CORBA in SIGINT Systems
A Case Study

Presented by
John F. Masiyowski, Software Engineering Manager
Raytheon
C3I and Information Systems
Strategic Systems Business Unit
7700 Arlington Boulevard
Falls Church, VA 22042-2900 USA

(703) 560-5000 jmasiyowski@raytheon.com
7 June 2001
Agenda

- Who is Raytheon?
- Terminology
- Legacy System Architecture
- Next Generation Sensor System Diagram
- Why CORBA?
- Next Generation Sensor System Architecture
- Sensor System Characteristics
- Example: Applications Loader
- Sensor System CORBA Usage
- Lessons Learned
- Concluding Remarks
Who is Raytheon?

- Headquartered in Lexington, Massachusetts
- Celebrated Its 79th Anniversary This Year
- $20+ Billion Global Technology Leader
- 105,000+ Employees World-wide
- Operates in Three Business Areas
  - Commercial Electronics
  - Aircraft
  - Defense Electronics
- C³I and Information Systems
  - providers of information to the warfighter

http://www.raytheon.com
**Terminology**

- **Real-Time** - an on-line computer processing system that receives and processes data quickly enough to produce output to control, direct, or effect the outcome of an ongoing activity.

- **Embedded System** - computers that are built into a product and whose prime function is not that of a computer.
Legacy System Architecture

Sensor:

3 General Purpose Processors - mostly C language
20 Embedded processors/controllers various types and capabilities
custom hardware and software - few commercial stds
Few COTS items (hardware and software)

• Architecture:
  • physical-functional organization
  • not distributed, “stove piped”
  • not based on a framework
  • message passing based
  • message API on-top of sockets layer
  • “closed”
Common Software Framework

Key Characteristics:
- provides layered information hiding
- implementation-application isolation
- supports reuse of components
- separation of concerns
- provides abstraction
- extended services
- protected access
- common functions available applications
- network isolation
- Real-Time and UNIX platform support
- proven technology
- provides platform portability to applications

Open Source Equivalent: Adaptive Communication Environment (ACE)
Software Characteristics

- Message Passing Concept
  - client-server exchange messages
  - events generate messages
- Message Limitations:
  - encoded bits
  - language implementation specific
  - interface not clearly defined
  - numerous messages
  - request/response not identifiable
- Interface Limitations:
  - not structured
  - not clearly defined
  - language dependent
  - no O-O concepts utilized
  - Somewhat platform/tool dependent
  - tweaked to platforms

<table>
<thead>
<tr>
<th>Word</th>
<th>Message Format (simplified)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Message Identifier</td>
</tr>
<tr>
<td>2</td>
<td>Source Identifier</td>
</tr>
<tr>
<td>3</td>
<td>Destination Identifier</td>
</tr>
<tr>
<td>4</td>
<td>Message Length</td>
</tr>
<tr>
<td>M</td>
<td>Data Word 1</td>
</tr>
<tr>
<td></td>
<td>Data Word 2</td>
</tr>
<tr>
<td></td>
<td>Data Word N</td>
</tr>
</tbody>
</table>

0 <-------- bit number --------> 15
Next Generation Sensor System Diagram

CARS

Sensor Architecture
Why CORBA?

Sensor System Development Goals:

- Open Architecture; Well Defined I/Fs
- Scaleable to Meet User Requirements
- Software Re-programmable
- Leverage COTS
- Utilize Commercial Standards
- Legacy System Compatibility
- Platform (HW & SW) Independence
- Match Technology to Application

Rationale

- C² System and Sensor are distributed software systems: A number of applications executing on different processors and software environments
- Need to have a simple, transparent, uniform, and common method for these applications to communicate with each other efficiently and effectively
- **CORBA (a Commercial standard) provides this inter-application communication solution at the application [interface] and not at the network (sockets) level**
- Numerous Commercial and Public Domain CORBA Implementations exist
- **was really the next logical evolution!**
Next Generation Sensor Software Architecture

Key Software Characteristics
- CORBA Interfaces Levels (Hierarchical Architecture)
- Command and Control (C²) Based [Client-Server]
- Distributed Object-Oriented Architecture
  - Thread versus Structured
- Configuration Driven for Scalability
- Separate Command and Data Paths
- Interoperability

