Communications Research Centre Canada
An Agency of Industry Canada

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

The SCA: Myths vs Reality

Is the SCA what you think it is?

Steve Bernier
Researcher, Project Leader
Advanced Radio Systems

Outline

1. **Overview of the Software Communications Architecture (SCA)**
2. **Is the SCA too slow ?**
3. **Is the SCA too fat ?**
4. **Summary**

1. 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 but an implementation-independent set of rules that constrain the design of systems to achieve the above objectives**

1. SCA Overview

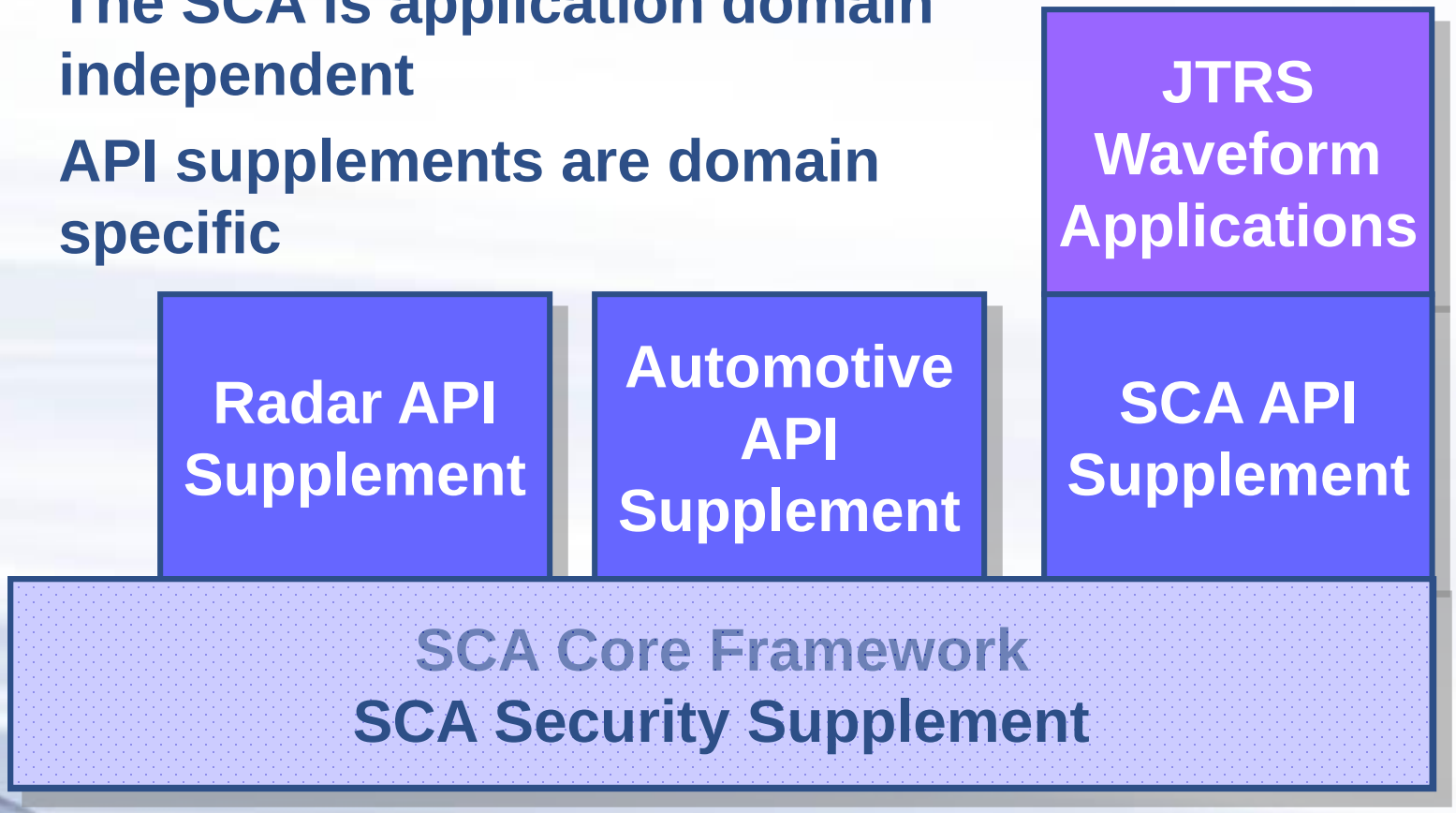
- **Myth #1: The SCA is only for military Radios**
  - While its true the SCA specification was developed for the US DoD JTRS program, the reality is the core framework specification contains no military features at all !
- **Myth #2: The SCA is for building Software Defined Radios**
  - None of the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

