Real-time and Embedded CORBA Technology Update for Ada

Brad Balfour
Objective Interface Systems Inc.
Herndon VA
brad.balfour@ois.com
http://www.ois.com/

Disclaimer

- Objective Interface is an ORB vendor
- On the positive side:
  - This talk is based on first hand from experience
  - OIS is part of the group that created the Ada mapping for the CORBA standard
    - We've been in the ORB business since 1995
  - OIS is one of the co-authors of the Real-Time and Fault Tolerant CORBA standards
- On the negative side:
  - We are not "neutral"

Discussions in this talk describe CORBA standard technology only. No product plans, features, or release dates are described or implied.
Brief Review of CORBA Fundamentals (in two minutes)

Changes in the IDL → Ada mapping

Real-Time CORBA

Asynchronous Messaging

Fault Tolerant CORBA

Other New Technologies (“CORBA 3.0”)
**IDL is the Key**

- Provides the glue to attach the object to the software bus
- OMG’s Interface Definition Language (IDL)
  - Specifies interfaces to remote objects
  - Part of the CORBA specification
  - Fully Object-Oriented
- A declarative language mapped to modern programming languages
  - Ada 95, C, C++, Java, Smalltalk, OO Cobol, Eiffel...

**Language Neutral Interface (API)**

**CORBA Development Process**

1. **IDL Source Code**
2. `Idl2ada` or `idl2cpp`
3. Spec/Header Files & Complete Bodies/Definitions
4. Source Code
5. Ada 95/C++ Compiler
7. Linker
8. Client
9. Server
Basic CORBA Runtime Architecture (Diagram)

- Client sends a request to the server
- Server receives request from Client
- Server sends a reply to the Client
- Client gets reply from the Server

Changes in the IDL → Ada mapping

Latest Revision Approved
April 2000
RFP Requirements

- Mandatory - Update the mapping of OMG IDL to the Ada language
  - Reflect additions and changes to the CORBA “core”
  - As of deadline for LOIs (1 June 1999)

- Optional - additional server-side mapping based on delegation

Outline of Mapping Changes and Additions

- Helper Packages
- Value Types
- Value Boxes
- Abstract Interfaces
- Additions and Changes to Pseudo-Interface
- Delegating Servants
- Local Interfaces
- Import
- Repository Identity Declaration
- Exception Clauses for Attributes
Helper Packages

- Interfaces are currently mapped to
  - An interface package containing
    - A “Ref” type derived from CORBA.Object.Ref
    - Mapped “set_attribute” and “get_attribute” subprograms
    - Mapped operation subprograms
    - “To_Ref” function that supports widening and narrowing
    - Type Any support
      - TypeCode constant
      - “From_Any” and “To_Any” functions
  - An implementation package containing
    - An “Object” type derived from PortableServer.Servant
    - Subprograms mapped from IDL attributes and operations

Helper Packages (cont.)

- Revision defines a “.Helper” package – moves in
  - Widening and narrowing support
  - Type Any support
  - Forward reference conversion, if needed

- Rationale
  - Cleaner, more direct interface package
  - Consistent with other language mappings

- This packaging pattern is used in mapping of new constructs (value types, etc.)
Example Interface Mapping

```plaintext
interface Feed {
    typedef long measure;
    attribute measure weight;
};
```

```plaintext
package Feed is
    type Ref is new CORBA.Object.Ref with null record;
    type measure is new CORBA.Long;
    procedure Set_weight (Self : in Ref; To : in measure);
    function Get_weight (Self : in Ref) return measure;
end Feed;
```

```plaintext
package Feed.Helper is
    function To_Ref (From : in CORBA.Object.Ref'CLASS) return Ref;
    function To_Any (From : in Ref) return CORBA.Any;
    function From_Any (From : in CORBA.Any) return Ref;
    TC_Feed : constant CORBA.TypeCode.Object;
end Feed.Helper;
```

```plaintext
package Feed.Impl is
    type Object is new PortableServer.Servant with ...
    function Get_weight (Self : access Object) return measure;
    procedure Set_weight (Self : access Object; To : measure);
end Feed.Impl;
```

Objects by Value

- **IORs and Interfaces** are an OO construct that is always passed (around the network) by *reference*.
- **There is a need/desire** for an OO construct that is always passed (around the network) by *value*.
- **ValueTypes** is the result
  - First order approximation Ada equivalent is: tagged record that can be inherited and extended
  - However, due to additional semantics in the standard, these actually map to a smart pointer
- **Unfortunately, Java RMI interoperability** perturbs the simplicity of the result.
**Inheritance of Base Types**

- CORBA.AbstractBase.Ref refers to CORBA.Impl.Object
- CORBA.Value.Base
- CORBA.Impl.Object
- CORBA.Value.Impl_Base
- PortableServer.Servant
- Interface Printer
  - Printer.Ref
  - Printer.Impl.Object
- Value ExampleA
  - ExampleA.Value_Ref
  - ExampleA.Value_Impl.Object

