PSO CORBA talk

Andrew Watson

Abstract

These slides were prepared for a presentation to a symposium organised by HP’s PSO. The intention is to give some background on why CORBA is important (both from a technical and a marketing point of view), and to summarise its key concepts.

Planned running time is about 50 minutes.
Common Object Request Broker Architecture

Andrew Watson
APM
ajw@ansa.co.uk
Why Objects?

• Don’t just jump on the bandwagon!

• There’s no single agreed set of features in “an object system”
  - Every object-based language/database/infrastructure has a different set

• Key concept is encapsulation [I assert]
  - Precise control over how object interacts with its clients -> modularity
  - Modularity is the key to developing application components separately
  - ..... to distributing application components across several machines
  - ..... to testing and verifying components separately
  - ..... to reusing the same components in many applications

• Other OO concepts (e.g. inheritance) are secondary
Market expectations

% of companies adopting object technology

OT in most companies

OT in more than 25%

All U.S. companies

My company
Whence came OMG?

- Non-profit organisation founded April 1989 to promote unified market for OO products
  - US-based but international in scope
  - Many vendors, some end-users, one or two researchers

- Mission: To develop a single architecture, using object technology, for distributed application integration
  - Provides reusability of components
  - Interoperability and portability of applications
  - Based on commercially-available software
  - “It should be as easy to plug a computer into the global information network as it is to plug it into mains power network”
# Org Chart

**OMG Board**
- Chris Stone (OMG)

**Technical Committee**
- Richard Soley (OMG)

## Subcommittees

- **Reference Model**
  - Mike Mathews (HP)

- **Policies & Procedures**
  - Richard Soley (OMG)
  - Requirements
  - Geoff Lewis (SunSoft)

- **Object Model**
  - Pat O'Brien (Object design)

## Ad-hoc working group

- Richard Soley (OMG)

## SIGs

- **End-user Requirements**
  - Pat Davis (Boeing)

- **OO databases**
  - Jacob Stein (Servio)

- **Parallel Object systems**
  - Gene Pierce (NCR)

- **Analysis & Design**
  - Andrew Hutt (ICL)

- **Class Libraries**
  - Tayloe Stansbury (Borland)

- **Smalltalk**
  - Duane Bay (ParcPlace)

## Task Forces

- **Object Request Broker 2**
  - Andrew Watson (APM)

- **Object Model Revision**
  - Bill Kent (HP)

- **Object Services**
  - G Lewis & W Andreas

- **ORB Revision**
  - Carl Soeder (HP)
Object Management Architecture

Application Objects

Common Facilities

Object Request Broker

Object Services
Populating the Framework

- Task Force is formed, issues Request for Information (RFI)
  - Everyone invited to respond with whatever material they think relevant

- TF studies responses, decides strategy
  - Possibly produces Roadmap or Architecture document

- TF issues Request for Proposals (RFP)
  - Members supply specifications of technology to fill stated requirement

- TF recommends a single response to TC, which recommends to board

- Task Force dissolved
Object Request Broker

- OMG’s first RFP cycle
  - RFI closed Aug. 1990 (8 responses), RFP closed Dec. 1990 (10 LOIs)
- Seven proposals presented March 1991
  - APM, Bull, DEC, DSET, HP/Sun, Hyperdesk, NCR/ODI
- Two merged submissions by demonstrations in May 1991
  - HP/Sun/NCR/ODI & Hyperdesk/DEC
- “90 day” team formed, presented merged proposal (CORBA) in September 1991
- Proposal accepted October 1991
Key CORBA concepts

• Object
  - “Classical” object model: each request directed to a particular object

• Object reference
  - “Handle” used by client(s) to make invocation on service-provider
  - Opaque (i.e. no handle equality test - see Powell’s paper)
  - May be passed as request parameter

• Request
  - Operation name + target object ref + zero or more parameters
  - Optional “request context” (to “pass additional data about the request”)
  - Outcome: results or an exception
  - Parameters may be IN, OUT or IN/OUT
Key concepts (cont.)

- **Interface**
  - Set of operation signatures
  - Identifies requests that can be made on object satisfying that interface
  - Interface = abstract type (abstract class in C++ speak)

- **Operation signature**
  - Operation name + parameter types & directions + exception spec + context spec + semantics (at-most once vs. one-way best-effort)
  - cf. C/C++ function prototype

- **Interface Definition Language (IDL)**
  - Written interface definitions
  - DOESN’T SPECIFY IMPLEMENTATION (despite looking like C++)
Interface repository

IDL & Stubs

Server sources

Implementation skeleton

Server implementation def

Implementation

IDL definitions

IDL preprocessing

Client stubs

Client source

Compile & link

Client program

Impl Repository
Key concepts (cont.)

