages necessary to run waveform application components on specialized hardware. The HAL resources include device drivers, published APIs to access specialized hardware board support packages and documented interface definitions released by the platform developer. A complete description of the STRS architecture can be found in the STRS Architecture Description [3] and STRS Architecture Standard document [4].

3.1 JTRS and NASA Space-Based Radio Differences

There are several differences other than size, weight and power constraints, as well as reliance on specialized signal processing for NASA space-based radios compared to JTRS communication systems. NASA radios generally operate at higher frequencies and higher data rate transmissions than the current SCA compliant radios. Also extensive testing is conducted in order to assure proper operation and characterization of operation. Access during NASA missions is generally limited to remote uploads for changing the behavior of the radios. Security concerns are not as stringent, although authentication may be required. High reliability requirements stem from safety concerns, as well as from the need to have the capability to contact and control the spacecraft through the radio as the mission requires.

Design aspects that drive the STRS architecture also come from an operational perspective. The SCA is designed to support dynamic deployment and/or reconfiguration of waveform applications for a variety of hardware platforms. NASA mission operations are pre-planned in advance and limited to a specific hardware platform, and full SCA capabilities for dynamic deployment and reconfiguration on different platforms are not needed.

3.2 SCA / STRS Commonality

Despite the environmental and operational differences described above there are many aspects of the SCA that can be leveraged for the STRS architecture. There are several elements of the SCA that have commonality with the STRS architecture approach and are shared between the two architectures:

- Seeks to separate the software waveform applications from the hardware
- Requires operating environments consisting of real time POSIX compliant operating systems.
- Describes an infrastructure within the operating environment that provides system management and communication services for waveform application and platform components.
- Configuration files are used for platform description and application deployment.

The middleware specification is one difference between the two architectures at the architecture level, but can be viewed as common at the implementation level. The SCA operating environment requires CORBA middleware for component communication and STRS does not specify any particular communication mechanism within its operating environment. Since the mechanism for component communication is non-specific for STRS, the use of CORBA for implementation of STRS operating environment communications is not precluded.

4. STRS INFRASTRUCTURE AND THE SCA

Domain management, application management and device management within the STRS infrastructure could leverage the SCA for greater compatibility between the two architectures. The STRS infrastructure is composed of multiple subsystems which include Radio Control, System Management, Device Control, and a Message Center. The SCA core framework defines components that provide related functionality. Some functionality in the STRS subsystems may be differently distributed across components when compared to the SCA components. An implementation of the STRS architecture based on SCA components would help evaluate differences between the two architectures and help evolve greater compatibility.

A mapping between STRS infrastructure components and corresponding SCA components can be made. Radio Control in the STRS infrastructure provides the interface exchange between the STRS Radio and the Spacecraft Bus. All command and telemetry processing is handled by the Radio Control subsystem. Within the SCA core framework, the DomainManager component is responsible for interfacing to the various subsystems interacting with the SCA software defined radio. The STRS infrastructure System Management subsystem controls the instantiation and teardown of applications and services within the STRS Radio. This activity includes keeping track of what resources in the radio are being used and when new functionality can be installed. The SCA core framework has ApplicationFactory and Application components that provide similar functionality. The Device component in the SCA core framework is comparable to the Device Control capabilities required in the STRS infrastructure.

The Message Center subsystem manages the inter-process messaging and queue allocations for the STRS Radio. It has the responsibility of processing requests made by applications and also controls communications between subsystems in STRS infrastructure. The SCA uses CORBA for its inter-component communications. CORBA adds a

Proceeding of the SDR 06 Technical Conference and Product Exposition. Copyright © 2006 SDR Forum. All Rights Reserved

layer of complexity, however CORBA is well implemented and tested, and the added complexity is out weighed by the benefits derived by simplifying distributed application development while at the same time providing a standard for consistent and interoperable systems. If the implementation of a STRS infrastructure included CORBA, then compatibility with SCA would be increased. Since the STRS architecture views CORBA as an implementation detail, an implementation of the STRS architecture could be built with current high speed, small footprint CORBA products to investigate resource, performance and reliability issues associated with the middleware.

