Communications Research Centre Canada
An Agency of Industry Canada

Centre de recherches sur les communications Canada
Un organisme d'Industrie Canada

Taking the SCA to New Frontiers

Steve Bernier & Claude Bélisle

Outline

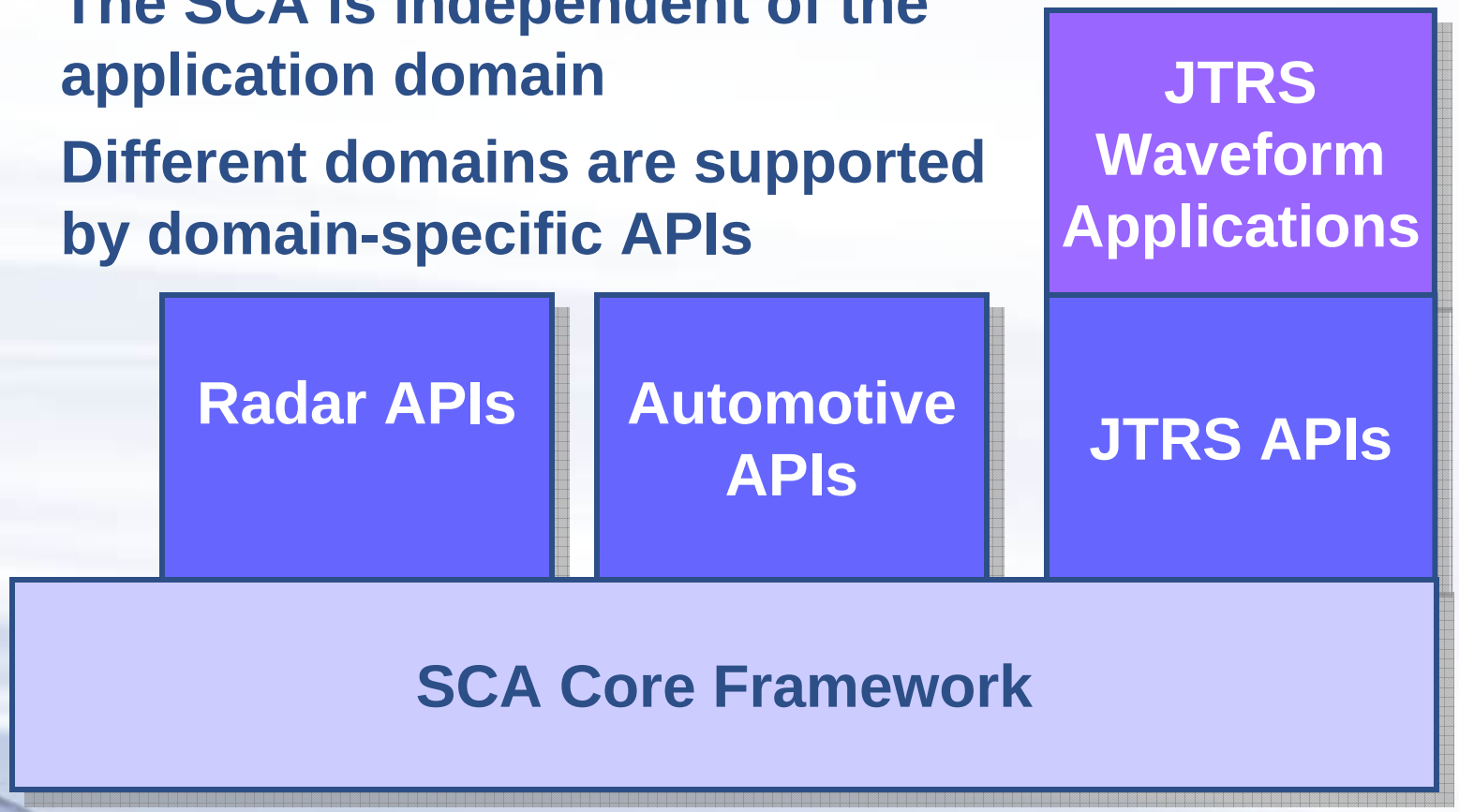
1. **Is the SCA what you think it is?**
2. **SCA Myths and Realities**
3. **Future SCA core frameworks**

SCA Overview

- **The SCA was developed to assist in the development of SDR for the Joint Tactical Radio System (JTRS). As such, the SCA has been structured to:**
  - Provide for portability of applications between different SCA platforms
  - Leverage commercials standards to reduce development costs
  - Reduce software development time with the ability to reuse design modules
  - Build on evolving commercial frameworks and architectures
- **The SCA is not a system specification!**
  - Provides an implementation-independent set of rules that constrain the design of systems to achieve the above objectives

Only for Military ?

