YU-INFO-2000
Advances in Infrastructure for E-Business on the Internet
Kopaonik, March 27 – 31, 2000

Internet and Object-oriented Design Methods

Andrea Domenici
SSSUP S. Anna, Pisa
andrea@sssup.it
http://www.ing.unipi.it/~d8651

Contents

1 Introduction

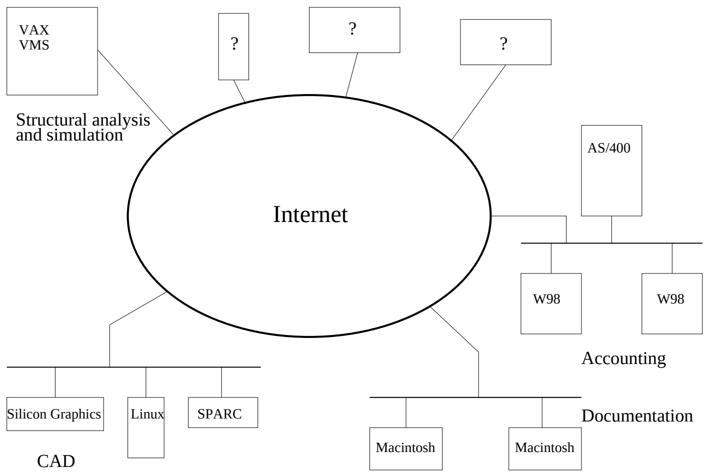
This talk introduces the basic concepts of the *Common Object Request Broker Architecture*. These concepts are tools that designers can use to meet the requirements of large scale distributed applications, such as those that are made possible by the existence of a world-scale Internet.

2 Internet and distributed systems

Like Proteus in the ancient myth, the Internet can show innumerable shapes and faces to its users. Designers must cope with ever evolving needs and demands, and deliver open-ended, adjustable and extensible systems.

Slide 1

Internet and Distributed Systems



Several services or activities are supported by clusters of computers connected by local networks. Each service or activity needs to interact with the others. Each service or activity may belong to a different organization. New suppliers and consumers of services show up on the Net in time.

Slide 2

How Do We Deal With. . . ?

- **Scalability**: distribution, implementation replacement.
- **Reusability**: “open/closed” principle, libraries, integration of legacy systems, *application* reuse.
- **Platform independence**: information hiding, standard protocols and interfaces.
- **Programming language independence**: common *interface* language (trade language).
- And more. . .

The bottom line is: a **large scale OO approach**.

3 A large scale object-oriented approach

In this section we sketch the evolution of the design/programming approach to distributed systems in three major phases.

Marshaling and *demarshaling* refer to the encoding and decoding, respectively, of application data into and from a format suitable to transmission over a communication channel, as specified by a transport protocol (e.g., TCP/IP).

Slide 3

A Large Scale OO Approach

In traditional distributed system design, much work is devoted to such issues as data representations and transport protocols.

In an OO approach for large scale distributed systems emphasis is on *application-centered* object interaction. We will describe some aspects of the CORBA architecture, that provides:

- An object-oriented computational model.
- A common interface language.
- A networking and distribution infrastructure.
- Standard component interface libraries.

The CORBA specifications are produced and maintained by the *Object Management Group* (OMG) (`www.omg.org`).

Slide 4

The Object Management Group

Founded in April 1989 *to create a component-based software marketplace by hastening the introduction of standardized object software.*

- Establishment of industry guidelines.
- Detailed object management specifications.
- Common framework for application development.
- Now includes over 800 members, including: 3Com Corporation, American Airlines, Canon Inc., Data General, Hewlett-Packard, IBM, Microsoft, Philips Telecommunications N.V., Sun Microsystems, Unisys Corporation.

4 The Common Object Request Broker Architecture

4.1 Computation model

Slide 5

The CORBA Model

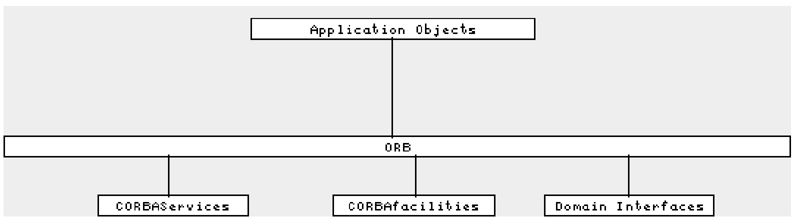
- Application developers design *application* (or *business*) *objects.*
- Objects interact according to the *client-server* paradigm.
- Objects offer services defined by object *interfaces.*
- Objects are accessed through *object references.* The operations defined by an object interface are called through its reference, and they are independent of the object's physical location and the communication infrastructure.

