MILCOM 2004 TUTORIAL

Introduction to the JTRS Software Communications Architecture (SCA)

Steve Bernier
Communications Research Center Canada (CRC)
Government of Canada
steve.bernier@crc.ca
Outline

• CRC’s and the Software Communications Architecture
• SCA Overview
• SCA Components
• Application Deployment
• Application Development Cycle
• Summary
Outline

- CRC’s and the Software Communications Architecture
  - SCA Overview
  - SCA Components
  - Application Deployment
  - Application Development Cycle
  - Summary
CRC’s SCA Background

• Designed a proprietary SDR architecture (1998)
• Joined the SDR Forum (1999)
  – Involved in design discussions regarding the SRA and the SCA
    • Socialized the concept of CCM Ports to enable true
      modularity of software components
• Implemented a proof of concept SCA SDR for the Canadian
  Department of National Defence (2000)
  – FM Line of sight application running on a Daytona board (dual
    TI C6201 DSPs) from Spectrum Signal Processing
  – Implemented a SCA v0.1 Core Framework
  – Pioneered the use ‘SCA adapters’ with a dynamic loader for run-
    time partial reloading of DSPs
CRC’s SCA Background

• Released an open source implementation of the SCA (2002)
  – Project called SCA Reference Implementation (SCARI)
  – Java™ open source implementation of the SCA version 2.1
  – Implemented with the participation of Defence R&D Canada - Ottawa
  – Sponsored by the Software Defined Radio Forum
  – Peer reviewed by the SDR Forum oversight committee
• MITRE JPO staff, US AFRL, L3-Communications, Mercury Computer Systems, Sun Microsystems, Space Coast Systems
CRC’s SCA Background

• SCARI Project
  – SCA Core Framework components
  – Simple voice transformation application
  – 60,000 lines of code, 300 pages of documentation
  – SCA Component Unit Test bed: Over 55,000 lines of code
  – More than 7000 downloads from worldwide organizations
  – Over 50 000 web hits since January 2001
  – CRC submitted 21 technical change proposals to JTRS / JPO in reference to SCA version 2.2
CRC’s SCA Background

- Demonstrated a Digital Audio Broadcast (DAB™) application
  - November 2002, SDRF General Meeting, San Diego
  - First demonstration of a commercial SCA SDR application
  - Demonstrator implemented using:
    - SCARI
    - CRC software implementation of a DAB receiver
    - CRC’s signal processing algorithms
    - CRC’s ADC messaging middleware
CRC’s current activities

• SCARI2 Project
  – CRC has been selected by the Software Defined Radio Forum to develop a JTeL certified SCAv2.2 Core Framework
  – In addition, a one-channel push-to-talk application will be released with source code
  – CRC is responsible for the development of the Core Framework (Java™) and the application (Java/JNI™ and C)
  – CRC’s partners :
    • JTRS JPO, JTeL, ISR Technologies, École des Technologies Supérieures, Rohde and Schwarz, Mercury Computer Systems, NASA Glenn Research Centre
  – Ready to be reviewed by SDR Forum
CRC’s current activities

- CRC offers a commercial SCA Core Framework for embedded platforms
  - C++ SCAv2.2 Core Framework
  - Development GUI tools for XML domain profile files
  - Development GUI tools for C++ source code generation
  - Radio control GUI tools
  - SCA Training course (3 days)
  - For more info visit our website at http://www.crc.ca/rars
Outline

• CRC’s and the Software Communications Architecture
  • SCA Overview
    • SCA Components
    • Application Deployment
    • Application Development Cycle
    • Summary
SCA Overview

• The SCA provides:
  – Interface specifications
  – Application program interfaces (APIs)
  – Behavioral specifications
  – Rules

• Is like an “operating system” for radios
  – Executes applications
  – Controls hardware devices

• Is an implementation-independent framework which ensures interoperability of SCA products
• The SCA specification consists of the following documents:
  – Software Communications Architecture (Core Framework)
  – Appendix A – Glossary
  – Appendix B – Application Environment Profile
  – Appendix C – Core Framework IDL
  – Appendix D – Domain Profile
  – API Supplement
  – Security Supplement
  – Rational Document