**Value Types Basic Mapping**

- A value type is mapped to 3 packages
  1.) Value Interface package containing
     - A Value_Ref type derived from CORBA.Value.Base
     - Accessor functions and set procedures for the public state members
     - Functions and procedures mapped from the operations on the value type
     - Functions returning a Value_Ref mapped from the initializers of the value
     - A null value constant.
A value type is mapped to 3 packages

2.) Value implementation package specification containing:
   - An Object type
     - Derived from CORBA::Value::Impl_Base
     - Contains components for each public and private state member
   - An Object_Ptr type - general access to Object type
   - Specifications of functions and procedures mapped from the operations on the value type
   - Factory functions returning an Object_Ptr mapped from the initializers of the value type

3.) Value Helper package containing:
   - TypeCode constants
   - To_Any, From_Any converters
   - A widening converter
Other CORBA 2.3 Changes

- Escaped Identifiers
  - Lexical mapping rules were adjusted to account for escaped identifiers

- Additional operations in CORBA standard pseudo-interfaces
  - These have now been mapped into Ada 95 according to the rules for mapping pseudo-interfaces

Delegating Servants

- Current servant mapping using inheritance
  - Delegation is an alternative design approach (known as the “tie” approach in the C++ mapping)

- Additional implementation delegation package may be generated for each interface. Generic package
  - With formal parameters specifying
    - Type to be wrapped
    - Functions and procedures with the same signature as
      - Mapped attribute accessors and setters
      - Mapped operations
      - Including signatures of all inherited interfaces
  - Defines
    - the servant type - Object
    - Create function
**Local Interfaces (from CORBA Components Specification)**

- Mapping of local interfaces differs from mapping of "unconstrained" interfaces in only minor ways
  - Interface package
    - Reference type is named "Local_Ref"
    - Semantics of CORBA.Object operations altered
  - Implementation package
    - Object type inherits from CORBA.Local.Object
  - Helper package
    - No need for type any support
  - LocalObject - mapped to type Object in CORBA.Local package

**Import (from CORBA Components Specification)**

- New statement introduces unit visibility rules into IDL
- Visibility rules from IDL import statements are incompatible with visibility rules for Ada’s "with" statements
  - No explicit mapping rules for import
  - Instead, general visibility mapping rules
- Visibility mapping rules
  - IDL visibility according to IDL visibility rules
  - Generate Ada "with" statements to reflect visibility required by Ada compilation
Other CORBA Components based Changes

- **Repository Identity Declaration**
  - Explicit typeID and typePrefix statements replace pragma
  - Affects string values used in TypeCode constants only

- **Exception clauses for attributes**
  - Adds "getRaises" and "setRaises" clauses to attributes
  - No explicit mapping needed for Ada

Real-Time CORBA
Definition of Real-Time for this Discussion

- A real-time system is one which ensures time constraints are met.
- A real-time system produces a value to the overall system which is a function of time.

<table>
<thead>
<tr>
<th>Value</th>
<th>Non-Real-Time System</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Hard Real-Time System</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Soft Real-Time System</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Time</td>
</tr>
</tbody>
</table>

A Truism

- Real-Time ≠ Real Fast!
- However, that being said, people usually want real-time systems to be very fast!
CORBA is an important part of the solution!

Why Use CORBA in a Real-Time System? [1]

- ... for flexibility
  - Many real-time systems must run on multiple computers & are written in mixed languages
  - CORBA solves many real-time system integration problems if the ORB is:
    - fast
    - predictable
    - small
    - has native support for embedded transports

CORBA doesn’t have to be slow, bulky or indeterminate

- ... to save money!
  - Future system changes are much easier
    - change CPU type
    - change O/S
    - change transport (bus or network)
  - Upgrade components one at a time
    - New components can interoperate with old components

Ingredients of a Real-Time ORB

- Predictable ORB infrastructure
  - Predictable memory management, etc.
  - Use of predictable, bounded algorithms
  - Analysis and documentation of behavior

- Priority Propagation & Distributed Priority Inheritance

- Predictable transport (i.e. not TCP)
  - Native ATM (ANI)  Reflective Memory  FDDI
  - FireWire (IEEE 1394)  Shared Memory  VME/PCI

- ORB must understand and use deterministic transports & Quality of Service (QoS) metrics
- ORB must allow designer to control QoS and map them from client → network → server
**CORBA Priority Propagation**

- **Windows NT:** Client Thread priority 7
- **OS priority 7 mapped to CORBA PRIORITY 28**
- **CORBA PRIORITY 28 passed in service context**
- **RTOS:** Server Thread executes at priority 64
- **CORBA PRIORITY 28 mapped to RTOS priority 64**

---

**History and Standardization**

- **CORBA Evolution:**
  - Introduced in 1991 by the Object Management Group (OMG)
  - OMG is an international consortium of over 800 companies
    - Users, Suppliers, Academics, Government
    - Industry spends 30M each year to participate in OMG
  - CORBA Specification continues to evolve

- **Real-Time CORBA Evolution:**
  - Effort to extend the CORBA specification to meet real-time requirements
  - Produced over a period of 1 ½ years
  - Based on experience of vendors and users with existing implementations
  - Real-Time CORBA 1.0 standard accepted 7/99
Overview of Real-Time CORBA Standard

- The basic concepts described earlier are embodied in the Real-Time CORBA standard
- In addition to the mandatory elements, ORB vendors can/will offer additional capabilities
- Some specific additional aspects of the Real-Time CORBA standard
  - Objective
  - Threads and Priorities
  - Threadpools
  - Mutex
  - Priority Banded Connections

Objective

- End-to-End Predictability =
  - Respecting thread priorities between client and server for resolving resource contention during the processing of CORBA invocations;
  - Bounding the duration of thread priority inversions during end-to-end processing;
  - Bounding the latencies of operation invocations.

