CORBA

Steve Goddard
goddard@cse.unl.edu

http://www.cse.unl.edu/~goddard/Courses/CSCE855
CORBA

- Object Management Group
  » consortium that created CORBA and related technologies
    - Sun, HP, Oracle etc. (700+ members)
    - Microsoft is not a member of the consortium
    - why?
  » OMG creates the specs…
    - that can then be turned into products by various vendors
    - if vendors stick to specs,
    - should be able to run with any CORBA implementation

The CORBA Vision

- Define a way to divide application logic among objects distributed over a network
  » using standards
    - allow any operating system, machine, language
    - to be involved on creating a component
  » as long as the standard is complied with
    - objects should be able to work together
    - the death of language religious wars!!
- Achieved through a distributed object architecture
  » i.e. object-oriented + distributed heterogeneous objects
The CORBA Vision (cont.)

- Separation of interface and implementation
  » create a well-defined interface to an object
    « parameters passed to the object
    « data returned
    « define the form (type, structure) of parameters and return values
  » given a well-defined interface, implementation doesn’t matter
    « can change the implementation without impacting rest of the application (O-O concept)
    « doesn’t have to be programmed in the same language
    « …or reside on the same machine

Object Management Architecture (OMA)

- OMA Components
  » CORBA
    « connects objects, not applications
  » CORBA Services
    « low-level functionality needed by objects, such as security, time, persistence, transaction, and naming services.
  » CORBA Facilities
    « user-level facilities, such as document management, help facilities, and system administration, provided to applications.
CORBA Components

- Object Request Broker (ORB)
- OMG Interface Definition Language (IDL)
- Language Mappings
- Interface Repository (IR)
- Dynamic Invocation Interface (DII)
- Object Adapters (OA)
- Inter-ORB Protocols (e.g. IIOP)

Object Request Broker

- Object Request Broker (ORB)
  » middleware that establishes client-server relationship between objects
  » links object requests to implementations
- Using an ORB
  » client requests a service
  » ORB intercepts the call and finds an object that can implement the request
    ❖ do necessary translation in passing parameters, getting results
  » client acts as though its calling a method in the system
    ❖ client doesn’t have to know details
ORB (cont.)

- ORB location services
  - ORB is free to choose an implementation
  - if a host is down, choose another object
  - that satisfies the request
- Requires that objects are written with certain requirements
  - global naming scheme (object reference)
  - registration of services
  - each object tells what services it provides

Interface Definition Language (IDL)

- IDL is a language used to define interfaces
  - objects must then implement the interface faithfully
  - clients only see the IDL interface definition
- IDL is a declarative language
  - meaning it only allows declarative statements
  - no conditionals, loops, etc.
  - supports most basic types (short, long, float, etc.)
  - also some derived types (string, structures, arrays, etc.)
Most important - creates an interface

» interface defined as a collection of specifications that
define an API set

» interfaces contained in a module

❖ modules define a local name space
❖ therefore global name space is only concerned with module names
❖ somewhat like packages in Java

» operation is the specification of a method call

❖ signature (operation name, return type, parameter list)

» exceptions can be raised (raises)

❖ handled by client code

» can also declare a context for an operation

❖ name-value pairs similar to UNIX or DOS environment variables

IDL Example

```idl
module <module name> {   <user-defined type declarations>;   <constant declarations>;   <exception declarations>;   interface <interface-name> [:parent-interface-name] {   <user-defined type declarations>;      <constant declarations>;      <exception declarations>;      <attribute declarations>;      [operation-type] <operation-name> (<parameter list>) [raises exception_name, ...] [context (context1, ...)];      [operation-type] <operation-name> (<parameter list>) [raises exception_name, ...] [context (context1, ...)];   }   interface <interface-name> [:parent-interface-name] ... }
```
**Language Mappings**

- Map IDL to language features
  - for example in C++
  - IDL Module → C++ namespaces
  - IDL interface → C++ class
  - IDL char → C++ char
  - IDL octet → C++ unsigned char
- OMG defines standard language mappings
  - C, C++, Smalltalk, Ada, COBOL, Java
  - other independent mappings (e.g. Perl, Eiffel, Modula-3)
- Mappings are embedded in IDL compilers
  - each language has a specific IDL compiler

**IDL Compilers**

- IDL compilers create stubs
  - referred to as stubs on the client side
  - skeletons on the server side
  - clients interface with stubs
  - to communicate with ORB
  - orbs interface with skeleton
  - to communicate with server
- Known as “static Method Invocation”
  - stubs and skeletons directly linked into code
**IDL Interface to ORBs**

- IDLs define stubs similar to RMI/RPC
  - stubs marshal parameters to/from the ORB

**Interfacing to an ORB**

- Multiple ways to interface with ORBS
  - static IDL stubs
    - as in previous slides
  - dynamic invocation

![Diagram of IDL Interface to ORBs](image)

![Diagram of Interfacing to an ORB](image)
**Interfacing to an ORB**

- Dynamic invocation
  - Dynamic Invocation Interface (DII)
  - Interface Repository (IR) stores
    - interfaces
    - references
    - objects’ inheritance hierarchy and all operations it supports
  - ORB matches dynamic request to a DSI
    - protocol for client to get object reference, interface

- Dynamic Skeleton Interface (DSI)
  - equivalent of DII for server-side
  - ORB access servers without static skeletons

**Interface Repository (IR)**

- Allows clients to programatically discover type information at run-time
  - primary utility is supporting dynamic method invocations
- IR stores object
  - interface definitions
  - inheritance hierarchy (graph)
  - all operations supported
- Services of IR can be accessed through
  - Standard IR IDL interface
  - Custom libraries provided by ORB vendor
Object Adapters (OA)
- Responsible for object activation transparency
  - intermediate layer that connects the ORB and the object implementation
  - different OA for each supported programming language
- Main duties
  - object registration
  - generation of object references
  - object activation
  - activation of server process
  - request handling

ORB-to-ORB Communication
Inter-ORB Protocols
- Direct ORB-to-ORB communication
  - ORBs are in same domain (common IDL type systems)
  - General Inter-ORB Protocol (GIOP)
    - Internet Inter-ORB Protocol (IIOP)
- Bridge-based communication
  - ORBs from different domains
  - Environment-Specific Inter-ORB Protocols (ESIOPs)
    - Distributed Computing Environment Common Inter-ORB Protocol (DCE CIOP)
      - allows for easy integration of CORBA and DCE applications
Java vs. CORBA

- Both have similar goals
  » creating software objects that can run “anywhere”
  » CORBA also adds a strong distributed theme
- Achieved in different ways
  » Java: architecture neutrality
    ❖ JRE allows code to run on any system (with a JRE)
    ❖ model is a single monolithic application
  » CORBA: transparent communication
    ❖ invocation of objects is transparent
    ❖ model is a set of (potentially distributed) objects working together to create an object
    ❖ platform dependency is allowed, but communication transparency hides dependencies

Java vs. CORBA (cont.)

- Language Dependencies
  » Java: one language
  » CORBA: bridge differences in languages
    ❖ can run into ORB dependence (commercial ORBs have differences)
- Distributed Services
  » Java: no explicit support for distributed objects
    ❖ although RMI is a start
  » CORBA: model is based on distributed objects
- Scale of systems
  » Java: works for large or small apps
  » CORBA: too much infrastructure for small apps