• Main document and appendixes contain several hundred pages of documentation and over a thousand requirements
SCA Overview

• **Software Communications Architecture Specification**
  – API and behavior for each SCA Core Framework Component
  – Contains 679 requirements

• **Appendix B. SCA Application Environment Profile**
  – SCA requirements for an operating system which is a subset of the POSIX standards

• **Appendix C. Core Framework IDL**
  – Specifies the public API for each software component

• **Appendix D. Domain Profile**
  – Specification of the descriptor files for software components and assemblies (i.e. components spec-sheet)
Introduction to the JTRS SCA,
Steve Bernier

• Attachment 1 to Appendix D. Document Type Definitions
  – XML DTD files describing the format of the domain profile files

• Attachment 2 to Appendix D. Common Properties Definition
  – Description of the SCA standard capability properties a
    Device can advertise

• API Supplement
  – Main document and 8 appendixes (A through G)
  – Describes APIs for Physical, MAC, and LLC layers

• Security Supplement
  – Contains over 270 requirements
SCA Software Structure

- Physical Component
- Non-CORBA Physical Component
- MAC Component
- MAC Adapter
- Link, Network Components
- Security Adapter
- Security Components
- Link, Network Components
- I/O Adapter
- I/O Components

Core Framework IDL
("Logical Software Bus" via CORBA)

- CORBA ORB & Services (Middleware)
- CF Services & Applications
- Operating System
- Network Stacks & Serial Interface Services
- Board Support Package (Bus Layer)

Black Hardware Bus

Corba ORB & Services (Middleware)
CF Services & Applications
Operating System
Network Stacks & Serial Interface Services
Board Support Package (Bus Layer)
Red Hardware Bus
Hardware Platform

• RF Hardware
  – Multi-Band Antenna
  – Wideband RF front end

• Digital Hardware
  – Converter Technology
    • Analog to Digital and Digital to Analog
  – Fast processors
    • GPP, DSP, FPGA
  – High speed data bus
    • PCI Express, VME, RapidIO, InfiniBand, StarFabric
Software Platform

• SCA-AEP POSIX Operating System
  – Ex: Integrity, QNX, Linux, LynxOS, VxWorks

• CORBA Middleware
  – Provides a platform independent communications framework for software components
  – Ex: ORBexpress, TAO, eORB

• SCA Devices
  – Provide access to hardware devices in a portable way
  – Ex: ExecutableDevice, ModemDevice, AudioDevice, SecurityDevice
Software Platform

- **SCA Waveform Applications**
  - Perform signal processing
  - Ex: SINCGARS, VHF-FM, UHF DAMA, APCO25, TETRA

- **SCA Core Framework**
  - Performs deployment and configuration of software components
  - Uses application-layer interfaces to abstract software and hardware components for application developers
SCA Core Framework

• Enhances portability of software providing a component-based development (CBD) paradigm
  – Based on the OMG CCM, a commercial CBD architecture
  – SCA could be used to deploy any kind of application, not only radio waveform applications
    • Post-processing of radar images
    • Medical imagery
    • Casino slot-machines

• The SCA CF is a Component-Based Development Architecture for embedded systems
  – Is device centric as opposed to the high-level application centric OMG CCM
Component-Based Development

- **Definition:** The creation, integration, and re-use of components of program code, each of which has a common interface for use by multiple systems.

- Using CBD, an application is ‘assembled’ using individual software components much like a board is populated with hardware components.