- Support both “Hard” and “Soft” Real-Time Applications
  - Tools in the standard for predictability for Hard Real-Time Systems
  - Tools in the standard for resource control and management for Soft Real-Time systems where these can be scarce
Threads and Priorities

- "Thread" is the schedulable entity
- POSIX Threads
- Native (RTOS) Priorities v.s.
- CORBA Priorities
  - 0 .. 32767
  - 32767 is the "highest" priority
  - Universal priority scheme
  - User installable mapping functions

Threadpools

- Threadpool abstraction is used to manage threads on server-side of real-time CORBA ORB
  - Pre-allocation, partitioning, bounding usage: predictability
- Threadpool parameters
  - Number of static threads
  - Dynamic thread limit
    - 0 = no limit. Same value as static = no dynamic threads
  - Thread stacksize
  - Default thread priority
    - Thread priority will change as required
- Laned Threadpools also exist
Mutex

- The Essential Abstraction for Resource Protection
- Context for Priority Inheritance Discussions
- Makes the mutex used by the ORB available to the application developer

Priority Banded Connections

- Multiple connections, to reduce priority inversion
  - Each connection handling different priority invocations
- Banding
  - Each connection may represent a range of priorities, to allow resources to be traded off against limited inversion
  - May have different ranges in each band, including range of 1
Current CORBA supports only Synchronous calls
- Oneway isn’t really what is wanted for Asynchronous calls
- It is limited to in parameters only and no exceptions
  - AMI does define new, additional semantics to oneway operations
- There were no delivery guarantees

AMI defines two new Asynchronous modes:
- Callback
  - Caller hands callee an IOR for a callback interface
  - Callee invokes callback interface when reply
  - Callback interface is automatically determined from IDL (ReplyHandler)
- Polling
  - Invocation returns an IOR to an interface which may be periodically polled for the pending result(s)
AMI Concepts [2]

- Message Quality of Service Characteristics
  - Delivery Characteristics including:
    - Birth and Expiry times for request and reply
    - Reliability
    - Scope of synchronization with target
  - Server Side Queue Management or Ordering:
    - Temporal based
    - Priority based

Fault Tolerant CORBA
Fault Tolerant CORBA Concepts

- Wide range of systems exist that might want/need this functionality
  - Compromises have been made in creating the FT CORBA specification
  - Once experience has been gained, revisions might happen

- Objectives:
  - Provide robust support for applications that require a high level of reliability
    - No single point of failure
  - Depends on entity redundancy, fault detection and recovery
    - Redundancy is based on Object replication and groups of Objects

FT CORBA Concepts [2]

- Objectives (continued):
  - Range of fault tolerance strategies, including:
    - Request retry
    - Redirection to an alternative server
    - Passive (primary/backup) replication and
    - Active replication (which provides more rapid recovery from faults)
  - Developer control of
    - Creation of replicas
    - Automation of replication of state
  - Support of Fault detection, notification and analysis
  - Minimal modification of programs
  - Transparency of replication and faults
  - No Single Point of Failure
Other New Technologies ("CORBA 3.0")

Some Highlights:
- Support for distributed components
- New messaging support
- Quality of Service (QoS) features (covered previously)
  - Invocation ordering and time outs
  - Real-Time, Minimum and Fault Tolerant CORBA
- Enhanced Java and Internet integration
**CORBA Components**

- Specifies a framework for the development of “plug and play” CORBA objects:
  - Encapsulates the creation, lifecycle, methods, and events for a single object
  - Meant to decrease learning curve for development and us of CORBA clients and servers
  - Scripting language
  - Multiple interfaces for a single object
  - Pass Objects by Value

- Leverage and extend the existing Enterprise Java Beans specification for Java

**Java and Internet Integration**

- Java-to-IDL Reverse Mapping
  - Unifies Java Remote Machine Interface (RMI) and CORBA IIOP
  - We are aware of a background effort to create an Ada to IDL mapping

- Java objects become CORBA-accessible:
  - Write JAVA server object
  - Automatically generate the interface in OMG IDL
  - Access its methods from a CORBA client written in any language
  - No longer a single-language distributed environment

- Firewall specification – allows secure use of CORBA with the internet
Minimum CORBA

- New specification for embedded CORBA environment
- Based on the idea that an ORB “must have at least” these features to comply
- Does not discuss memory footprint or dynamic memory constraints
- Designed for fixed, predictable systems
  - Eliminates dynamic aspects of CORBA
    - DII, DSI, Dynamic Any
    - Interface Repository
    - Some POA features and policies
  - Supports ALL OMG IDL types