Experience Report on Developing a Software Communications Architecture (SCA) Core Framework

OMG SBC Workshop
Arlington, Va.
September, 2004

Dominick Paniscotti
Communication, Navigation, Identification and Reconnaissance

SCA Overview
SCA Definitions and FAQs

• What exactly is a Core Framework?
  – Per the SCA…A set of defined interfaces in three categories:
  – Base Application, Framework Control, Framework Services

• Industry groups these according to deliverable products
  – CF developers provide implementations for Application deployment and configuration
  – CF developers may also provide tools to help end users of their product
    • To help Application developers write SCA compliant Waveforms
    • To help Radio System developers build SCA compliant radios
  – Application Developers provide implementations needed to support their waveform specifications
  – Radio System Developers provide implementations needed to supply SCA compliant radios to Application developers and to the end-users

• Different developers use different CF interfaces to accomplish their goals
SCA Definitions and FAQs

• What does it mean to have “A CF running on a radio”?
  – The radio has SCA-compliant implementations supporting deployment and configuration
  – Radio processing elements have SCA-compliant DeviceManagers
  • Allowing instantiation of SCA-compliant Devices
  – Radio HW elements are abstracted using SCA-compliant Devices
  – One processing elements runs an SCA-compliant DomainManager
  • Allowing the installation of Applications
  • Supporting the creation of SCA-compliant ApplicationFactory(ies)

• The SCA refers to the above as an Operating Environment
  – Also encompasses the Operating System, associated device drivers, and CORBA ORB
The Parts of SCA Software Development

- Core Framework Developers
  - Implement DomainManager, DeviceManager(s), ApplicationFactory, and Application
  - To a large degree, the finite set of functions the SCA specifies must be coupled
    - e.g. DomainManager and ApplicationFactory implementations are coupled together beyond their SCA interfaces
  - The end products are capable of parsing Domain Profiles and instantiating Applications via Devices

- CF developers may also specialize in implementation of Loadable and Executable Devices
  - These platform-dependent entities differ from other Devices in that they only load and execute component software on particular platforms
    - e.g. - SCA compliant Loadable and Executable Devices can be developed for Linux, LynxOS, Integrity, VxWorks etc.
The Parts of SCA Software Development

• Radio System Developers
  – Develop the remaining Devices that abstract physical hardware.
    • e.g – Device abstractions for Modems, Antenna, I/O, etc.

• Application Developers
  – May develop entire Applications
  – Or may specialize in the development of particular components
    • For use by other Application Developers

• Service Developers
  – Develop entire radio appliqués that serve general radio needs
    • E.g. – Fault Management, Spectrum Management, etc.
  – Note however, Services are still quite loosely defined in the SCA
Waveform Development

System Definition/Layout
Waveform Component Definition
Waveform Component Implementation
Partitioning
Component Connections
Waveform Definition
Generate Output

Device

Device Component Library

Device

Waveform Component Library

Waveform Developer

Install onto Radio

Domain Profile & Implementations

Physical Radio

BAE SYSTEMS Information and Electronic Systems Integration Inc.
Communication, Navigation, Identification and Reconnaissance (CNIR) - Copyright 2004
Application Deployment

Physical Radio

Application

Application Factory

Physical Element

Physical Element

Device

Device

Reads Installed Files

Connects components

load/execute
# Legacy Radios Contrasted to SDRs

<table>
<thead>
<tr>
<th>Typical Legacy Radio</th>
<th>SDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 channel / waveform</td>
<td>Multiple simultaneous channels and waveforms</td>
</tr>
<tr>
<td>No application abstraction</td>
<td>Software architectures (e.g. JTRS SCA, OMG SWRADIO)</td>
</tr>
<tr>
<td>CRYPTO – Single waveform, non-programmable encryption</td>
<td>Programmable multiple waveform encryption</td>
</tr>
<tr>
<td>Mature CPUs/custom radio OS</td>
<td>Immature CPUs/POSIX OS</td>
</tr>
<tr>
<td>1 or 2 CPUs – few interfaces</td>
<td>Many distributed CPUs – tens of interfaces</td>
</tr>
<tr>
<td>Service designed custom networks</td>
<td>Self-Forming Networks</td>
</tr>
<tr>
<td>Voice and limited data services</td>
<td>Voice and data, routing, radio services (message translation, spectrum management, etc.)</td>
</tr>
</tbody>
</table>
Benefits and Impacts of Mandating a Framework
Software/Middleware Benefits and Impacts