- The SCA is independent of the application domain
- Different domains are supported by domain-specific APIs



What is so unique about the SCA?

- **Platform independent**
  - Supports any operating system*, processor, and file system
- **Scalable distributed system**
  - Supports single processor applications the same way it supports multi-processor applications
- **Designed to meet commercial as well as military application requirements**

* OS must meet a subset of POSIX APIs

What is so unique about the SCA?

- **The availability of SCA COTS solutions**
  - Operating systems
  - Object Request Brokers
  - SCA Core Frameworks
  - Reference platforms

- **The most unique attribute of the SCA is that it's actually a *Component Based Development architecture !***

* OS must meet a subset of POSIX APIs

Component Based Development

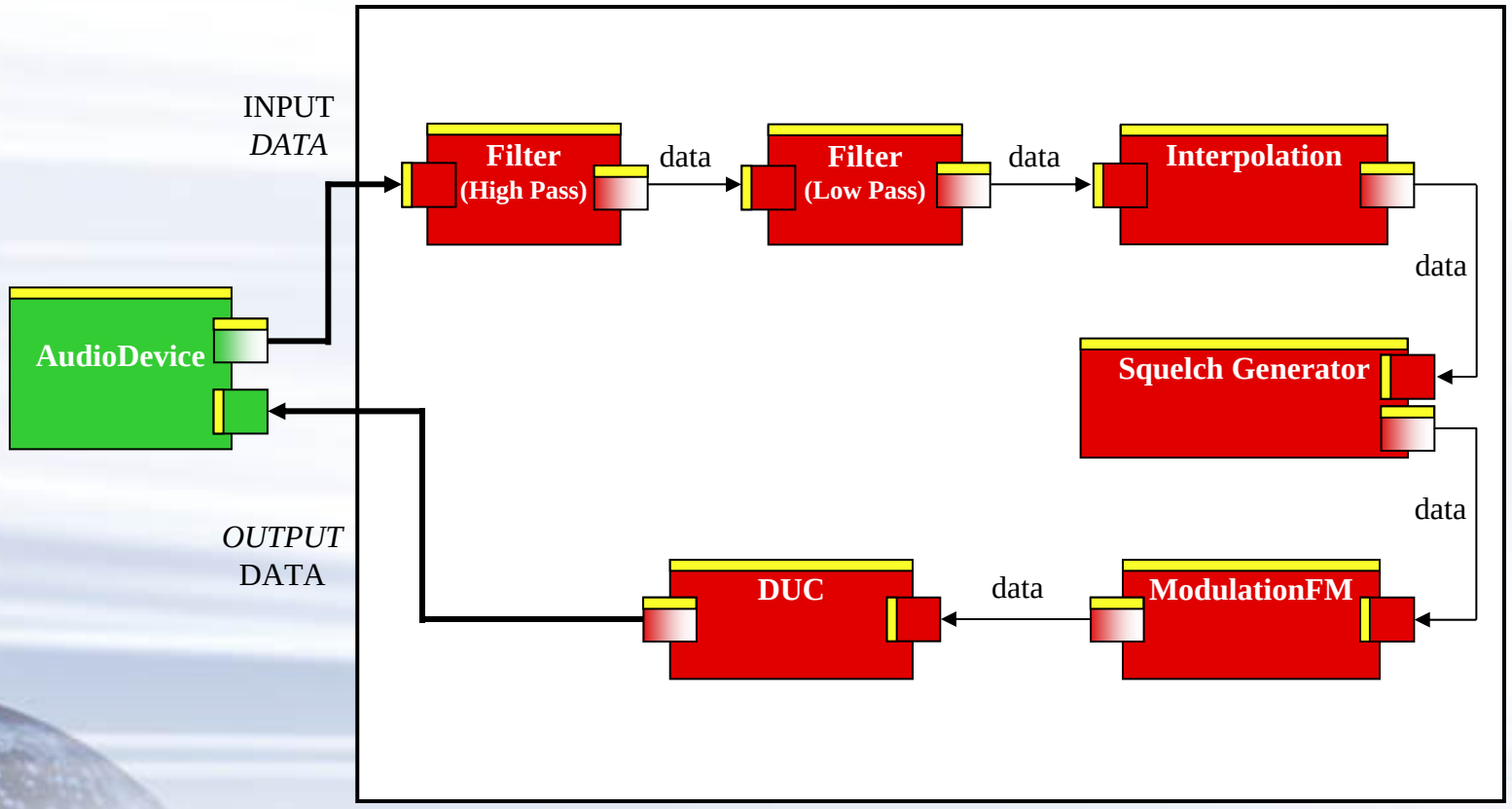
- **What is Component Based Development (CBD) ?**
  - **Definition:** an architecture which allows the creation, integration, and re-use of components of program code
  - CBD is a new development paradigm where the smallest unit of software is a **component**
  - With CBD, an application is 'assembled' using **software components** much like a board is populated with hardware components