5. STRS WAVEFORM DEPLOYMENT AND THE SCA

The SCA uses a set of files called the Domain Profile to describe the components and the connections between components. These files, in XML format, describe the identity, capabilities, properties, and inter-dependencies of the hardware devices and software components that make up the system. Appendix D of the SCA specification [1] has detail descriptions and format specifications of the various XML files that comprise a domain profile.

It is the parsing of the XML data and its interpretation by the SCA core framework that loads software components and creates the connections between them. The STRS infrastructure could use the XML domain profile formats to describe its radio platforms and waveforms. A waveform can be described with SCA domain profile XML semantics and syntax but deployed by a STRS infrastructure.

Under the SCA, an XML parser dynamically retrieves information from the domain profile as the waveform is loaded onto the software radio platform. For STRS, since the platform and device resources are known before the time of waveform deployment, the XML files can be parsed ahead of time and pre-processed to save resources by not having an XML parser as part of the infrastructure. There are two options for pre-processing the XML domain profile for the STRS architecture:

1. The domain profile pre-processor could generate actual code to deploy the waveform onto the specific hardware.
2. Convert the XML domain profile into a static binary format that would be input to a STRS waveform deployment routine that loads the waveform.

The first option has the benefit of deploying the waveform as fast as possible, since the deployment code is specific to the waveform on the specific platform. The disadvantage of this approach would be that the deployment code would have to be regenerated for all waveforms that move to a different platform. The second option provides a more flexible approach, such that the XML files are translated into a standard binary format used by all waveforms and platforms. If the platform changes for a group of waveforms, only a new deployment routine has to be created for each new platform and nothing has to be generated for each specific waveform.

Following either option, the use of the SCA domain profile for STRS allows the use of existing SCA testing software to verify waveform and platform configurations. Commercial tools for SCA development can also be used to automatically generate accurate STRS configurations files increasing productivity and reliability while reducing waveform development time. Keeping the domain profile common between the SCA and the STRS architecture will help to continue the focus on closing the gaps between SCA and STRS as each architecture evolves.

6. CONCLUSION

NASA is in the process of developing STRS as an open architecture for software defined radios in space. The STRS objective is to provide a consistent and extensible environment on which to construct and operate future NASA space communication systems. STRS shares many of the goals and attributes of the SCA already developed by the JTRS program. However, requirements and constraints associated with space-based systems prevent NASA from utilizing the current SCA specification, primarily due to the large footprint and resources, as well as the complexity due to dynamic deployment capability of SCA waveform applications. For a NASA SDR architecture to be sustainable, it must carefully consider the unique constraints and needs of the space environment.

There is commonality in a number of areas between the two architectures that NASA could potentially leverage from the considerable assets derived from the military and commercial application of the SCA. STRS compatibility with the SCA would allow NASA to utilize tools, share waveform components and reduce programmatic costs of maintaining a separate architecture. Today distinct differences between the STRS architecture and the SCA are related to the differences in available technologies and operational requirements. However, as technologies for the space environment evolve, they should allow the STRS architecture to incorporate more features and capabilities of the SCA.

7. REFERENCES

[1] Modular Software-programmable Radio Consortium, *Software Communications Architecture Specification*, MSRC-5000SCA V2.2, Joint Tactical Radio System (JTRS) Joint Program Office, November 17, 2001

Proceeding of the SDR 06 Technical Conference and Product Exposition. Copyright © 2006 SDR Forum. All Rights Reserved

[2] Object Management Group, *Common Object Request Broker Architecture: Core Specification*, Version 3.0.3, March 12, 2004

[3] Space Operations Mission Directorate, *Space Telecommunications Radio System STRS Open Architecture Description*, Phase 1 Architecture (Revision 1.0), NASA Headquarter, Washington DC 20546-0001, April 2006

[4] Space Operations Mission Directorate, *Space Telecommunications Radio System STRS Open Architecture Standard*, Phase 1 Architecture (Revision 1.0), NASA Headquarter, Washington DC 20546-0001, April 2006

Proceeding of the SDR 06 Technical Conference and Product Exposition. Copyright © 2006 SDR Forum. All Rights Reserved