Lecture Notes on Middleware

Elizabeth Bigelow
School of Computer Science
Carnegie Mellon University
19 November 1998

Odds and Ends

- ODD Presentation Reminder
- Database API
- 15-499!!!!!

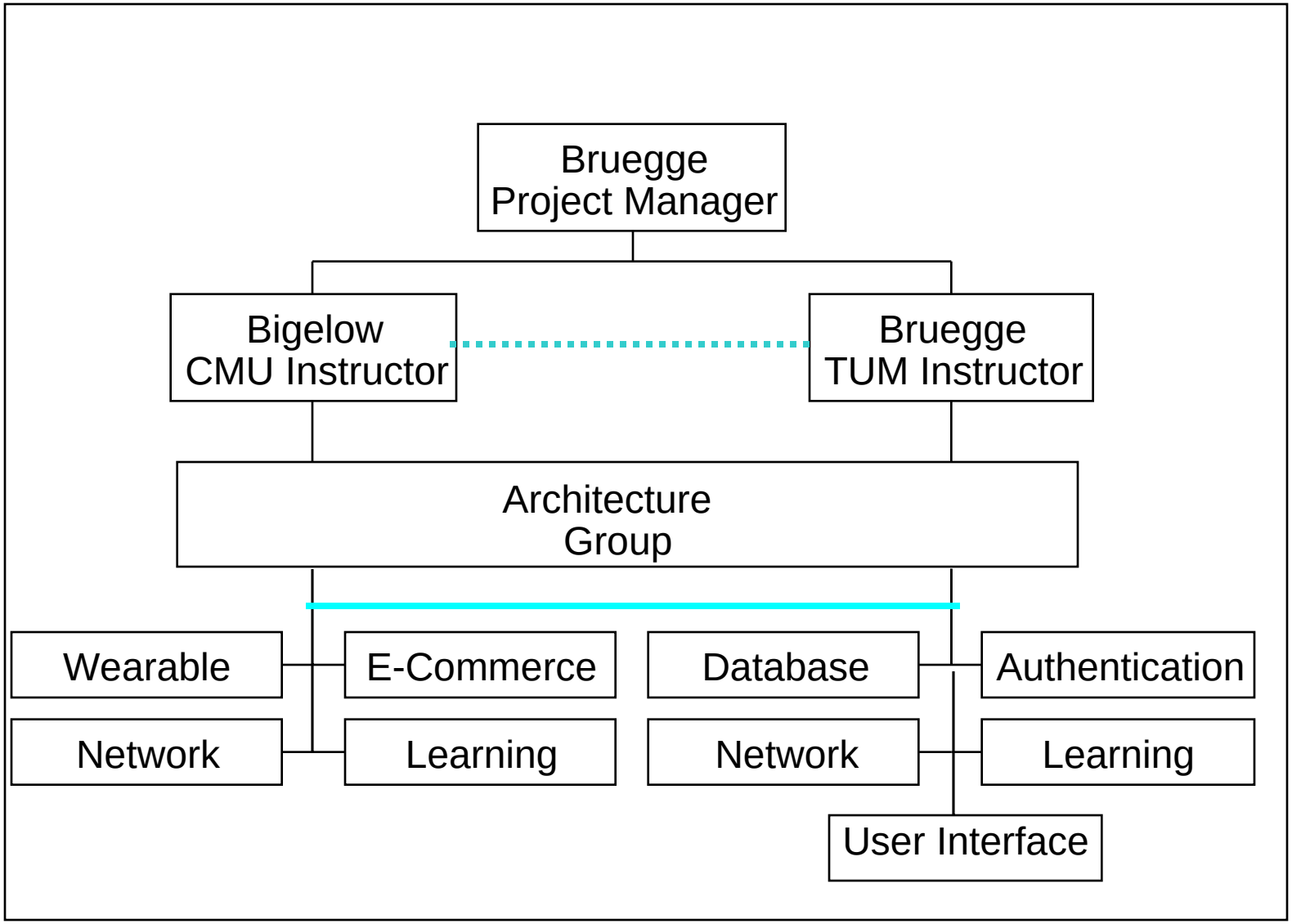
ODD Presenters

- Should post slides by this afternoon
- Should have review with me by Sunday afternoon
- Dry run will be Monday evening at 5:30; check announce bboard for location
- Presentation will be shared with TUM students via posted video (you can also view yourself on the Web later via Quicktime)--URL will be announced
- Bagels will be served (on me)

Next Semester's Advanced Software Engineering 15-499

- Will be continuation of PAID
- "International Project Management"
- All students will need to participate in communication with Germany--lots of technological alternatives will be available
- Experimental approaches--some travel required
- Time ???