- Promotes the commercial of the shelf (COTS) culture for software.
Component-Based Development

• **Software Component**
  – **Definition**: is a small, reusable module of executable code that performs a well-defined function. It is designed, implemented, and tested as a unit prior to integration into an application
  – Behaves the same way no matter where it runs
  – Exposes same interfaces for other components
  – Uses same interfaces from other components

• **A software component is treated as a “black box”**
  – Application designer is concerned with what a component does, but not how it does it
  – Creating an application requires component assembly knowledge
SCA Core Framework

• Defines several types of software components that must be implemented

• With the SCA, the “black box” component information is stored in a database called the domain profile
  – Describes the public API of a component
    • Number of ports, port types, configuration properties, etc
  – Describes the capability and capacity requirements of a component

• Performs the deployment of Applications (i.e. assembly of components)
Outline

- CRC’s and the Software Communications Architecture
- SCA Overview
- SCA Components
  - Application Deployment
  - Application Development Cycle
  - Summary
SCA Radio Layers

- **SCA Core Framework** (radio management)
- **CORBA Middleware**
- **SCA Devices**
- **POSIX SCA-AEP Operating System**
- **Digital Hardware**
- **RF Hardware**
- **SCA Applications**
- **POSIX APIs**
SCA Components

- Software components are partitioned into three groups: Radio management, Devices, Applications

  1. Radio management
     - Used to install/uninstall/deploy/configure applications, health monitoring, introspection, etc
     - DomainManager, ApplicationFactory, Application

  2. Devices
     - DeviceManager
       - Used to control radio nodes
     - Device, LoadableDevice, ExecutableDevice, AggregateDevice and more
       - Provide access to hardware components
       - Devices can be started, stopped, tested, configured, etc
SCA Components

• Software components are partitioned into three groups:
  
  – 3. Applications
    • Deployable software implementation of communication standards such as: FM LoS, EPLRS, Link 16, etc
    • An application is composed of Resources
    • Applications can be started, stopped, tested, configured, etc

• An SCA radio contains
  – Exactly one DomainManager
  – One-to-many DeviceManagers
  – One-to many Devices and services for each DeviceManager
  – One ApplicationFactory for each type of application installed
  – One Application for each deployed application
Concept of an SCA application

• An application is composed of a number of software components (called Resources) that perform signal processing
Concept of an SCA application

- Simplified block diagram of the CRC’s Digital Audio Broadcast (DAB™) application
Concept of an SCA application

- A **Resource** is much like a hardware component
  - It has a certain combination of input, output, and control ports that need to be connected to other components
  - To use a Resource in an application, it must be deployed onto a **Device**
  - A Resource has requirements a Device must meet
    - Capacity requirements: MIPS, kilo bytes of memory, etc.
    - Capability requirements: OS, Processor, etc.

- The behavior of a Resource can be altered by changing the value of its configuration properties
  - Ex: code rate
Concept of an SCA application

- Since an application is composed of many *Resources*, by aggregation, it contains all the ports and configuration properties of its Resources.

- A *Resource* is a software component, it knows nothing about other *Resources* or about the application it’s being used in.

- Therefore, control and configuration of an application needs to be coordinated.
Concept of an SCA application

• The Assembly Controller *Resource* represents the central component through which application control and configuration is performed
  – Configuring the Assembly Controller might cause the configuration of many Resources
  – Stopping an application may only require stopping one *Resource*

• The SCA Core Framework provides encapsulation protection for the application’s *Resources*
  – CF must only provide access to the Assembly Controller *Resource* of an application

• The Assembly Controller is developed by the application designer and bundled with the rest of the *Resources*
Concept of an SCA application

Assembly Controller

Local Property

Aggregated Property

Proxy Property

Resource1

Resource2

Resource3

footer

Introduction to the JTRS SCA,
Steve Bernier

33
Concept of an SCA application

- An application can be viewed as a single *Resource*:
  - It has a certain combination of input, output, and control ports that need to be connected to other components
  - Its behavior can be altered through configuration properties
Concept of an SCA application

• An application can be installed in a radio from a remote location

• Installation process includes copying all the application files:
  – Resource implementations
  – Assembly descriptor files (domain profile database)
  – Component descriptor files (domain profile database)

• To be installed, an application must be validated by the DomainManager
Concept of an SCA application

• Before an application can execute, it must be deployed

• Deployment involves sending all the Resources implementations to Devices
  – Target Devices are chosen based on:
    • Capability (ex: os, processor) and capacity (ex: mips, memory, channels) requirements of the Resources
    • Advertised capabilities and capacities of Devices
SCA Radio Layers