- Interfaces can be derived from other interfaces by extension
  - Creates a subtype (since redefinition of operations not permitted and IDL has no self-reference)
  - An interface can be derived from multiple ancestors - but illegal if operation names conflict
  - CORBA calls this derivation “inheritance” (sic)
  - Merely saves the effort of writing out the new definition in full [I assert]
  - Interface B inherits from interface A is a sufficient, but not a necessary, condition for object with interface B to be used with client that expects one with interface A [I assert]
Interface inheritance & subtyping

interface A {void f {in float x}}
interface B {long g {in long x}}
interface C: B, A {void h {in long x}}

• **Interface C** is completely equivalent to:

  interface C: {void f {in float x}
  long g {in long x}
  void h {in long x}}

• **C** (either derivation) is a subtype of both **A** and **B**
  - object with interface C may be substituted wherever clients expect one with interfaces A or B
Key concepts (cont.)

- **Object Adaptor**
  - “Glue” that passes incoming invocation into server object
  - Includes implementation skeleton generated from IDL
  - Only Basic Object Adaptor (BOA) specified by CORBA

- **Dynamic Invocation Interface (DII)**
  - Client’s alternative to invoking object’s operations via stubs
  - Req’d only where Lisp programmer would use ‘eval’ (i.e. almost never)
  - Complicated and error-prone: use only if unavoidable

- **Interface Repository**
  - Provides ability to find interface of arbitrary object at run-time (see above)
Why the DII isn’t for you

• For client to make request using stub, target object need only have operation with the right name and signature
  - No need to know the implementation of the object
  - No need for server’s IDL to be identical to (or derived from) the client’s
  - No need for server even to exist until client actually makes request
  - Therefore, usually no need for DII: stub-based client can do the job

• DII forces programmer to build parameter lists etc “by hand”
  - No check that he got it right until run-time
  - Large API, lots of code
  - Only necessary if writing language interpreter, object browser etc

• “If you can write it in C++, you can use a stub”
Invocation semantics

- **CORBA invocations are request/response ("synchronous")**
  - Therefore your CORBA implementation must have threads for clients

- **“One way” invocations also available**
  - Unreliable
  - Correct implementation of CORBA could throw away every 10th one-way invocation ... or every second one ... or all of them
  - ... and before you say it, yes this isn’t much use to you!
  - One-way provided for access to the comms, to build blast protocols etc
  - Queued message delivery positioned as object service

- **Deferred synchronous also available via DII**
Availability

• One or two companies have ORBs available today (9/93)
• Several are in beta test
• Many companies (100?) are working on ORBs and related products
• Compliance issue
Future developments

• CORBA = ORB 1.1

• ORB 1.2 Revision Task Force working on clarifying some obscurities in 1.1

• ORB 2.0 Task Force about to issue RFPs
  - Interoperability & Initialisation - initial responses due March 94
  - Interface repository - initial responses due May 1994
Equality

• Must distinguish (at least) two kinds of equality (sameness):
  - Same contents (equality of information an object represents)
  - Same container (often detected via equality of object references)
  - “Same container” implies, but is not implied by, “same contents”
  - “Same pointer” implies, but is not implied by, “same container”

• Application programmers care about information, not representation
  - Example: CLtL explains built-in data types in terms of abstract data
  - Lisp \texttt{equal} can be explained in these terms, \texttt{eq} cannot; \texttt{equal} is about equality of abstractions, \texttt{eq} about “same pointer” (CLtL2, p103-104)
  - Numbers are defined to be \texttt{equal} to themselves, but fixnums “may or may not be” \texttt{eq} to themselves
Equality (cont.)

• If “same pointer” test fast and easy, it can be used to short-circuit “same contents” test
  - eq is one or two instructions in a good CL implementation
  - Experienced programmers in Lisp, ST-80 design their data types so that “same contents” (equal) does imply “same pointer” (eq)
  - Makes application equality tests efficient, at some expense when setting up data representation (e.g. interning symbols)

• “Same pointer” test (eq) is the “same container” test only if pointer(x) = copy(pointer(x)) always
  - Easy & natural in non-distributed infrastructure
  - Useful distributed infrastructure designs may not maintain this invariant
Same container vs. same pointer

Client A

Client B
Same container vs. same pointer (cont.)
Same container vs. same pointer (cont.)
Equality - conclusions

• “Same container” test in a distributed environment may be as expensive as calling application-defined “equal” method

• In any case, placing “same container” test in the application programmer’s model is a (weak) violation of encapsulation
  - Clients’ correct operation could depend on equality of operation results
  - Apparently-innocuous changes to object implementation would cause clients to behave differently

• “Same pointer” and “same container” tests should be part of the implementor’s (engineering) view of an ORB, but absent from the application programmer’s (computational) view
Type issues

- Anyone want to discuss F-bounded quantification?