1. SCA Overview

- **The SCA Core Framework specification (version 2.2.2) is made of five documents:**
  - Main document (130 pages)
  - Appendix B – Application Environment Profile (21 pages)
  - Appendix C – IDL (41 pages)
  - Appendix D – Domain Profile (64 pages)
  - Appendix D – Attachment 2 – Common Properties (4 pages)
- **Previous releases of the SCA specification had two extra documents named Security Supplement and API Supplement**
  - These documents were last published in 2001
  - The security supplement adds RED/BLACK centric APIs
  - The API supplement adds communications/radio centric APIs

1. SCA Overview

- The SCA is application domain independent
- API supplements are domain specific



1. SCA Overview

- **The SCA specification describes how to create a platform that can host SCA-compliant applications**
  - It describes how a platform makes its devices and services available to applications
  - It also describes how applications are deployed

- **The SCA describes an architecture capable of doing what every real-time operating systems does:**
  - Load and execute applications
  - Specify priorities and stack sizes for individual tasks

1. SCA Overview

- **So what is so unique about the SCA ?**
  - It is platform independent
    - Supports any operating system*, processor, and file system
  - It is a scalable distributed system
    - Supports single processor applications the same way it supports multi-processor applications
  - An SCA platform can be made of several nodes with different processor architectures running different operating systems supporting different file systems
- **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

1. SCA Overview

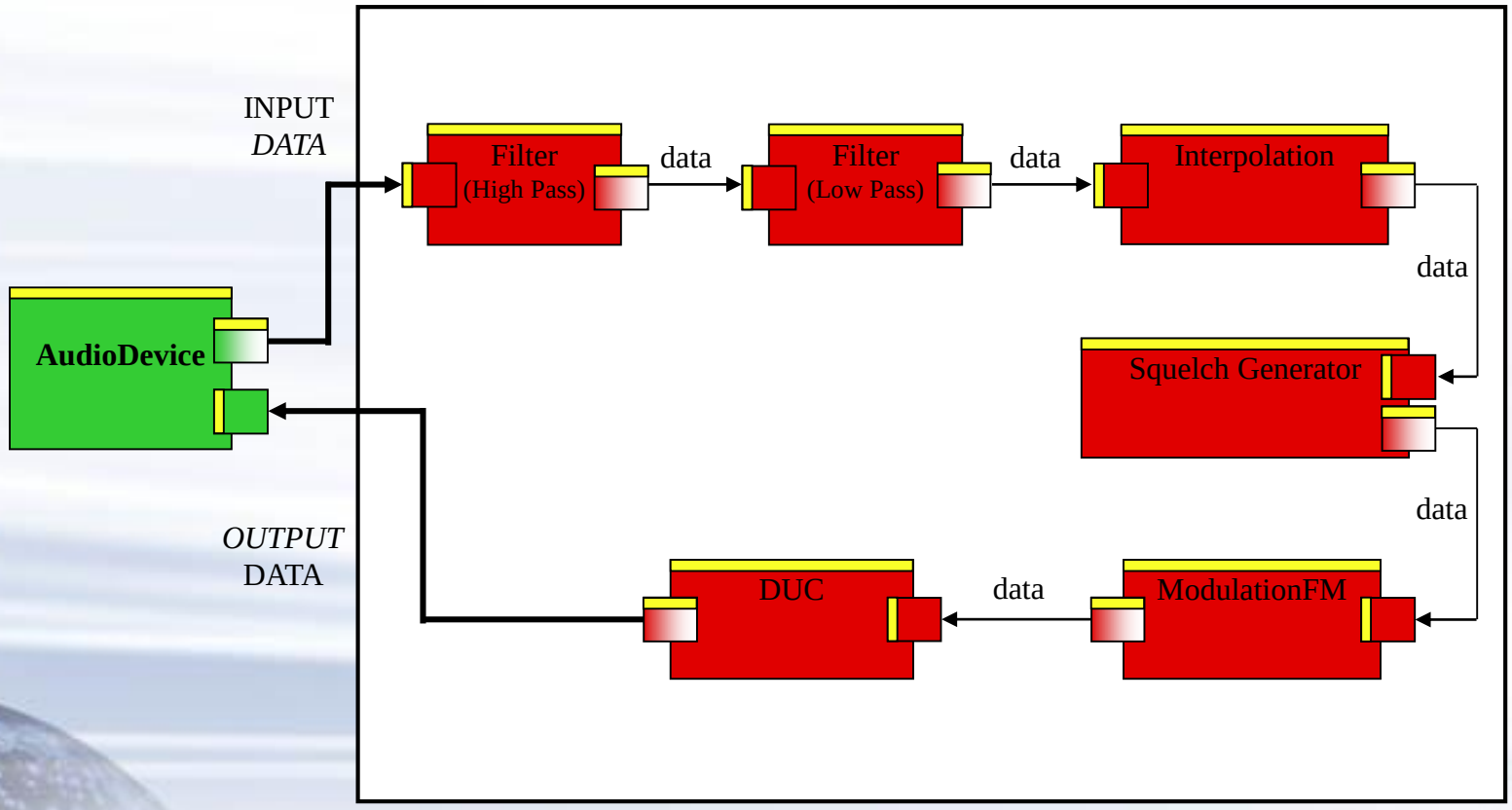
- **What is Component Based Development (CBD) ?**
  - **Definition:** an architecture which allows the creation, integration, and re-use of components of program code
  - CBD is a new development paradigm where the smallest unit of software is a **component**
  - With CBD, an application is 'assembled' using **software components** much like a PCB is populated with hardware components
- **CBD is a very popular paradigm for application development**
  - **'.Net'** (from Microsoft) and '**EJB'** (from Sun Microsystems) are two very popular CBD architectures
  - The OMG CORBA Component Model (**CCM**) is another example of a CBD architecture