- SCA Core Framework (radio management)
- CORBA Middleware
- SCA Devices
- POSIX SCA-AEP Operating System
- Digital Hardware
- RF Hardware
- SCA Applications
- POSIX APIs

Software Platform

Hardware Platform
1. Radio Management Components

- **DomainManager** is the central component for radio management
  - Accepts registration of *DeviceManagers*, *Devices*, and services
  - Performs interconnections between node components when specified by a *DeviceManager*
  - Responsible for the installation of applications
  - Offers introspection services
  - Used by UI to control/monitor the radio
1. Radio Management Components

- **ApplicationFactory** is responsible for instantiating specific types of applications
  - Deploys an application
  - Creates an *Application* object for each deployed application
  - *Application* object is used as a proxy to the deployed application
1. Radio Management Components

- An **Application** component is used to:
  - Start/stop the signal processing (delegates to Assembly Controller)
  - Change the behavior of the application through configuration (delegates to Assembly Controller)
  - Release the application which terminates components and frees acquired capacity
  - Get information for each **Resource** instance created
    - Which **Device** has been used to deploy each **Resource** object
    - Process ID for each **Resource** object
    - Etc.
2. Device Components

• A *DeviceManager* is responsible for a single radio node
  – Launch/shutdown node components: *Devices* and services
  – Registers to a *DomainManager*, making its node components known to the radio where they become available to applications

• A Radio Node
  – Provides access to a set of collocated *Devices*
  – For example, a PowerPC board in a Compact PCI Chassis can be considered a radio node in a SDR platform
  – In fact, any device capable of running some CORBA enabled code upon power-up can be considered a node
2. Device Components

- A better name for *DeviceManager* would be *NodeManager*
  - The term *DeviceManager* is in use since SCA v0.1 even though its behavior and responsibilities have changed drastically
2. Device Components

- **Devices provide access to hardware devices**
  - Ex: ExecutableDevice could give access to a DSP
  - Ex: LoadableDevice could give access to an FGPA

- **Implement ITU X.731 state management behavior**
  - Usage state (capacity) to control the deployment of software

- **SCA Devices and services are launched and shutdown by a DeviceManager**
2. Device Components

• SCA *Devices* are used for the deployment of software components:
  – Application *Resources*
  – *Devices*
  – services

• A service (ex: Timer) *cannot* be a target for software deployment

• SCA *Devices* and services must register to a *DeviceManager*, making them known to the radio domain in order to be used by applications
SCA Mapping: Solution 1
SCA Mapping: Solution 2

SBC

- Device Manager 1
- Executable Device
- GPP Device
- Audio Device
- A/D Device

DSP Board

- GPP

Audio Board

FPGA

A/D
Outline

• CRC’s and the Software Communications Architecture
• SCA Overview
• SCA Components
  • Application Deployment
• Application Development Cycle
• Summary
Deployment of an Application

GPP 1
- GUI

GPP 2
- Domain Profile
  - Domain Manager
  - Application Factory
  - Application

Software Bus (CORBA ORB)

GPP 3
- Device Manager 1
- Log 1
- Audio Device 1
- Executable Device 1
- Assembly Controller
- Resource 1

GPP 4
- Executable Device 2
- Resource 3
- Resource 2
- Log 2
- Audio Device 2
- Device Manager 2
Deployment of an Application
Domain Profile

- Is a database of files containing meta-data describing the different software components of an SCA radio

- Is composed of 7 types of files that can be grouped in two categories:
  - Component Descriptors: SPD, PRF, SCD, DMD, DPD
  - Assembly Descriptors: DCD, SAD
Component Descriptors

- Describe the properties (capabilities, capacities, configuration, etc) of a software component

- Describe the *Devices, Services* of a SCA SDR platform

- Describe the *Resources* of an SCA Application

- Used by the Radio Management to perform software deployment of components
Component Descriptors