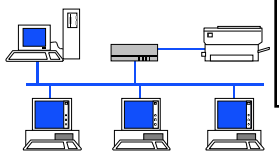
Collaborative Project Structure

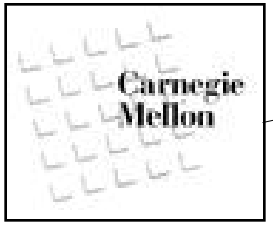


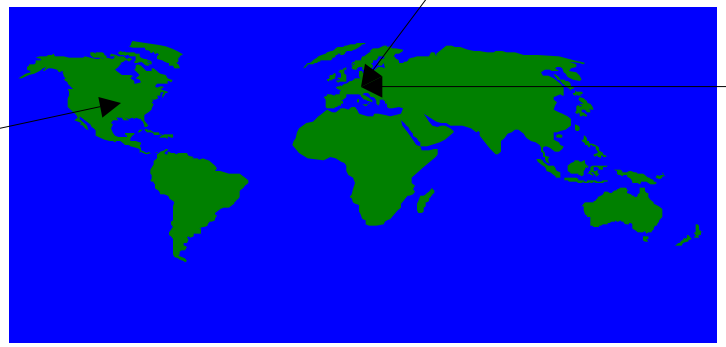
Areas for Work at CMU

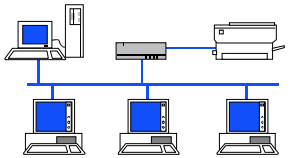
- Wearables
  - **Investigate current state of the art**
  - **Pick a wearable computer**
  - **Design and implement a user interface on it within the design space of CPU, Network and memory considerations**
- E-Commerce
  - **Investigate current state of the art**
  - **Reuse existing billing software from Deutsche Telekom**
  - **Provide a secure interface to this billing software from within PAID**
  - **Integrate with Authentication**

JAMES Infrastructure

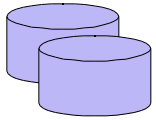




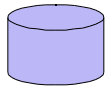




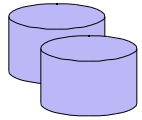
Web Servers

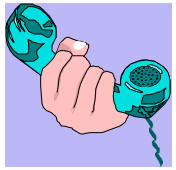


Configuration Management Server



Domino Servers





CMU, continued

- Learning
  - **Investigate current state of the art**
  - **Interface to existing learning algorithm (written in Lisp) and adapt to PAID if necessary**
  - **Utilize simulator built by TUM (utilizing strategy pattern)**
  - **Write learning algorithms and test without recompilation of PAID**
  - **Incorporate learning algorithm from fall 15-413**
- Network
  - **Investigate different telecommunication modes and protocols for wireless and satellite communication**
  - **Provide a state of the art investigation on networks**
  - **Demonstrate disconnected operation and operation in the context of failure**

Areas of Work for TUM

- Database
  - **Interface to Daimler Benz' headquarter databases**
  - **Integrate the updates algorithm from Ingo Schneider**
- Learning
  - **Investigate the current state of the art**
  - **Write a testbed for exploring different learning algorithms (plug and play using different learning algorithms)**
  - **Develop metrics and tests for characterizing the quality of learning algorithms**
- Authentication
  - **Provide a secure authentication system**

CORBA - Outline

- Where does CORBA come from?
  - **OMG (Object Management Group)**
  - **OMA (Object Management Architecture)**
  - **ORB (Object Request Broker)**
  - **CORBA (Common Architecture for ORBs)**
- What is CORBA?
  - **Why CORBA?**
  - **Design Goals**
  - **Status**
- Using a CORBA service
- Structure of an Object Request Broker
- CORBA Interfaces and IDL
- Recommended Readings and References

Where does this come from?

- Object Management Group (OMG)
  - **More than 750 software vendors, software developers and end users**
  - **Goal: Improve the development and use of integrated software systems by supporting and encouraging modular production of software , reuse of code, integration and long-term maintenance**
  - **How? By providing a common architectural framework for object oriented applications based on widely available interface specifications**
  - **Benefits**
    - **Portability**
    - **Reuse**
    - **Interoperability**
- Object Management Architecture (OMA)