1. SCA Overview

- **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
  - It is **not a function** compiled and stored in a static library; it's executable code which provides a service
- **A software component is a "black box"**
  - Application designer is concerned with what a component does, not how it does it
  - Creating an application requires component assembly-level information; the equivalent of a "spec sheet"
    - With the SCA, this information is located in a database called the "domain profile"

1. SCA Overview

- **Here's an example of a component assembly**
  - FM modulation application

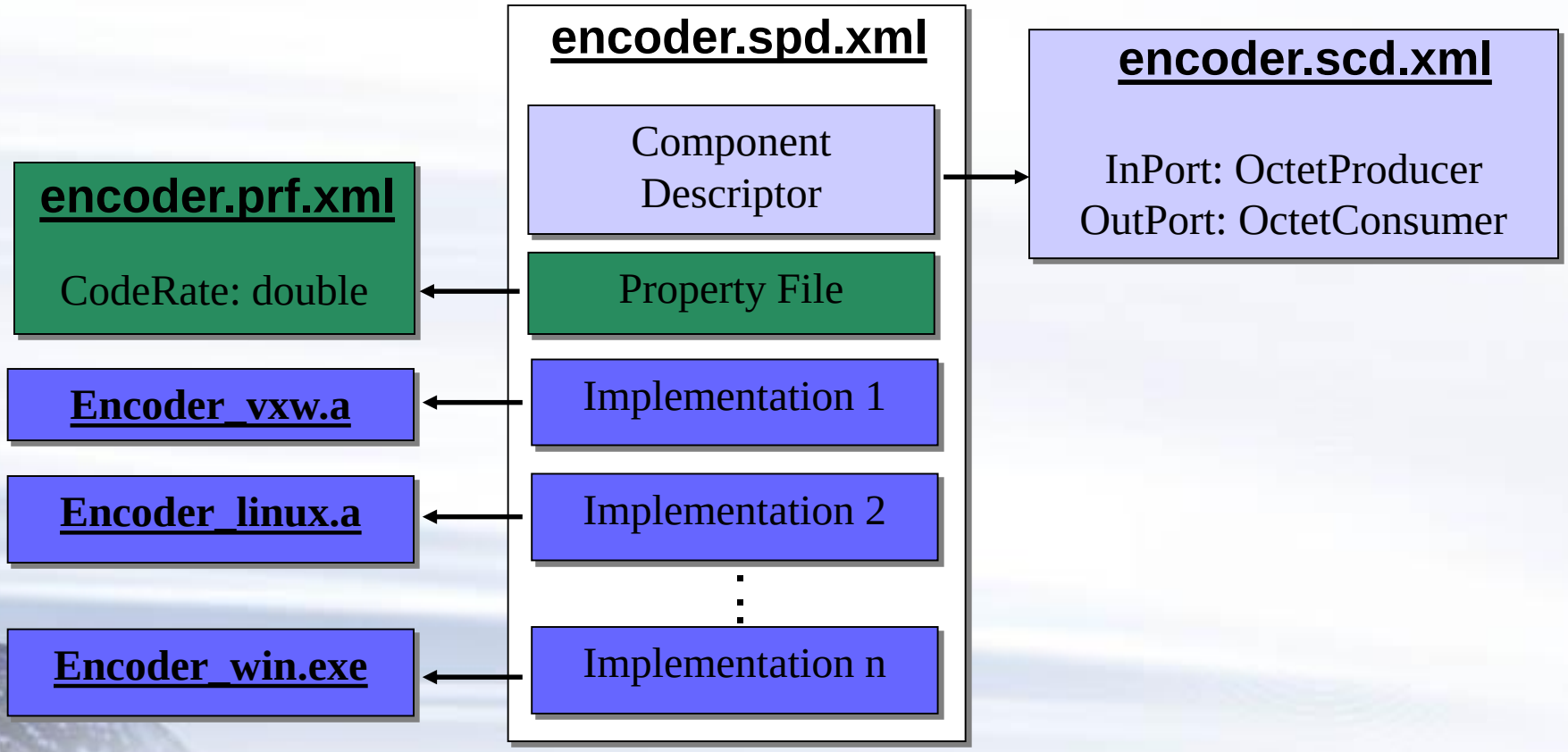


1. SCA Overview

- **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
    - Provides fine control over the deployment of components
- **With the SCA, a software component can be packaged with several implementations**
  - Each implementation is characterized by **capacity** requirements (run-time memory, mips, channels, etc.) and **capability** requirements (OS, processor, etc.)

1. SCA Overview

- **Here's what the definition of an SCA software component (spec sheet) looks like:**



1. SCA Overview

- **In summary, the SCA is a Component Based Development architecture which is platform-independent and device-centric**
- **The SCA is <u>not</u> specific to SDR or military applications**

Outline

1. Overview of the Software Communications Architecture (SCA)
2. Is the SCA too slow ?
3. Is the SCA too fat ?
4. Summary

2. Is the SCA too Slow ?

- **In order to measure the speed of the SCA, lets look at different common use cases for an SCA platform:**
  - Use Case 1: Booting an SCA platform
  - Use Case 2: Installing an application
  - Use Case 3: Running an application
- **Use Case 1 involves starting a number of SCA components**
  - Starting software components means creating a number of process/tasks
  - This is not unique to the SCA, it's required for any SDR platform
  - How fast can your RTOS create/spawn a process/task ?
  - How fast can application artifacts be copied from storage memory to run-time memory ?

2. Is the SCA too Slow ?

- **Use Case 2 involves loading all the artifacts associated with an application into storage memory of an SCA platform**
  - Again, this is not unique to the SCA
  - Depends on the speed of the bus/memory and the size of the artifacts
  - Installation of an application is typically done at the factory when time is not very critical

2. Is the SCA too Slow ?

- **Use Case 3 involves starting application software components**
  - A target device must be chosen for each component
    - This may take some time, but the SCA offers a way of avoiding run-time decisions
  - The chosen implementation for each component must be loaded into the runtime memory of the target device
    - Depends on the speed of the bus/memory
    - This can be an issue; not unique to the SCA
    - Better SCA implementations can alleviate this problem