4.2 The Object Management Architecture

The *Object Management Architecture* builds upon the core architecture (the ORB and its mechanisms and interfaces) to provide a comprehensive framework of higher-level services.

Slide 6

The Object Management Architecture (OMA)



- The *CORBA Services* are standard library objects that offer utility sevices beyond the basic ones offered by the ORB (see Slide 7).
- The *CORBA Facilities* are standard library objects that offer higher level utility services, i.e., services more application-oriented than CORBA Services (see Slide 10).

- The *Domain Interfaces* are oriented to specific application domains (see Slide 10).

Slide 7

CORBA Services

The *CORBA Services* are standard library objects that offer utility sevices beyond the basic ones offered by the ORB.

- **Naming**: finding objects by name.
- **Trading**: finding objects by services offered.
- **Life cycle**: creating, deleting, copying and moving objects.
- **Events**: notifying events between objects.
- **Object persistency**: retaining and managing the persistent state of objects.
- **Transactions**: Atomic, Consistent, Isolated, Durable units of work.
- And more...

Slide 8

The Naming Service

- An object is identified by a *marker* (local to its server), the name of the server, and the host where the server runs.
- If this information is known, it can be used to obtain the object's reference. However, this method is inflexible and creates unnecessary dependencies.
- A *Naming Service* allows clients to find an object reference without relying on server and host information.
- With the Naming Service, objects are identified by compound logical names (similar to pathnames in hierarchical file systems).
- Not all objects need to be registered in the Naming Service!

Slide 9

The Trading Service

While the Naming Service allows clients to be decoupled from knowledge of object locations, the *Trading Service* allows them to look for objects offering the required services.

- An object may register itself to the Trading Service, along with a description of its services.
- An object service is described by a *service type name*, an IDL interface, a set of *base service types*, and a set of *properties*, i.e., name/value pairs.
- A client may query the Trading Service for objects offering services of a given type (or derived from a given type), and with properties satisfying given *constraints*, expressed in a simple *constraint language*.

Slide 10

CORBAfacilities

- The Horizontal CORBAfacilities offer such general purpose services as printing or system management.
- The Vertical CORBAfacilities (aka Domain Interfaces) are oriented to specific application domains, such as Electronic Commerce, Manufacturing, and Communications.
- For example, the *CORBAMed* set of specifications defines interfaces for a *Person Identification Service* and a *Lexicon Query Service*, aimed at applications in the medical care domain.

Slide 11

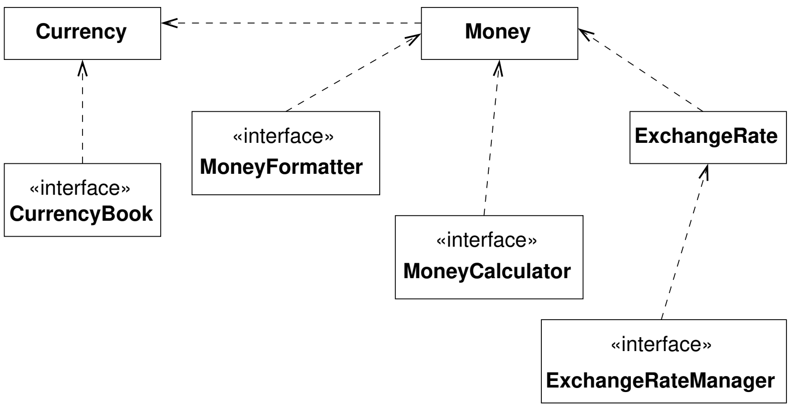
The CORBAFinance specifications

These specs address the basic concepts that financial applications deal with:

- **Currency**: name, symbol, fractions…
- **Money**: amounts of currency, and related operations.
- **Exchange rates**: conversions between currencies.

Slide 12

CORBAFinance specifications



Slide 13

Currency IDL

```
value Currency {
   wstring    getMnemonic()         raises(FbcException);
   wstring    getName()             raises(FbcException);
   wstring    getFractionName()     raises(FbcException);
   wstring    getSymbol()           raises(FbcException);
   CBO::DTime getIntroductionDate() raises(FbcException);
   CBO::DTime getExpirationDate()   raises(FbcException);
   // ...
}
```

Slide 14

MoneyCalculator IDL

```
interface MoneyCalculator {
   void  setInternalPrecision(/*...*/)
                                   raises (FbcException);
   void  setRounding(/*...*/)      raises (FbcException);
   Money add(/*...*/
           in Money oper1,
           in Money oper2)         raises (FbcException);
   // ...
}
```

5 A Frequently Asked Question

Slide 15

Nothing special...

The above specifications have nothing special, and that's why they are appealing to software designers:

- They aren't special as they define the basic (but high-level, application-oriented) facilities that financial applications need: basic tools are good tools.
- They aren't "special" as they are *standard*: they define a common conceptual (as opposed to programming) language.