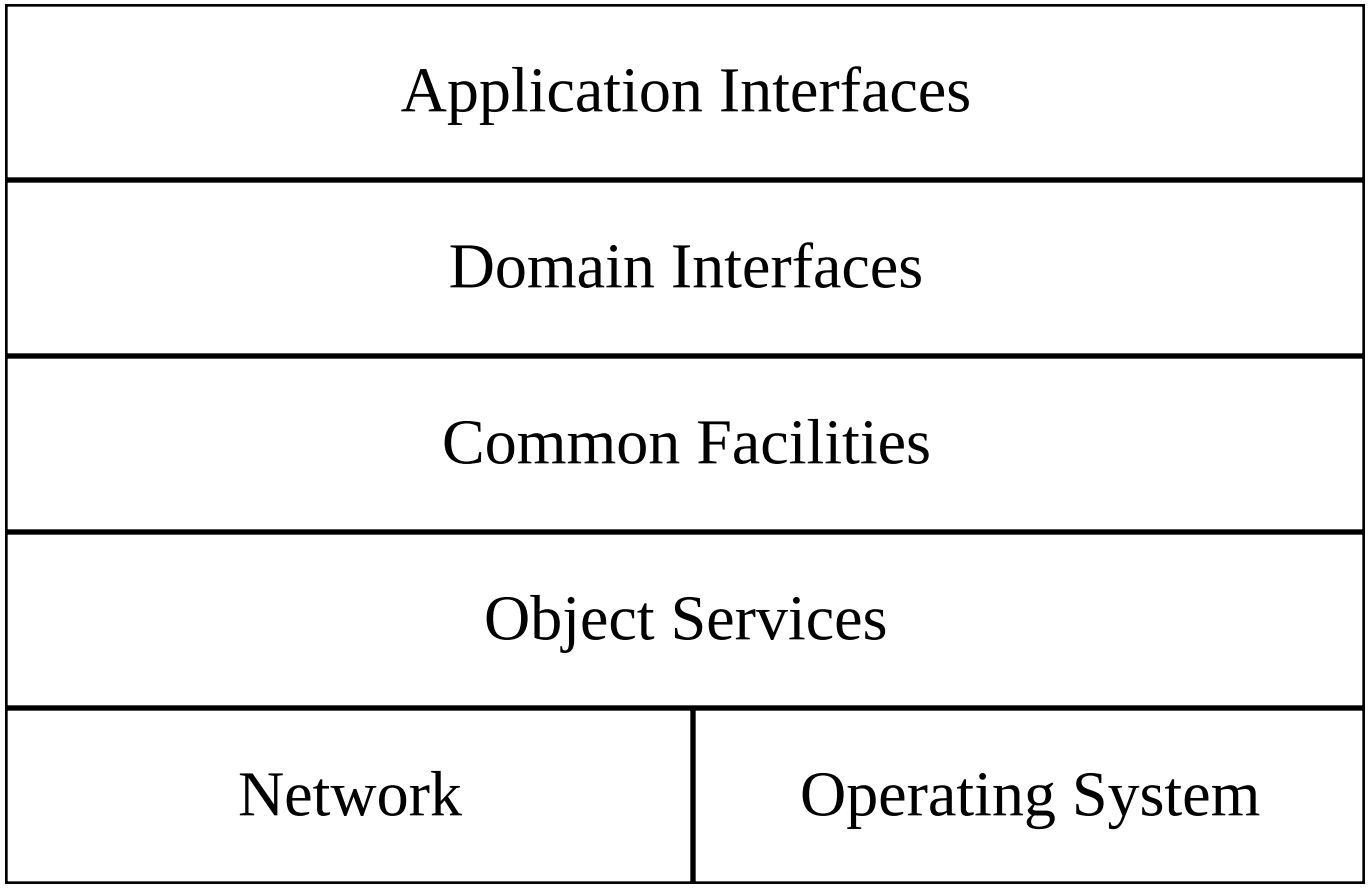
Object Management Architecture

- Focuses on managed objects
  - **Managed objects are subject to systemwide administration and control.**
  - **Managed objects are installed, activated and dynamically controlled**
- Managed objects are the primary building blocks of the OMA object model
  - **Application objects**
  - **Domain objects**
  - **Object Services**
  - **Common Facilities**