2. Is the SCA too Slow ?

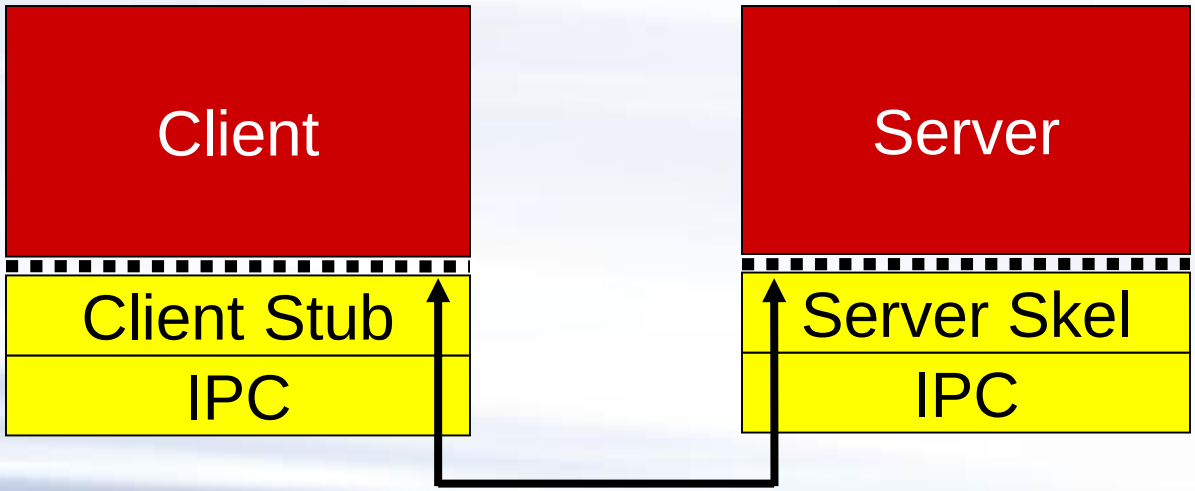
- **Use Case 3 also involves data processing**
  - SCA application components must communicate with each other to perform signal processing
  - With the SCA, communications are normally implemented using CORBA
  - Application throughput is therefore limited by CORBA
  - How fast is CORBA?

2. Is the SCA too Slow ?

- **CBD requires inter-process communications (IPC) to allow components to interact**
  - A software component can run as a process or task
  - Cannot assume components always run in a process
- **The SCA mandates the use of CORBA as the primary form of communications between software components**
  - CORBA is very scalable and provides a single model for component communications
    - Communications APIs are the same whether components are across the network, on the same board, or in the same process
  - CORBA is COTS

2. Is the SCA too Slow ?

- CORBA supports several IPC mechanisms
- However, most commercial CORBA products are implemented using the Socket IPC mechanism for TCP/IP



2. Is the SCA too Slow ?

- **Myth #3: CORBA is slow!**
  - The speed of communications between components is directly related to the IPC mechanism being used
  - Using TCP/IP can be slow and it's often a bad choice for embedded systems
  - In reality: CORBA is NOT slow but TCP/IP can be.
- **Real-time CORBA products typically support several IPC mechanisms**
  - UDP, Multicast, Shared Memory, etc.
  - Developers can add support for other IPC mechanisms

2. Is the SCA too Slow ?

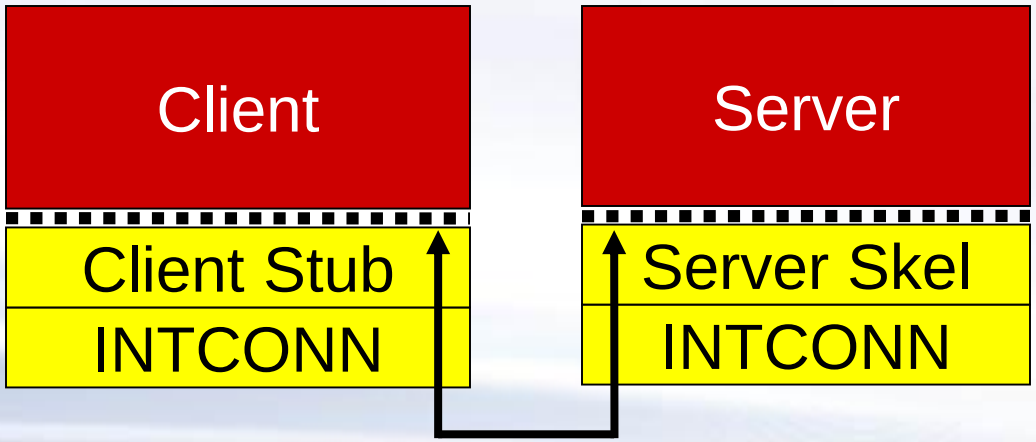
- **Using a Real-time ORB makes a great difference!**
  - For instance, ISR Technologies manufactures an SCA radio which comes with two applications: Voice over IP and Video
  - Using the ORBexpress (i.e. CORBA) and the INTCONN IPC, they were able to lower the ping delay between two radios to ~10μsec vs ~300 μsec for TCP/IP