- Key enablers for portability & re-use are software-based waveforms and middleware
- This portability comes at a cost
  - Additional software layers
  - Remote procedure calls and associated transport overhead
- Software/middleware costs
  - Increased I/O overhead, due to middleware layering
  - Reduced determinism (higher jitter, latency) due to packetization
- Operating system costs
  - More layers to access physical hardware
  - Address space separation overhead
    - Hardware independence comes at cost
    - Layers between application software, operating system, board support package and device drivers
Addressing Software/Middleware Impacts

• Moore’s Law continues to hold
  – Processing power is increasing
  – Interconnect speed is increasing
  – Interconnect latency is decreasing
  – Power consumption is decreasing
  – Size is decreasing
• These benefits advance at compounded rates
• Middleware vendor advances
  – Standardization allows ORB vendors to increase performance and lower footprint without impact to application software
    • Real-time deterministic ORBs implementations available
    • High-speed, low-latency pluggable protocols
    • Collocation optimizations
    • Zero-copy techniques
• Preemptive real-time operating systems
  – Low-latency context switching
  – Efficient compilers
Software Architecture Benefits and Impacts

- Software frameworks are key enablers to provide
  - Deployment and configuration mechanisms
  - Component-based software development techniques
  - Radio platform standardization
  - Scalability across radio platforms
- They properly segregate the responsibility of the waveform, radio platform and radio service developers
- Software frameworks themselves typically have little performance impact
  - Once done deploying a waveform the framework is not involved in waveform processing
  - In other words… “It steps out of the way”
  - Does not involve itself in the data-flow paths of a waveform
  - Re-involves itself on request to tear down waveform
Software Architecture Benefits and Impacts

- **Framework speed optimizations**
  - Focused on the speed of waveform deployment and teardown
  - Cannot optimize speed at which waveform moves data
    - This is a function of the middleware and O/S used
- **Framework footprint optimizations**
  - Focused on reducing the “run time” footprint
  - Examples
    - Memory used per component deployed, per connection, etc.
    - Middleware and O/S play a role here as well
    - O/S’s supporting dynamic libraries lower total memory cost
High-Speed interconnects and I/O Benefits and Impacts

- The evolution of higher-speed I/O interfaces and higher-performance processors mitigates much of the additional software layering and middleware overhead.
- Efficient transports that increase throughput and lower latency can be written for these interfaces.
- Middleware can be layered on these efficient transports to lower middleware impact even more.
- Advancement in these areas are mostly revolutionary vs. evolutionary.
  - Requiring users to abandon entire implementations each time a new quantum leap is made.
Technology areas being addressed by industry for application in SDRs

• Software/middleware ✓
  – Metrics successfully collected, all near-term requirements satisfied

• Software frameworks ✓
  – JTRS/SCA already being applied to systems
    • JTRS Steps 2a, 2b, 2c, Cluster 1, AJCN, WIN-T, FAB-T

• High-performance, low-power computing resources ✓

• High-speed I/O and interconnects ✓
  – Gigabit Ethernet now directly integrated into many CPUs, bridge chips, and FPGAs
  – High-speed switching fabrics on vendor roadmaps, gaining acceptance, e.g., RapidIO, PCI-Express
Techniques required to meet performance constraints

• Use high performance middleware designed for real-time environments
  – Use middleware constructs that eliminate “deep copies” between waveform components wherever possible

• Use efficient and high performance coding language and compilers
  – Apply optimization techniques
  – Use profiling tools to find additional inefficiencies
  – Disabling native exceptions can provide significant performance increases

• Proper partitioning of software components is key
  – Partition components with high throughput/low latency requirements in the same address space.
  – Use middleware collocation optimizations
    • e.g. Turns CORBA calls into C++ call - Overhead virtually disappears
Collocation Optimizations a must!