Object Management Architecture Components

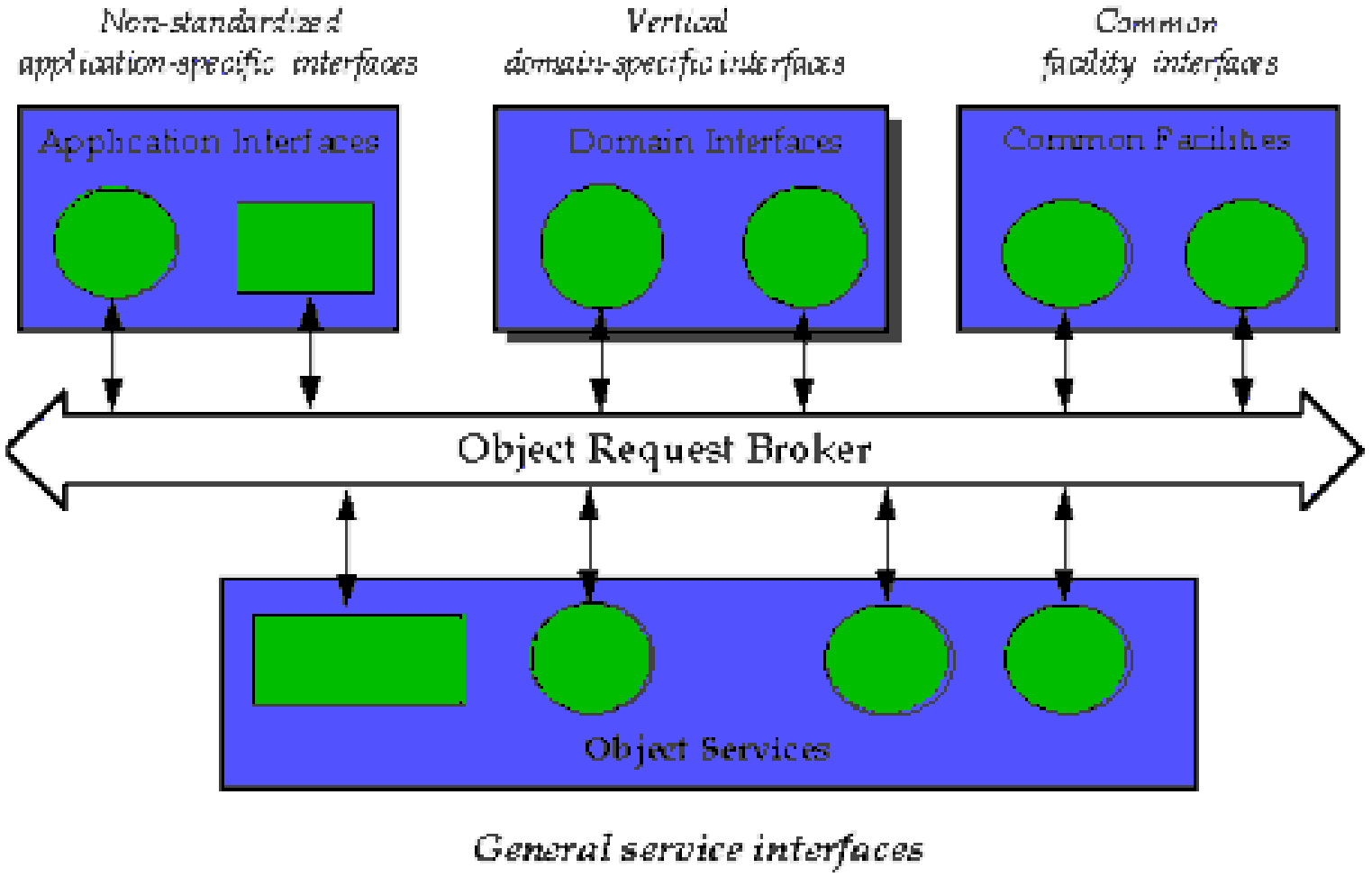
- Application Domain-Specific Interfaces
  - **Non-standardized application-specific interfaces**
- Domain-Specific Interfaces (added in 1996)
  - **Domain-specific interfaces**
  - **Use for specific application domains, such as Finance, Healthcare, Manufacturing and Telecom**
- Common Facility Interfaces
  - **Interfaces for horizontal end-user-oriented facilities**
  - **Use across application domains**
- Object Services
  - **General purpose services that provide a universal application domain-independent, basis for application interoperability**

OMA's object model: Hierarchical view with layers



Problem with this view: It does not show peer-to-peer property of the OMA architecture

Object Management Architecture: Canonical View



Object Request Broker

- The Object Request Broker provides scalability for a distributed object application
- Allows an object to call methods on objects independent of the location of the objects (location-transparency, location independence)
  - **Different process or different machine**
  - **Think of Object-Oriented RPC (Remote Procedure Call) across multiple languages and multiple platforms**
- The ORB provides objects services as well as object facilities. Primary difference between object services and object facilities:
  - **Object services interoperate mostly with the ORB**
  - **Object facilities operate mostly with application and domain objects**
- Standardizing ORBs: CORBA (“an instance of the class ORB”)

Introducing CORBA

- A *specific standard interface for an Object Request Broker*
  - **Common Object Request Broker Architecture**
  - **Selected by the OMG in 1991**
- The CORBA specification defines interfaces, not their implementation
- Abstracts network services and OS services, making them appear as objects within the ORB
  - **Does not hide the network or operating system, but allows programmers to hide them**
- CORBA supports multiple ORBs
- A CORBA object is an interface definition in the Interface Definition Language (IDL)
- CORBA objects: CORBAservices, CORBAfacilities, application objects, domain objects

Why CORBA?

