Rockwell Collins, Inc.
Advanced Technology Center

An ORB for High-Assurance Avionics
“Safety Critical Real-Time CORBA”

OMG Safety Critical RFI response
http://cgi.omg.org/docs/mars/02-03-01.pdf

David Haverkamp
phone: 319.295.0758
e-mail: dahaverk@rockwellcollins.com

June 10, 2002
Developing High Assurance Software

- How is the development of software for safety critical systems different?
  - Focus on correctness and extreme reliability (high assurance)
  - Focus on process and certification (e.g. In accordance with DO-178B)
  - Focus on Verification & Validation (analysis and testing)
  - Focus on formal models and methods
  - Focus on (typically cross cutting) policies and strategies:
    - error detection and monitoring
    - fault tolerance
    - resource allocation
    - data integrity
    - partitioning
    - concurrency and synchronization
    - scheduling
    - testing
High-Assurance Middleware Issues

• **Do178B - Certification**
  – Software Level Definitions Levels A through E
    • Level A -- Most Critical. Potentially causing loss of life and aircraft.
    • Level E -- Minimal to no effect in safety or crew workload.
  – Failure Condition definitions
    • Catastrophic - conditions that prevent safe flight or landing.
    • Hazardous - conditions that reduce safety or functional margins.
    • Major - conditions that do not significantly reduce safety margins.
    • Minor - conditions that do not significantly reduce aircraft safety.

• **System Partitioning**
  – RTCA DO-255

• **Fault Tolerance**

• **Quality of Service**
High-Assurance Middleware Issues cont.

• A system is certified, not the software.
  – If the software is “reusable” the associated artifacts can be reused with another certification.

• Computer Languages typically used:
  – Ada - Traditional Safety Critical language.
  – C/C++
  – Java is of interest.

• AVSI work is helping with O-O certification issues
Source Code Access

• Source Code Access is necessary for Safety Critical Certification (e.g. Level A)
  – Licensing COTS source can be an issue.
  – Export compliance can be an issue.

• Open Source ORBs
  – Allow collaboration with other Researchers
  – Export compliance is not a problem.
  – Allow us to leverage expertise from a larger community of developers.
## Some Open-Source ORBs

<table>
<thead>
<tr>
<th>Lines of Code</th>
<th>Kernel Space</th>
<th>Programming Language Support</th>
<th>OMG</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>ACE TAO</strong></td>
<td>~180,000 in C++</td>
<td>No.</td>
<td>C/C++</td>
<td>Real-Time</td>
</tr>
<tr>
<td><strong>MICO</strong></td>
<td>~60,000 in C++</td>
<td>No.</td>
<td>C/C++</td>
<td>CORBA 2.3 + Components</td>
</tr>
<tr>
<td><strong>ORBbit</strong></td>
<td>~15,000 in C</td>
<td>Yes for Linux</td>
<td>C, C++, Ada, + several scripting languages.</td>
<td>CORBA 2.2</td>
</tr>
<tr>
<td><strong>ORBbit2</strong></td>
<td>~15,000 in C</td>
<td>Yes for Linux</td>
<td>C, C++, Ada(?), + several scripting languages.</td>
<td>CORBA 2.3</td>
</tr>
<tr>
<td><strong>Zen</strong></td>
<td>~9,000 in Java</td>
<td>No. (JVM)</td>
<td>Java</td>
<td>Real-Time Java</td>
</tr>
<tr>
<td><strong>JacORB</strong></td>
<td>~13,500 in Java</td>
<td>No. (JVM)</td>
<td>Java</td>
<td>CORBA 2.3 with Java OBV support.</td>
</tr>
</tbody>
</table>
Size Comparison (Simple Echo Client/Server) Open Source ORBs

David Haverkamp
phone: 319.295.0758
e-mail: dahaverk@rockwellcollins.com
ORB Performance - ORBit

Average Transfer Times

Plot of "Call Return and OneWay Operation" averages

David Haverkamp
phone: 319.295.0758
e-mail: dahaverk@rockwellcollins.com

June 10, 2002
ORB Performance - ACE TAO

Average Transfer Times

Plot of "Call Return and OneWay Operation" averages

June 10, 2002
Our View of ORB implementations

- Current Middleware is designed to solve problems in many problem domains.
- Code and Functionality that is not used increases Certification Costs and Effort.
- A Safety Critical Avionics Application must be able to restart in < 1 sec.
  - R-T is focused on Deterministic Operation.
- For Avionics applications only a subset of ORB functionality may be used
- Aspect Oriented programming can help reduce the code/footprint.
  - Improve initialization speed. Etc.
Performance Comparison
(Minimalist approach vs Real-Time ORB)