- Each component may be described using up to five different descriptors:
  - Software Package Descriptor (SPD)
  - Property Descriptor (PRF)
  - Software Component Descriptor (SCD)
  - DomainManager Configuration Descriptor (DMD)
  - Device Package Descriptor (DPD)
Assembly Descriptors

• The SCA performs deployment of groups of components called assemblies
  – SCA supports two kinds of assemblies: node assemblies and application assemblies

• Specify the assembly ‘blue print’
  – Identify components of an assembly
  – Identify number of instantiations for each component
  – Indicate if components must be co-localized
  – May specify where components must be deployed
  – Identify how instantiations must be connected with each other
Assembly Descriptors

- Assembly block diagram:
Assembly Descriptors

• The two assembly descriptors are named:
  – Device Configuration Descriptor (DCD)
    • Node assembly descriptor
  – Software Assembly Descriptor (SAD)
    • Application assembly descriptor
Device Configuration Descriptor

- Provides a description of the components that are initially started by the `DeviceManager` for a node
- Identifies the `DomainManager`
- Describes connections between node components
- Specifies number of instances
- Specifies which component should be deployed where
- Specifies aggregation relationship between `Devices`
Device Configuration Descriptor

- Device Configuration Descriptor (DCD)
- Components
- Instantiations
- Connections

- SPD: DeviceManager
- SPD: AudioDevice
- SPD: ExecutableDevice
- SPD: FPGADevice

Connections:
- FD
- DM
- ED
- AD 1
- AD 2
- LS
Software Assembly Descriptor

- Identifies each component of an application
- Specifies how many instantiations of each component must be created
- Specifies any co-location restrictions for groups of instantiations
- Specifies which component is the assembly controller
- Specifies how the connections between component instantiations must be established
SAD Concepts

Software Assembly Descriptor (SAD)

Components

SPD
- R1

SPD
- R2

SPD
- R3

SPD
- R4

Instantiations

Connections

R1

R2

R2

R3

R4

R1

R2

R3

R4

SAD Concepts

Software Assembly Descriptor (SAD)

Components

SPD
- R1

SPD
- R2

SPD
- R3

SPD
- R4

Instantiations

Connections

R1

R2

R2

R3

R4

R1

R2

R3

R4

footer

Introduction to the JTRS SCA, Steve Bernier
Outline

• CRC’s and the Software Communications Architecture
• SCA Overview
• SCA Components
• Application Deployment
  • Application Development Cycle
• Summary
Application Development Life-Cycle

Application Development

1. Create Signal Processing Code
2. Create SCA Resource
3. Create SCA Application

Application Deployment and Test

4. Install/Uninstall Application on SCA Platform
5. Deploy and Test Application
1. Create Signal Processing Code

- Developer creates a software to perform signal processing (ex: FM_Modulator)

- Software must be implemented in a language supported by a compiler for the target processor
  - Can be developed tools like MATHLAB®

- Software can be compiled for a number of different target platforms
  - Proper implementation will automatically be chosen by the Core Framework when deployed
Application Development Life-Cycle

1. Create Signal Processing Code
2. Create SCA Resource
3. Create SCA Application

Application Deployment and Test

4. Install/Uninstall Application on SCA Platform
5. Deploy and Test Application
2. Create SCA Resource

- Every **Resource** must be CORBA-enabled and implement standard interfaces (specified in CORBA IDL):
  - PortSupplier, LifeCycle, PropertySet, TestableObject, and Resource

- A **Resource** can be a proxy to non-CORBA native code (DSP code)

- Each component must be described with meta-data used for deployment (domain profile database)
2. Create SCA Resource

- SCA IDL Interfaces
  - Glue Code
  - Signal Processing Code

- Resource implementation
  - Generated from CORBA IDL
  - CORBA specific code
  - Standard signal processing code
2. Create SCA Resource

• The meta-data for SCA components must be provided using 3 types of SCA XML files:
  1. Software Package Descriptor (SPD)
     • Describes component implementations
     • Describes capacity and capability requirements
  2. Software Component Descriptor (SCD)
     • Describes CORBA interfaces
     • Describes ports for interconnections
  3. Property Descriptor (PRF)
     • Describes execution parameters
     • Describes configuration properties
2. Create SCA Resource