- How did we integrate distributed components before CORBA?
  - **Sockets, Net DDE, DCOM, DCE RPC**
- These technologies did not address key issues:
  - **Object-Oriented technologies (Java, C++)**
    - **Sockets, RPC, etc don't support objects**
  - **Cross-platform, cross-language, multi-vendor support**
    - **Try writing portable networking code with RPC sometime…**
    - **No easy integration with legacy systems**
  - **There is a need for "Common services"**
    - **Security, transactions, persistence, events**
    - **No need to reinvent these services (Build vs buy)**
  - **Full location transparency (location independence)**

Design Goals of CORBA

- Hardware / OS / Network / Language independence
  - **Abstract definition of objects and types**
  - **Open, vendor-neutral specification**
- Location independence
  - **No knowledge necessary of where objects reside**
  - **Object locations determined during deployment/installation (after development)**
- Implementation flexibility allows both
  - **Easy quick/dirty/simple implementations**
  - **Full-strength fast, fault-tolerant, production-quality implementations**
  - **Developer can decide on quality of components**

CORBA Status

- CORBA 1.0 introduced in 1991
  - **Interface Definition Language (IDL)**
  - **Basic Object Adapter (BOA)**
- CORBA 2.0 finalized in 1996
  - **Internet Inter-Object Protocol (IIOP)**
    - **Provides interoperability between different vendor ORBs**
- CORBA / IIOP 2.1 finalized in 1997
- CORBA Services Specification (COSS)
  - **Naming Service, Event Service, Persistence, Security, Lifecycle, etc**
- CORBA Facilities
  - **User Interface, Information and System Management**

CORBAservices Overview

- CORBAservices are intrinsic part of the reference model
- CORBAservices come in 7 flavors
  - **Class management:** Abilities to create, delete, modify, copy, move, distribute, describe class definitions and class interfaces
  - **Instance management:** Same as class management, minus "distribute and describe", plus: "invocation"
  - **Storage:** Persistency for all sizes of objects, including attributes and operations
  - **Integrity:** Consistency both within and among objects, needed for transactions.
  - **Security:** Ability to define and enforce access control on objects
  - **Query:** Use of a predicate to select objects
  - **Version:** Ability to manage variant objects.