Component Based Development

- **Software Component**
  - 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

SDR Components

- **Example of a component assembly**
  - FM modulation application



Other CBDs

- **How is the SCA different as a CBD ?**
  - As opposed to **EJB**, the SCA supports native components
  - As opposed to **.Net**, the SCA is platform-independent
  - As opposed to **CCM**, the SCA is device-centric

Outline

1. Is the SCA what you think it is?
2. SCA Myths and Realities
3. Future SCA core frameworks

Myths and Realities

- **Myth #1: The SCA is only for military radios**
  - While its true the SCA specification was developed for the US DoD JTRS program, the reality is the core framework specification contains no military features at all !
- **Myth #2: The SCA is for building Software Defined Radios**
  - None of the core framework APIs are radio specific !
  - An SCA platform can host any kind of application

Myths and Realities

- **Myth #3: The SCA only supports general purpose processors (GPPs)**
  - The lack of standardized techniques for supporting code running on DSP and FGPA lead to portability nightmare
  - The HAL-C (SCA 3.0) was created to offer a first solution to this problem
    - Lead to several change proposals some of which are implemented and work
  - ORB vendors now offer CORBA support for FPGAs and DSPs

Myths and Realities

- **Myth #4: SCA radios take for ever to boot**
  - Early SCA radio prototypes took up to 15 minutes to boot
    - The keyword to remember is "**prototypes**"
  - My cell phone takes 30 seconds to boot…and it can't be reprogrammed!
  - Booting software takes time, but let's not exaggerate!
  - Over the years, those problems have been solved with better Core Frameworks and clever engineering

Myths and Realities

- **Myth #5: SCA applications are too slow because of CORBA**
  - The SCA require inter-process communications (IPC) to allow components to interact
    - Components are developed as black boxes and have no prior knowledge of other components
  - The SCA mandates the use of CORBA as the primary form of communications between software components
    - CORBA is scalable and provides a single model for component communications
    - CORBA is COTS

Myths and Realities

- **Myth #5: SCA applications are too slow because of CORBA (cont'd)**
  - CORBA can be used with several IPC mechanisms (pluggable transports)
    - Shared memory, VME bus, Etc.
    - Default is TCP/IP
  - With a Real-time ORB, switching the IPC mechanism is done by simply changing a configuration switch
    - CORBA applications don't even need to be recompiled
  - Unfortunately, most SCA platforms still use TCP/IP as their transport
    - TCP/IP is slow!

Myths and Realities

- **Myth #5: SCA applications are too slow because of CORBA (cont'd)**
  - ISR Technologies claims to have reduced CORBA transactions latency from 300 μsec to 10 μsec by using the INTEGRITY INTCONN IPC instead of TCP/IP
    - *INTEGRITY* (Green Hills Software)
    - *ORBexpress* (Objective Interface Systems)
    - *SCARI++* Core Framework
  - Selex Communications claims to be able to reach a throughput of 56 Mbytes/s using the VME Bus as an IPC (packets of 200 octets)
    - *VxWorks* (Wind River)
    - *ORBexpress* (Objective Interface Systems)