iSR TECHNOLOGIES

CRC SCARI++ CF

Green Hills SOFTWARE, INC.

OBJECTIVE INTERFACE



2. Is the SCA too Slow ?

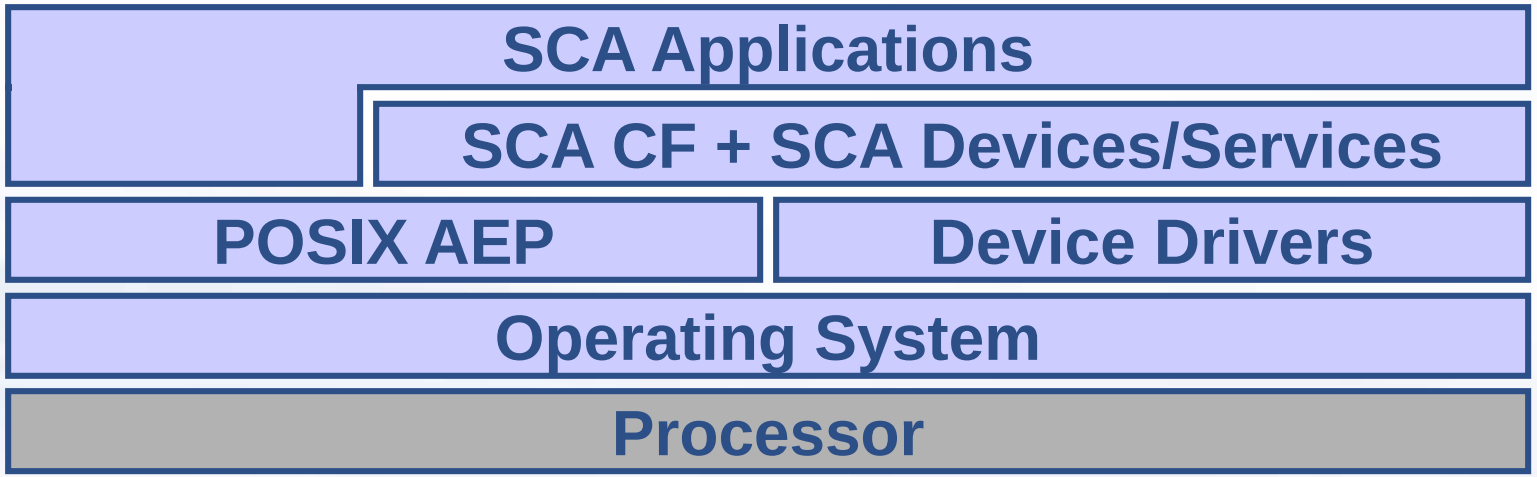
- **Is CORBA slow?**
  - The real question is: How fast is your IPC mechanism?
- **If there's an IPC mechanism that's fast enough for your application, then you should use CORBA!**
  - no learning curve for the IPC
  - Provides IPC independence
    - if a new and faster IPC becomes available, you can use it without changing any source code
- ***Conclusion:*** **The SCA is as fast as the CORBA product being used**
  - The SCA does not get involved in the communications between application components; only CORBA does!