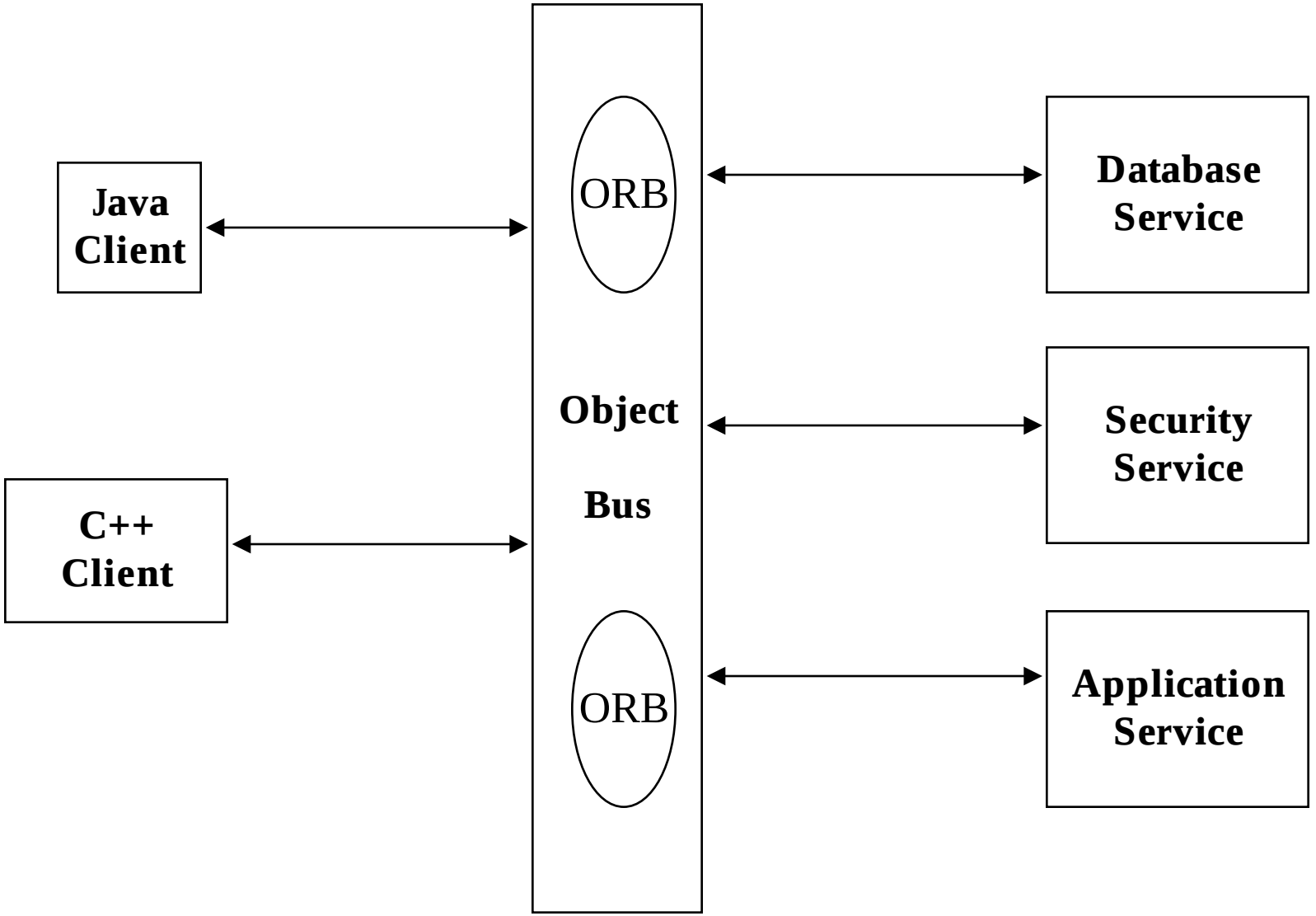
CORBAfacilities Overview

- CORBAfacilities are optional (not part of the reference model)
- Proposed facilities:
  - **Catalog and browser for objects and classes**
  - **Link Manager**
  - **Reusable user interface component**
  - **Printing and spooling**
  - **Error facilities**
  - **Help facilities**
  - **Mail facilities**
  - **Computer-based training**
  - **Information repository access**
  - **Agents**
  - **User preferences**

CORBA Application and Domain Objects

- Application and domain objects are on the same level as CORBAfacilities
- Difference:
  - **CORBAfacilities are general services across many application domains**
  - **Application objects are reusable components within some application domain**
  - **Domain objects: Objects that are relevant to more than one domain but perhaps not all applications**
  - **Domain-specific objects are handled by the DTF (Domain-specific task force). Special cases:**
    - **Business Object Domain Task Force (BODTF)**
    - **Financial Domain Task Force (FDTF)**