Outline

1. Is the SCA what you think it is?
2. SCA Myths and Realities
3. Future SCA core frameworks

Future SCA Core Frameworks

- **Next generation core frameworks will be smaller and faster; that's no secret**
  - The current core frameworks have come a long way; They are faster and smaller than before
  - But the next generation core frameworks will bring the SCA to a whole new level
- **Core Frameworks optimizations techniques fall into two different categories:**
  - Task optimizations
  - Static deployment optimizations

Future SCA Core Frameworks

- **Task optimizations**
  - This technique consists in optimizing tasks that a core framework performs during the deployment of a component
  - The current generation of core frameworks typically implement several task optimizations
  - Examples of tasks optimizations:
    - using native file system access
    - Transforming indirect connections into direct connections
    - Eliminating the use of an XML parser

Future SCA Core Frameworks

- **Static deployment optimizations**
  - This technique consists in eliminating tasks that a core framework performs during the deployment of a component
  - Next generation Core Frameworks will be able to achieve this by saving deployment context information
  - Examples of static deployment optimizations:
    - Remembering where components have previously been deployed
    - Avoiding remote copies of component artifacts by using a caching mechanism
    - Remembering the resolved value for component properties
    - Etc.

Future SCA Core Frameworks

- **Static deployment optimizations**
  - Ultimately, a core framework could remember every decision made to deploy an application
    - Would allow applications to be redeployed skipping all the tasks but those related to instantiation, configuration, and inter-connection
  - The good news is that static deployment optimizations don't require new SCA APIs
    - Doesn't require SCA change proposals
    - Existing SCA application don't need to be modified

Summary

- **The SCA is a Component Based Development architecture**
  - Not specific to military SDR
  - Can be used for any embedded application
  - There is an eco-system of COTS SCA products
- **Real-Time CORBA is essential for performance**
- **SCA Core Frameworks can be made smaller, faster, and more deterministic without having the change the specification**

Questions ?

SCARI++ Software Suite

- **CRC offers the most complete solution for SCA development**
  - Development tools
  - Zero-merge code generation
  - Monitoring tools
  - Core Framework
  - Training
  - Consulting
  - Certification expertise

SCARI++ Software Suite

- **Team has over 6 years of SCA experience**
  - CRC trained companies around the world
  - CRC offers professional services to help companies gear-up for the SCA market

SCA Architect™ Highlights

- **SCA Architect™ is based on the widely adopted Eclipse framework**
  - Provides platform independence (Windows, MAC, Linux, etc)
  - Every major vendor embedded software supports Eclipse
  - There is a enormous number of plug-ins to choose from to help with every aspect of software development (code authoring, documentation, unit test, configuration management, UML, etc.)
- **Implements real-time model validation; prevents you from creating invalid XML descriptors**
  - Validation messages are hyperlinked to models

SCA Architect™ Highlights

- **Provides model re-factoring capabilities**
  - Common model validation errors can be fixed through suggested re-factoring
- **Can reverse-engineer models for existing components**
- **CRC's development tools have been designed with an intimate knowledge of the SCA specification**
- **Simplifies Configuration Management**
  - Perform CM tasks at the model level instead of at the artifacts level

SCARI++ CF Highlights

- **CRC also provides a Core Framework: SCARI++**
  - Built from the ground-up for embedded platforms
  - Implementation of the SCA version 2.2
  - Very portable POSIX implementation
  - Implemented with lessons learned from the JTRS Certified SCARI Core Framework
  - Comes with a POSIX Executable Device, an AudioDevice and demo applications

SCARI++ CF Highlights

- **Provides extra APIs for introspection**
  - Optimized way of obtaining deployment information
  - Can show established connections during run time
- **Supports the deployment of components on standalone remote *Devices***
  - *Devices* can be started manually and report to a remote *DeviceManager*
- **Allows *Devices* to be collocated in a same address space**
  - Dramatically increase rate of communications between *Devices*

SCARI++ CF Highlights

- **Transparently optimizes connections so they can be performed as fast as possible**
  - Indirect connections are transformed into direct connections which requires much less CORBA interactions
- **Supports orderly shutdown of devices even when running applications**
  - A Device can be released or killed while it is running an application

SCARI++ CF Highlights

- **Available for different operating systems:**
  - INTEGRITY
  - VxWorks
  - Linux
  - Yellow Dog
  - and soon for LynxOS

- **Available for different ORBs:**
  - ORBexpress
  - TAO