High Assurance CORBA for Software Based Communications

Kevin Buesing
Objective Interface Systems
OMG SBC Workshop 2005

Table of Contents

- Safety-Critical or High-Assurance systems today require software that must meet stringent criteria
  - Reliability
  - Safety
  - Security
- Traditionally these systems have been custom designed
  - Expansion of this type of system => stove-pipe designs have become impractical
  - Looking to COTS

Introduction

- Availability of COTS High Assurance RTOSes will create demand for same level of robustness in middleware
  - RTOS is only part of the solution
  - CORBA, Minimum CORBA and Real-Time CORBA specifications provide a solid foundation to begin addressing middleware needs

What is High Assurance?

- High Consequence Attached to System Failure
- Undeniable evidence required to validate proper system functionality
- Typical Fields
  - Flight Control Systems
  - Secure Communication Devices
  - Medical Surgery Equipment
  - Unmanned Aerial Vehicles
  - Military Command and Control Systems
  - Nuclear Reactors
- Expanding to Other Fields
  - Automotive
  - Voice over IP (911 calls)

What is High Assurance?

- To the FAA:
  - One failure per 10^9 (1 Billion) hours of operation
    - How long *is* a Billion hours? Do the math!
      - $$1{,}000{,}000{,}000 \text{ hours} \times \frac{1 \text{ day}}{24 \text{ hours}} \times \frac{1 \text{ year}}{365.25 \text{ days}}$$
      - 114,077 *YEARS!*
- For National Security Systems processing our most valuable data under severe threat:
  - Failure is *Unthinkable*
- *How do we implement systems that we can trust to be this reliable?*

- High-Quality development process
  - Rigorous traceability from requirements to code
  - Quality assurance
- Predictable, rigorous base
  - Predictable language subsets
  - Predictable language runtime
  - High quality tools: compilers, linkers, operating systems

- Keep it simple
  - As small as possible
  - Restrict scope of evaluation
- Independently evaluated or certified
  - Certification currently varies greatly by industry

Overall goal: allow evaluation of software

Industry Standards

- RTCA DO-178B, *Software Considerations in Airborne Systems and Equipment Certification*
  - Adopted by FAA
  - Encompasses the entire project
- ARINC-653, *Avionics Application Software Standard Interface*
  - Time and space partitioning to prevent cascading failure of applications
- ISO-15408, *Common Criteria for Information Technology Security Evaluation*
  - International standard for assurance in IT
- DCID 6/3, *Protecting Sensitive Compartmented Information Within Information Systems*
  - Procedures for storing, processing and communication of classified intelligence
  - U.S. Federal directive

Industry Standards

- RTCA DO-178B, *Software Considerations in Airborne Systems and Equipment Certification*
- ARINC-653, *Avionics Application Software Standard Interface*
- ISO-15408, *Common Criteria for Information Technology Security Evaluation*
- DCID 6/3, *Protecting Sensitive Compartmented Information Within Information Systems*

Challenge: different standards for different industries

Challenge: Safety evaluation is done on a system.
Limited ability to re-use, discourages commercialization

Assurance Certification Goals

Common Criteria Basic Robustness (EAL3) Medium Robustness (EAL4+) High Robustness (EAL6+)	*MSLS / MLS Separation Accreditation* System High Closed Environment System High Open Environment Multi Level Separation
DCID 6/3 Protection Level 5	*Multi Nation Separation Accreditation*
DO-178B Level A	*Failure is Catastrophic*

Challenge: different requirements for different goals

Platforms Now Available

- Certifiable/Certified RTOS
  - Designed to conform to one or more standards
  - Three RTOS systems are under consideration to provide proof of concept:
    - Green Hills Software: INTEGRITY-178B
    - LynuxWorks: LynxOS-178
    - Wind River Systems: Platform for Safety Critical ARINC 653
- Future
  - MILS Separation Kernels (Green Hills, LynuxWorks, and Wind River)

Overall goal: allow evaluation of software

- Not Covered in the High Assurance ORB Profile
  - High-Quality development process
    - Not the subject of profile
    - Covered by DO-178B, etc.
  - Predictable, rigorous base - High quality tools
  - Independently evaluated or certified

Observation: Certification costs more than development

Overall goal: allow evaluation of software

- Covered in the High Assurance ORB Profile
  - Predictable, rigorous base
    - Predictable language subsets
      - IDL
      - target language
  - Keep it simple
    - Reduce code size of ORB run-time
    - Restrict code size of generated code

- Reduce code size of ORB run-time
  - Restrict functionality
    - Example: eliminate shutdown
    - Example: eliminate LocateForward
  - Resolve resources at program initialization - eliminate most/all dynamic behavior:
    - Thread creation.
    - Memory allocation.
    - Runtime symbol resolution.
    - Runtime path resolution (eg. virtual functions.)
    - Transport connections

- Reduce code size of generated code
  - Example: JTRS SCA IDL generates
    - 20K of C++ (ORB*express* for C++),
    - 144K of C++ (TAO)
    - 25K of Java (ORB*express* for Java)
    - 12.5K of Ada (ORB*express* for Ada)
  - Solution approach
    - Restrict IDL types
    - Look for other savings

Developing a High-Assurance ORB Profile – Language Profiles

- Pairs of profiles involved
  - One for IDL
  - One for the target programming language (“safe subset”)
- Plus profile of language mapping
- Target Language Mappings
  - Current languages used for High Assurance
    - Ada – SPARC subset, Ravenscar run-time restrictions
    - C – Motor Industry Software Reliability Association (MISRA) C
    - C++ - not as popular
  - Current “safe subsets” being considered
    - Ada and C++ are the forerunners
    - C would require updating the CORBA C mapping

Language Subsets

- Programming Language Considerations:
  - Late/Dynamic Binding must be avoided. So...
    - Limit or eliminate virtual inheritance/functions.
    - No exceptions allowed.
  - Code must be traceable, especially for certification. So...
    - No templates.
    - Limit/eliminate multiple inheritance.
  - Memory management.
    - IDL types that always have memory constrained limits.

IDL Subset

- IDL Considerations: Limits will be based on ability to map to safe programming language subsets.
  - Different programming languages have different mappings for IDL constructs
    - E.g.,fixed types map to
      - Native type in Ada,
      - ORB generated class in C++
  - Different programming languages should have a common IDL subset to promote interoperability,
    - E.g.,fixed types
      - OK in Ada, not in C++
      - => Eliminate from profile
- Upcoming list is a work in progress

IDL Subset
IDL Data Types

- Octet
- Boolean
- Char
- Enumerated Type
- Short
- Unsigned Short
- Long
- Unsigned Long
- Long Long
- Unsigned Long Long

- Float
- Double
- Array
- Structures
- Strings
- Sequences
- Unions
- Any
- Fixed

Challenge: what about Object References?

Conclusion

- Although significant challenges remain
  - Reducing lines of code
  - Reconciling restrictions of high assurance language subsets
- Significant progress has been made in defining a High Assurance CORBA standard
- It will be possible to define a CORBA subset suitable for High Assurance implementation
  - That retains “interoperability within the subset”
  - That offers advantages of CORBA
    - Portability
    - Time to market
    - Location transparency

For Additional Information

- *http://www.omg.org/cgi-bin/docs?realtime/2005-05-02*
  - latest submission
  - In response to RFP - *realtime/2004-02-24*
- Submitters
  - Objective Interface Systems, Inc.
  - Rockwell Collins, Inc.