Structure of a CORBA application



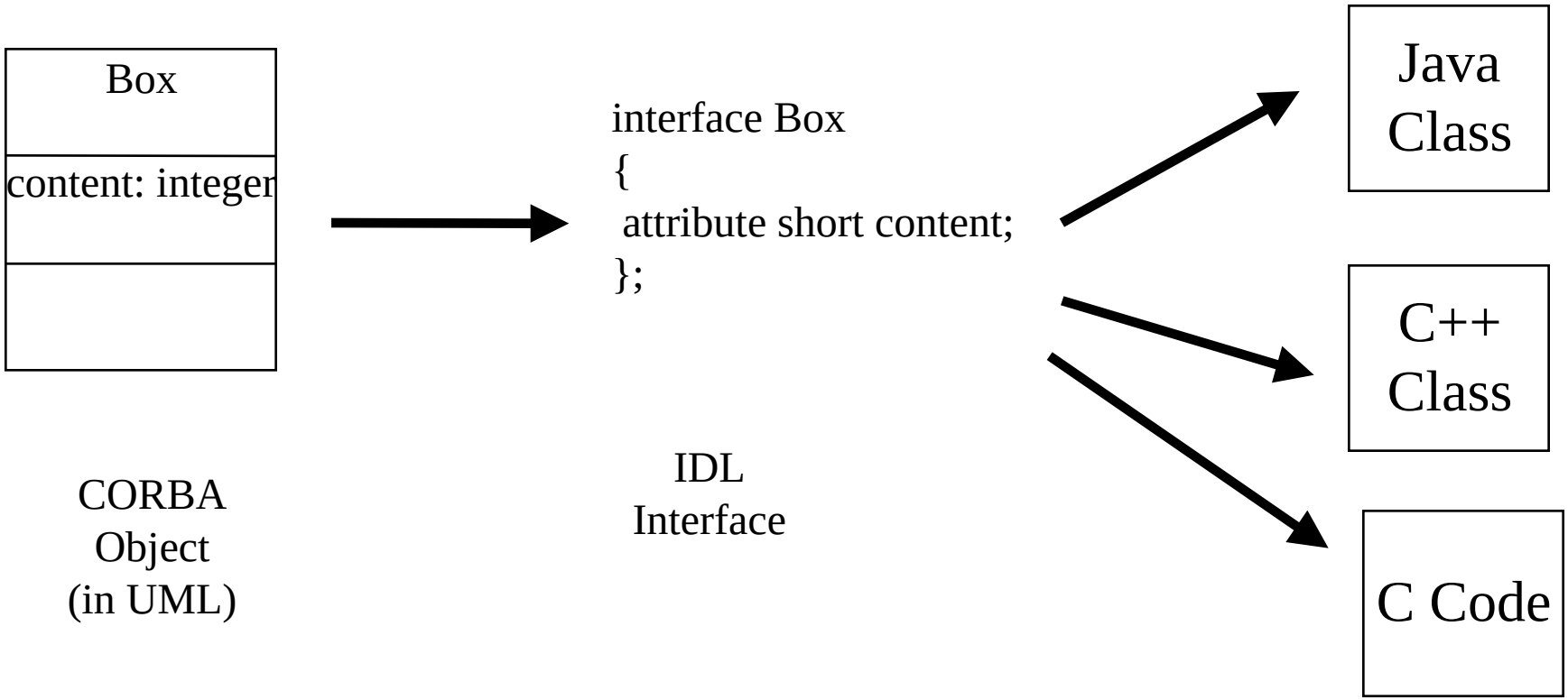
IDL

- IDL is CORBA's object contract language
  - **A language for expressing complex types called interfaces**
- IDL is not a complete programming language
  - **No iterates or control flow**
- The IDL compiler consists of two parts
  - **Front End**
    - **Understands IDL syntax, creates intermediate representation**
  - **Back End**
    - **Understands the target language (C++, Java,..)**
    - **Takes intermediate representation and produces language specific source code**
    - **Creates "stubs": Client side stubs of the interface**
    - **Creates "skeletons": Server side stubs of the interface**

2 minute IDL primer

- IDL looks and smells like C++
- Types are familiar
  - **char, long**
  - **do arrays with “sequence of <type>“**
- Other stuff
  - **Exceptions - class that represents an “exceptional condition”**
  - **Modules - allows encapsulation of naming (avoid clashes)**
  - **in/out/input - Parameter passing foo**
  - **Good IDL references is available**

IDL Compiler: From Object Model to Target Code



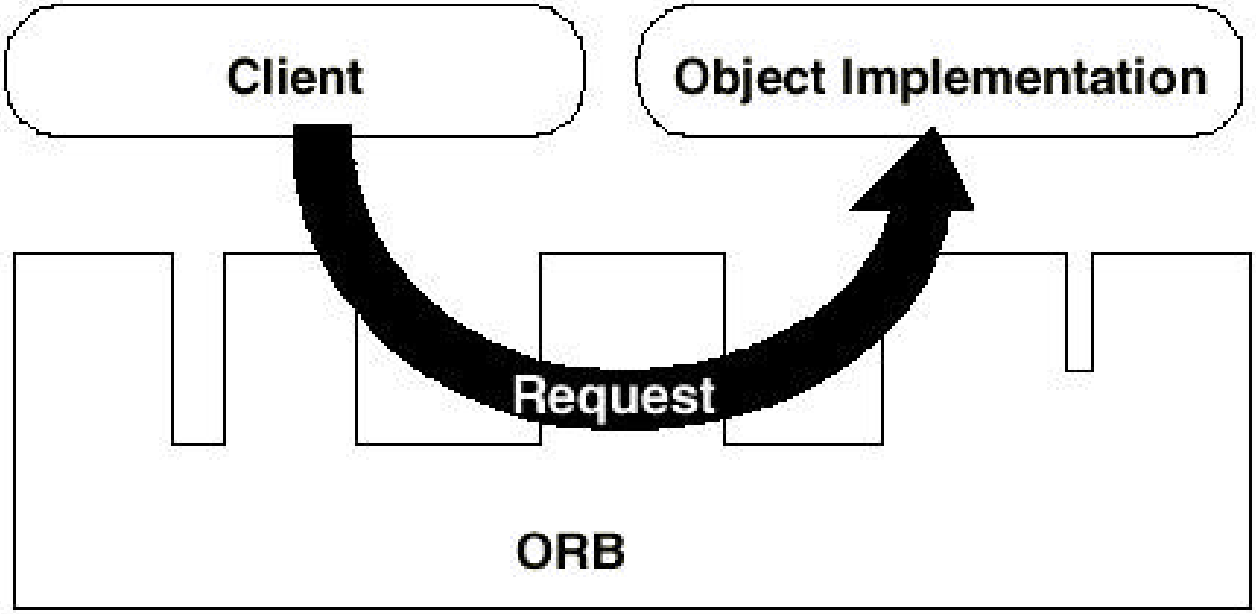
IDL Type Mappings

IDL	C Mapping	C++ Mapping
char	signed char	signed char
octet	unsigned char	unsigned char
boolean	unsigned char	unsigned char
enum	enum	enum