**SPD**
- Descriptor
- SPD Level Property File
- Implementation 1
  - Impl Level PRF
- Implementation 2
  - ...
- Implementation n

**SCD**
- API
- Port

**PRF**
- Configuration Property
- Execution Parameter
- Execution Parameter

---

*Introduction to the JTRS SCA, Steve Bernier*
Application Development Life-Cycle

Application Development

1. Create Signal Processing Code
2. Create SCA Resource
3. Create SCA Application

Application Deployment and Test

4. Install/Uninstall Application on SCA Platform
5. Deploy and Test Application
3. Create SCA Application

- An SCA Application is composed of a number of Resources exchanging data using Ports.
3. Create SCA Application

- For an application to be deployed/executed on an SCA radio, it must be described using meta-data
  - Software Assembly Descriptor (SAD)
    - Identifies each component of an application
    - Specifies how many instantiations of each component must be created
    - Specifies any co-location restrictions for groups of instantiations
    - Specifies which component is the assembly controller
    - Specifies how the connections between component instantiations must be established
3. Create SCA Application

Software Assembly Descriptor (SAD)

Components

Instantiations

Connections
3. Create SCA Application

- To be installed, an application can be packaged into a single file for easy transportation
  - Contains domain profile descriptors for each component and the assembly
  - Contains binary code for *Resources* implementations and libraries
Application Development Life-Cycle

Application Development

1. Create Signal Processing Code
2. Create SCA Resource
3. Create SCA Application

Application Deployment and Test

4. Install/Uninstall Application on SCA Platform
5. Deploy and Test Application
4. Install/Uninstall Application on SCA Platform

• Installing an SCA Application involves the following steps:
  – Copying the following Application files on the SCA radio:
    • One Software Assembly Descriptor (SAD)
    • n Software Package Descriptor (SPD)
    • m Software Component Descriptor (SCD)
    • k Property Files (PRF)
    • x binary files (compiled code and shared libraries)
  – Finalizing the application installation by informing the Radio Management
Application Development Life-Cycle

Application Development

1. Create Signal Processing Code
2. Create SCA Resource
3. Create SCA Application

Application Deployment and Test

4. Install/Uninstall Application on SCA Platform
5. Deploy and Test Application
5. Deploy and Test an SCA Application

Deployment of an application consists in:
- Reading the application assembly meta-data (i.e. SAD)
  - Identifying components to be deployed
- Selecting Devices of the SCA radio for each application component
- Making capacity reservations against selected devices based on component requirements
- Deploying code files onto selected devices
- Initializing and configuring deployed components
- Establishing connections between deployed components
Application Development Life-Cycle

- Tools can be used to facilitate each step of the development life-cycle
  - SCA-CORBA code can be generated for specific Resources
  - Meta-data can be generated via GUIs
  - Package an application
  - SCA Core Framework can be controlled via GUIs
    - install/uninstall, deploy/shutdown, start/stop/control an application
    - Radio introspection: log messages, view deployed components, etc
Outline

• CRC’s and the Software Communications Architecture
• SCA Overview
• SCA Components
• Application Deployment
• Application Development Cycle

• Summary
Summary

• The SCA Specification contains several hundred pages of documentation and over a thousand requirements

• The SCA defines a set of software components that can be implemented by different organizations
  – Enhances software interoperability
  – Promotes the COTS culture

• The SCA Core Framework is a Component-Based Development architecture for embedded systems
  – Can be used for more than radio applications
Links

• [www.crc.ca/rars](http://www.crc.ca/rars)
  – The CRC SCA web site

• [www.sdrforum.org](http://www.sdrforum.org)
  – The Software Defined Radio Forum web site
  – Technical Conference and Annual meeting: Nov 15-18 - Phoenix

• SCA on a chip: Xilinx and Intelligent Software Radio (ISR)
  – Partial reconfiguration of a FPGA
  – Do a search on “Xilinx ISR partial reconfiguration”