Software Architecture in Practice

Chapter 9: CORBA - A Case Study of an Industry-Standard Infrastructure

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Sponsored by the U.S. Department of Defense

Objectives

This lecture will enable students to

- **understand how the business needs of a consortium led to a particular architecture**
- **understand how the software architecture of CORBA allows program interoperability**

Business Motivations

The Object Management Group (OMG) was chartered in 1989 to

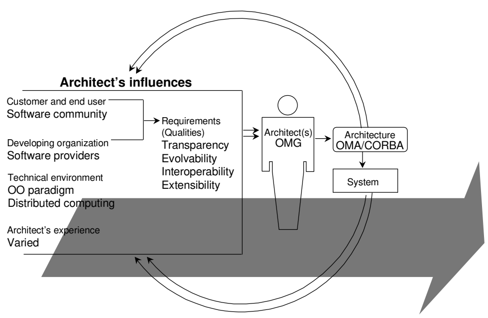
- **provide a common software platform in which distributed object-oriented programs can be communicated and interact**
- **create a component-based marketplace for vendors to compete with Microsoft's Object Linking and Embedding (OLE)**

Business Model

OMG included most non-Microsoft vendors (IBM, Digital, Sun, HP).

They specified the object-management architecture (OMA) and common object request broker architecture (CORBA).

ABC for the OMA



Quality Goals of the OMA

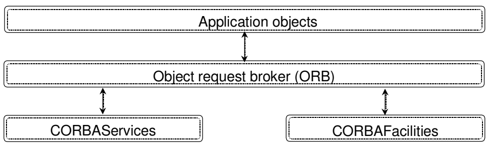
Interoperability: support interoperation of objects implemented by different vendors

Evolvability: be responsive to development of new object and distributed systems technologies

Extensibility: provide a core set of component integration servers but be extensible to support market niches served by independent software vendors

Buildability: be readily constructed with available technology (time to market)

OMA



Components of OMA -1

CORBA is an architecture specification. It defines a standard architecture and services for OMA-compliant object request brokers (ORBs).

The object request broker (ORB) component of CORBA provides communication services, e.g., remote method invocation, event channels.

Components of OMA -2

CORBAServices: provides universally useful services
- **naming service**
- **event service**
- **persistent-object services**
- **transaction services**
- **concurrency-control services**

CORBAFacilities: provides higher level and domain-specific services (not yet well defined)

How OMA Supports Qualities - 1

Interoperability

- **interface definition language (IDL) used as common basis for object interfaces**
- **Internet, inter-orb protocol (IIOP) provides standard “on the wire” representation for request/responses**

Evolvability: specification partitioned into separately evolvable subcategories

How OMA Supports Qualities - 2

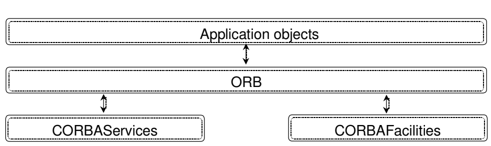
Extensibility

- **CORBA core and CORBAServices provide stable core**
- **CORBAFacilities allow niche-specific extensions**

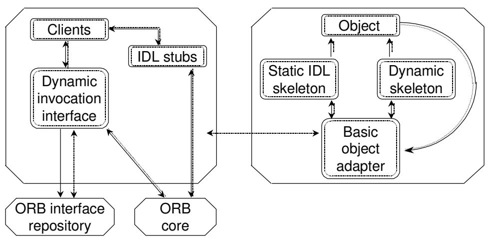
Buildability: selected specifications must have commercial-quality implementations within one year of acceptance

CORBA

Consists of ORB and various interface mechanisms that allow objects to interact via the ORB



CORBA Architecture



Carnegie Mellon University
Software Engineering Institute

ORB Features

ORB interface
- **identical for all ORB implementations**
- **provides instance binding and object identification services**

Interface definition language stub/skeleton
- **specific to each object type**
- **stub called by client**
- **skeleton calls implementation**
- **specified statically**

Dynamic invocation: methods and parameters determined at runtime

Carnegie Mellon University
Software Engineering Institute

Interface Definition Language

All object interfaces are defined in IDL.

IDL allows
- **language independence of objects**
- **“wrapping” of legacy systems**
- **explicit publication of interfaces**
- **dependency on only public properties**

IDL syntax was chosen to resemble that of C++.

No overloading of method names is allowed to simplify bindings to existing languages.

Simple Bank Abstraction in IDL

```
interface Account {
        readonly attribute string name;
        readonly attribute float balance;
        void deposit (in float amount);
        void withdraw (in float amount);
 };

 interface CheckingAccount : Account {
        readonly attribute float overdraft_limit;
        void order_new_checks();
 };
 interface SavingsAccount : Account{
        float annual_interest();
 };
```


Object Adaptors

Object adaptors provide the following services to object implementations.

- **generation and interpretation of object references**
- **mapping of object references to object implementations**
- **activation and deactivation of object implementations**
- **invocation of methods on object implementations**
- **registration of object implementations**
- **access control for object implementations**

Basic Object Adaptor

Every CORBA-compliant implementation must have a special kind of object adaptor, basic object adaptor (BOA).

The BOA uses the remote procedure call (RPC) model to implement objects, where objects are implemented as server processes.

Specifications for BOA were published by OMG, ensuring availability in the marketplace.

How Do Systems Built with Different ORBS Interoperate ?

CORBA version 1
- **did not address the issue of interoperation among different clients of different ORB vendors**
- **allowed faster time to market for ORBs**

CORBA version 2 (1995) specifies interoperability.

Because of the delay in specification, the system must introduce interoperability between vendors without affecting clients of existing ORBs.

CORBA 2 Interoperability -1

Mandated Internet inter-ORB protocol (IIOP)

Uses concept of "bridge" protocol adaptors between objects

All ORBs must support "half bridges" that convert object requests into IIOP (or IIOP into object requests).

Some ORBs use IIOP as a native protocol.

CORBA 2 Interoperability -2

Client request
- **goes into ORB1**
- **converted via half bridge to IIOP**
- **sent to ORB2**
- **converted via half bridge from IIOP to object implementation**

CORBA/WWW Comparison

Both WWW and CORBA address interoperability.

WWW
- based on data model
- relies totally on protocol
- has no central control
- provides runtime linkage of one page to another

CORBA
- based on software objects
- protocols are adapted both within and between ORBs
- has central “software bus” with registration
- interfaces are linked at specification time or later

Lecture Summary

CORBA was designed to allow interoperation, fast time to market, and evolvability.

CORBA provides basic services for object invocation and referencing.

IDL allows objects written in multiple languages to be bound to ORB and provides common interfaces independent of ORBs.

IIOP allows different ORBs to interoperate.

Discussion Questions - 1

1. The discussion of CORBA in this lecture centers around programmatic integration of applications. The discussion of the World Wide Web (WWW) in the previous lecture centered around data integration of applications. What is the major difference between these two approaches? What are the strengths and weaknesses of each? What would be involved in moving CORBA toward a datacentric view or the WWW to a programmatic view?

Discussion Questions -2

2. **What are the issues involved in using CORBA to obtain real-time performance? To put it another way, what would be involved in maintaining real-time quality of service in a distributed object world (as was required in many of the case studies in this book)?**