Message Length

Milliseconds

64 1200 2400 3600 4800 6000 7200 8400 9600 10800 12000

- ORBIT CR Long Struct
- ORBIT CR Record Struct
- ACE TAO CR Long Struct
- ACE TAO CR Record Struct

David Haverkamp
phone: 319.295.0758
e-mail: dahaverk@rockwellcollins.com

Advanced Technology Center

June 10, 2002
A High-Assurance ORB as outlined in RFI response

• High-Assurance CORBA will be a profile:
  – Minimum-CORBA with extensions
    • Features from Fault-Tolerance
    • Features from Real-Time
    • Small Footprint
  – No dynamic facilities
    • Interface Repository, Dynamic Invocation Interface, and Dynamic Skeleton Interface

• Have been doing some modifications to an ORB to test ideas.
  – Batching of Requests
  – Tool based optimization and code evaluation.
  – Application of Aspects
  – ORB Initialization changes.

• Have partnered with several universities on related work.
  – Washington Univ St. Louis, Kansas State University, Iowa State University.
Safety Critical CORBA

Gary Daugherty
Technical Contact for the Rockwell Collins
OMG Safety Critical RFI response
gwdaughe@rockwellcollins.com
319.295.4065
CORBA Certification Concerns (RFI Response)

• Related to the adaptation/use of COTS and open-source software, the most important issues are:
  – Size
  – Pedigree
  – Degree of openness/adaptability
  – Cost
  – Evolution of the software once adapted
  – Legal liability
  – Trust
Developing High Assurance Software

• **Approach represents three fronts:**
  – Fault avoidance
  – Fault removal
  – Fault tolerance
Developing High Assurance Software

• Fault avoidance
  – Formal process (DO-178B, FAA certification, independent V&V)
  – Safety assessment and analysis
  – Simplicity of design (KISS, use of language subsets)
  – Reuse of mature and formally verified policies and strategies (patterns)
  – Reuse of mature and formally verified components
  – Adoption of well established industry standards
  – Design automation (correctness by construction)
  – Formal models and methods
Developing High Assurance Software

• **Fault removal**
  – Reviews (requirements, design, coding reviews)
  – Simulation
  – Analysis (flow analysis, range checking, stack usage, timing, shared resource analysis, identification of dead and deactivated code, traceability of source to object code, model checking)
  – Requirements based testing (to structural coverage criteria, e.g. MC/DC)
  – Fault injection
Developing High Assurance Software

• **Fault tolerance**
  – Detection
  • Watchdog timers
  • Run time checks
  • Sanity checks and audits (acceptance tests)
  • Performance monitoring
  • Partitioning
  • Dissimilar hardware/software
  • Voting (of redundant and dissimilar units)
Developing High Assurance Software

• **Fault tolerance**
  – **Recovery**
    • Replacement of failed unit
    • System or subsystem level reconfiguration
    • Roll back to checkpointed state
    • Restart with next cycle
    • System or subsystem level reset (reboot)
High-Assurance Middleware

- **Safety critical versions of CORBA and Real-time Data Distribution standards**
  - With internal interfaces for plugging in safety critical patterns from the catalog
  - Standard sets of logical join/transformation points defined by the OMG
  - Safety critical software developer implements policies and strategies along the safety critical axis
  - Middleware provider identifies join points and transformation points within the supplied software in accordance with OMG standard
  - Others provide standard tools for analysis (used by middleware providers) and adaptation (used by safety critical software developers)
  - Tools may or may not be qualified, but provide reviewable/checkable output if they are not
High-Assurance Middleware cont.

• Layered functionality
  – OMG breaks existing middleware standards into sets of features
  – Middleware providers deliver kernel application + feature sets
  – Safety critical software developers use standard adaptation tools to compose tailored versions of the application
  – Tools may or may not be qualified, but provide reviewable/checkable source output if they are not
High-Assurance (Safety Critical) Proposal

• Safety critical UML profile
  – With support for multi-dimensional software development
  – Including a more general view of patterns (aspects, refactoring, general transformations, optimization for specific contexts)

• Patterns catalog
  – Extending work by AVSI, the FAA and NASA, Rockwell Collins, others
  – With all patterns reviewed and mapped to safety related issues and certification objectives

• MOF level model transformation standard
  – XSLT based or XSLT like transformation rules
  – Meta level foundation for patterns
High-Assurance (Safety Critical) Middleware Proposal

• Safety critical middleware standards
  – For Real-time Distributed Data (publish/subscribe)
  – For CORBA (a profile)
  – With internal join point/transformation point interfaces defined by OMG to support patterns in catalog
  – With functionality delivered in terms of kernel application + feature sets