Device Drivers

Non-WF Addr Space

Device

Waveform Component

Waveform Component

Device Drivers

Non-WF Addr Space

Device

CORBA

Inter-Process Communication

CORBA

Physical Processor

CORBA
Advantages and Successes from Using Middleware

- Lessons learned in building and fielding JTRS radios
  - Of all the elements of the software system middleware capable components were the least problematic to build and integrate
    - RTOS, device drivers, middleware, software frameworks, and application components
  - Apart from developing the waveform software, a large percentage of effort is in debugging drivers, BSPs, hardware, and RTOS
- Middleware like CORBA is standardized, therefore easy to use
- Location transparency and architecture portability allows
  - Software development and testing on non-target platforms, which eases integration onto target
  - For the reuse, repartitioning and redeployment of software onto other radio platforms
Communication, Navigation, Identification and Reconnaissance

SCA Myths
SDR/SCA Myths

- “Moore’s Law won’t help”
  - Faster, lower-power processors and interconnects make SDRs a reality in wider bands and smaller footprints

- Wideband can’t be done
  - It is already being done in SCA-based systems

- Processing overhead and power consumption unacceptable
  - High-performance, low-power, and low-cost processors with power-saving capabilities available today
  - Look no further than your own PDA or Blackberry
SDR/SCA Myths

• Doesn’t scale to handheld devices
  – BAE Systems has demonstrated full SCA-compliant framework and waveform running in iPAQ PDA

• Portability and reprogrammability ends at the MODEM boundary
  – DSP middleware available today and mature

• Requires extensive experience with CORBA, XML, C++, SCA, etc. and takes a long time to come up to speed
  – Pattern-oriented software development, OMG MDA, and model-integrated computing techniques can readily address this
Patterns and Principles of framework/component design applicable to the SCA
Patterns and Principles of framework/component design applicable to the SCA CF

• Extension Interface/Objects Pattern
• Component Configurator Pattern
• Proxy
• Wrapper Façade, Facade
• Template Method
• Strategy

• Details of how these patterns are applied to the CF can be found in Monday’s *Effective Component and Application Development using the Software Communication Architecture* Tutorial
Extension Objects Pattern – Benefits Summary

- Maintain the CF::Resource and CF::Device abstractions while simultaneously providing for unanticipated “interface” extensions

- Keeps clients of waveforms decoupled from various interfaces that they don’t use

- Allows for easy addition of new services inside components with a minimal impact to client code

- Prevents bloated interfaces
Component Configurator – Benefits Summary

- Waveform applications can load and unload their component implementations at run-time without having to modify or statically relink the entire application.
- Waveform applications can be easily redeployed and redistributed depending on changes in operation environment (e.g. load balancing, tighter security concerns).
- Deployment of waveforms no longer coupled to the deployment of the Core Framework on any other part of the Operating Environment.
- CF::Application provides a centralized repository and administrative mechanism for waveform management.
- SAD file provides fast and easy mechanism to change the “schematic” of the waveform without recompiling or relinking.
Proxy and Wrapper Façade Pattern – Benefits Summary

- Complex waveforms hidden behind single abstraction
- Well known interface established between Core Framework and the Waveform (CF::Application and Assembly Controller bridge the two)
- Hide distribution mechanics (e.g. Components)
- Seamlessly allow the addition of new connection semantics (e.g. fan-out and controller number of connections)
- Simplify access to OS APIs (e.g. APIs for Threading)
Template Pattern – Benefits Summary

- Allows clean separation of much of the SCA required component mechanics from the business logic of the particular waveforms
- Provides higher level frameworks for many of the SCA facilities.
Strategy Pattern – Benefits Summary

• XML Parsing
  – Encapsulate XML parsing algorithms and technologies and make them pluggable
    • e.g. DOM, SAX, proprietary method
  – Decouple the Core Framework from the particular XML technology being used.

• Capacity Allocation
  – Encapsulate capacity model algorithms from the Devices that implement them.
  – Allows component developers to leverage “capacity frameworks” in higher level classes while at the same time “tweaking” their behavior with component-specific behavior
Questions?