Interface (IDL) Architecture

System Level (External)
Subsystem Level (Internal)
Internal subsystem level

Software Object Categories
- BIT
- SP
- AM
- C²
- LAN

REMOTE/AUTOMATED SUBSYSTEM

System Processor
DSP
I/O
VMEbus
ATM
I/F & xlator
CORBA I/F
Custom I/F

Increasing granularity level (coarse to fine)
Sensor System Characteristics

• Embedded, Remotely Controlled, Instrumented, Heterogeneous Environment, Complex (C++ & IDL, C; 12-604 PowerPCs [200MHz] and 116-32040 DSPs; ~CORBA 2.0)
  System architecture is not a database centric model (no DB present in system)
  Locally distributed (close proximity) Clients and Servers
• Deterministic threads/paths: C² (i.e., signaling) and data (distribution & processing)
• Hard Real-Time (non-CORBA)
  • Sensor data manipulation and distribution (ex: streaming)
  • internal and external sources and destinations
  • latency < 1 uSec; throughput ≈ 10’s M bps
• Near Real-Time (CORBA)
  • IDL for all system, subsystem and internal subsystem C² functions
  • IDL organized in a hierarchical level with degrees of granularity
  • latency < 100 mSec for time-critical threads and events
  • majority of client-server relationships are one-to-one
  • 8 main system level interfaces expressed in IDL; total of 16 system level interfaces functional threads; ≈ 5k lines of IDL and ≈ 500k SLOC code (C & C++)
• Organization
  • System Level - Entire Sensor is a Server (multiple categories of services)
  • Internally - CORBA based-clients and servers
Example: Applications Loader

- **Purpose**: Provides the ability to:
  - load and unload software modules
  - facilitates system initialization
  - local CPU task management (start/stop)
- responses to commands via IDL interfaces
- resides on each PowerPC
- controlled locally or remotely

**IDL (Class Representation)**

```
AppLdr (server)
SetExecHost() GetBit1DataSize() ReportBit1Data() ProcessEcho()
LoadModule() UnloadModuleByKey() GetNumModules() TaskSpawn() TaskDelete() GetNumTasks()
```

**UML Sequence Diagram**

```
1: LoadModule ()
loadModule will return a MODULE structure

2: loadModule
3: new
4: AddModule ()
5: LoadModuleRsp ()
```

**IDL (Class Representation)**

```
AppLdrR (call Back)
CheckIn() GetBit1DataSizeRsp() ReportBit1DataRsp() ProcessEchoRsp()
LoadModuleRsp() UnloadModuleByKeyRsp() GetNumModulesRsp() TaskSpawnRsp() TaskDeleteRsp() GetNumTasksRsp()
```
Sensor System CORBA Usage

CORBA Features Utilized in the Sensor:
• Core ORB (DII, IR, ImR not utilized)
• Static Interfaces
• Lightweight Naming Service (transparent to application)
• mostly asynchronous invocations with calls backs

CORBA Features Desired for Sensor System:
• Naming Service
• Event or Notification Service
• Various forms of asynchronous invocations /w & /wo timeouts
• ORB instrumentation (health, status, & performance via HTTP or SNMP)
• higher fidelity access to and control of the network transport characteristics

Other:
• IDL Style Guide (would have been very helpful)
Lessons Learned

- Training is Highly Recommend for O-O, Language and CORBA
- Concentrate on Architecture/Interface Design (IDL & Hierarchy) - takes time
  - level of abstraction versus granularity - a critical tradeoff
  - careful with chatty client-server interfaces (LAN → WAN considerations)
- Must Configuration Manage System and Subsystem Interfaces (IDL files)
  - some changes can cause major ripples throughout the system
- Evaluate CORBA Implementations (Prototype, Benchmark, Measure)
  - Verify interoperability
- CORBA can be use for soft/near time embedded applications
  - understand CORBA performance and limitations, then architect system
- Still need to handle application level Quality of Service (QoS)
- Utilize standard CORBA features for interoperability and portability
- Evolve system components along with CORBA standards & implementations
- Not all RT/Embedded Systems needs ALL capabilities of [RT] CORBA
- Architect/Design System around CORBA features available today; try to allow
  for flexibility to incorporate future CORBA specification enhancements
Concluding Remarks

• CORBA is suited for certain Soft Real-Time Command & Control applications!
• Permits well-defined, interoperable, platform independent system level I/Fs → concentration on application versus low-level mechanics of communication
• Utilize technologies as appropriate; based upon applicability
• Knowing a CORBA Implementation’s operational characteristics (interactions) underneath the IDL interface is very important in embedded and real-time applications
• CORBA utilized in the CARS C² System as well!
• Considered using JAVA [back in 1996, JAVA was not ready for Real-Time or embedded Applications]
• Real-Time/Embedded CORBA Standard(s) will further promote CORBA based real-time/embedded applications

References:
Presentation: Masiyowski, J., Iona World 1999, “ORBIX and Object-Oriented Real-Time Distributed Processing System on an ATM Network”