Outline

1. Overview of the Software Communications Architecture (SCA)
2. Is the SCA too slow ?
3. Is the SCA too fat ?
4. Summary

3. Is the SCA too Fat ?

- **Here's a block diagram of an SCA platform**



- **The SCA requires an operating system capable of loading new code dynamically**
  - Many SDRs only use a simple scheduler/kernel which only supports static images
  - Essential to support new applications without rebooting

3. Is the SCA too Fat ?

- **The SCA does not require just any OS**
  - OS must provide a subset of the POSIX APIs
  - Essential to enhance application portability
- **The SCA Core Framework**
  - Provides platform control
    - Install/launch applications
    - Start node components to gain access to devices
  - Requires an XML parser
    - Xerces-C++ requires 2.6 MB of static footprint and typically around 4 MB of dynamic footprint
  - Requires CORBA generated code
    - Static footprint: 750K (ORBexpress) or 3.3 MB (TAO)

3. Is the SCA too Fat ?

- **SCA Application**
  - Is an assembly of several software components
  - Each component requires CORBA generated code
    - Static footprint: 730K for ORBexpress or 3.3M for TAO
- **Quantifying the footprint requirement for an SCA radio is difficult**
  - Is directly related to the number of software components required by the platform and the applications
  - Currently, a full featured SCA CF and a node with a couple devices and services will require around 25 MB of footprint
    - The Xerces-C++ XML parser will use ~40%
    - CORBA generated code ~30%

3. Is the SCA too Fat ?

- **The CRC AudioEffect demonstrator runs in ~50 MB of total footprint**
  - Embedded Planet PPC405 board (EP405), 128MB RAM
  - CRC' SCARI++ CF for INTEGRITY/ORBexpress
  - Node description:
    - Full featured *DeviceManager*
    - *ExecutableDevice*
    - *Log* service
  - Application with 3 components which perform Echo and Chorus effect on an input voice signal
  - Xerces-C++ XML parser
  - INTEGRITY Kernel with POSIX and VFS/NFS support
  - ORBexpress Name Service

3. Is the SCA too Fat ?

- **The ISR JTRS Demo Set requires ~51 MB of total footprint**
  - VoIP 256 Kbits/s BFSK, Video Waveform1024 Kbits/s BFSK
  - Xilinx Virtex-4 FPGA, 128MB RAM
  - CRC' SCARI++ CF for INTEGRITY/ORBexpress
  - Node description:
    - *DeviceManager*, *DDCDevice*, *DUCDevice*, *EthernetDevice*, *FGPAExecutableDevice*
  - 2 SCA applications of 2 components each
  - Xerces-C++ XML parser
  - INTEGRITY Kernel with POSIX and VFS/FFS support
  - ORBexpress INTCONN support
  - ORBexpress Name Service

3. Is the SCA too Fat ?

- **Is the SCA is too fat?**
  - Reality: the SCA can be large for a small form factor SDR which will never be upgraded post-manufacturing
  - Won't fit on a cell phone…yet!

- **SCA CF Implementations can be made "lighter" while maintaining compliance with the SCA**
  - Its just a question of time…

Outline

1. Overview of the Software Communications Architecture (SCA)
2. Is the SCA too slow ?
3. Is the SCA too fat ?
4. **Summary**

4. Summary

- **The SCA is a Component Based Development architecture**
  - Not specific to military SDR
  - Can be used for any embedded application
- **The SCA can be slow**
  - Using a Real-Time CORBA product is essential
- **The SCA footprint is reasonable and will improve with time**
  - 64 MB is enough for many platforms
  - The SCA can be made smaller without having to change the specification

Questions ?

SCARI++ Software Suite

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

SCARI++ Software Suite

- **Team has over 6 years of SCA experience**
  - CRC trained companies from around the world
  - CRC helps companies to gear-up for the SCA market
- **CRC's SCARI++ Core Framework is available for the most popular operating system and processors**
- **CRC will soon offer a completely new Eclipse-based Integrated Development Environment (IDE)**

IDE Highlights

- **CRC offers an complete Integrated Development Environment (IDE) for the SCA**
  - Core Framework Independent
- **Implements real-time model validation; prevents you from creating invalid XML descriptors**
  - Validation messages are hyperlinked to models
- **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**

IDE Highlights

- **Based on the widely adopted Eclipse framework**
  - Provides platform independence (Windows, MAC, Linux, etc)
  - Every major vendor of the embedded domain support 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.)
- **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