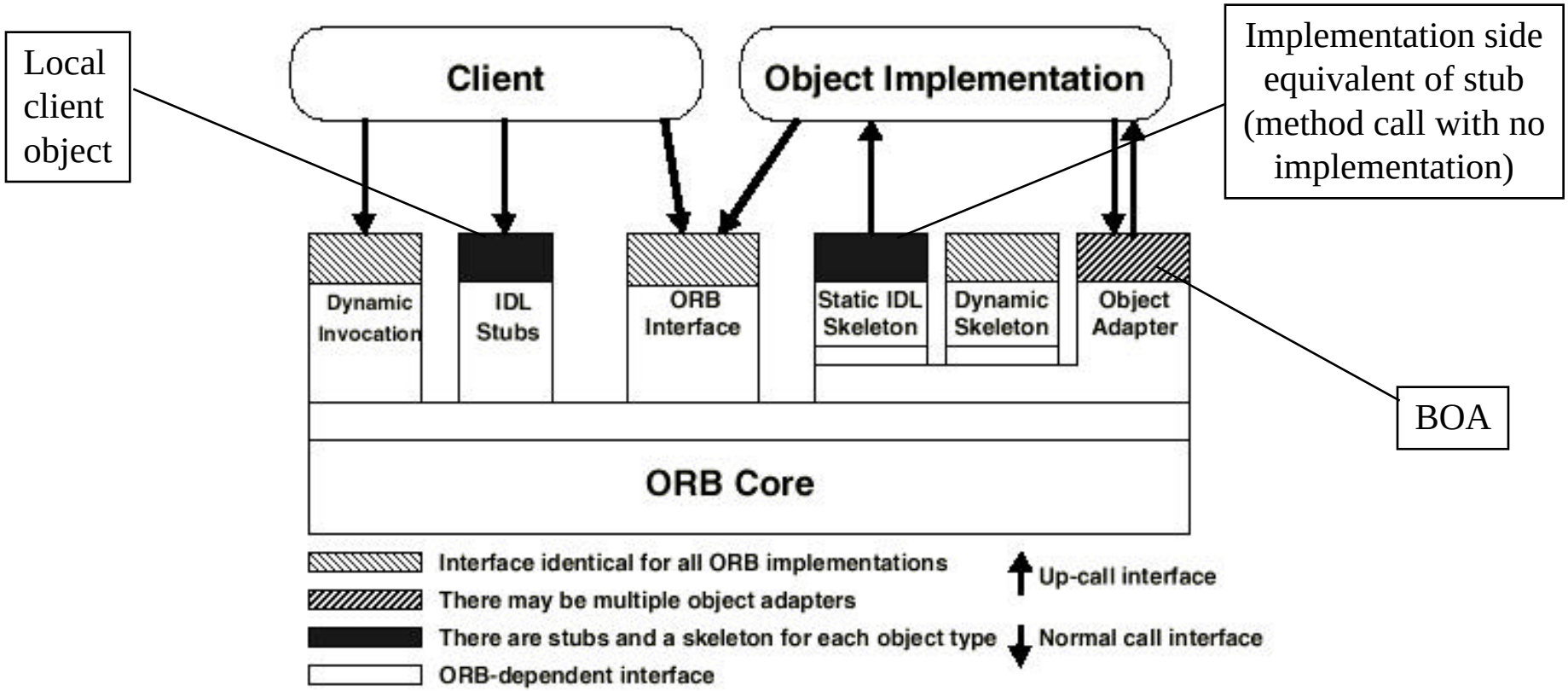
any	typedef struct any {
	TypeCode _Type;
	voide * _value;
	} any;
		class Any {
		....;
		};

- C Mapping in all specifications since CORBA 1.1
- C++ Mapping is part of CORBA 2.0 specification

Structure of an Object Request Broker



Structure of an Object Request Broker (2/24/98)



Proxy Objects, Marshaling, Unmarshaling

- Goal: When IDL definitions are compiled by the IDL compiler, code is generated that allows an operation to be invoked as if it were a method on a local object
- *Marshaling:*
  - **Conversion of programming language data types into a format ready for transmission on a lower layer (physical layer). Done by stub code**
- *Proxy object:*
  - **An object, in which all methods are forwarded such that they are received by the (possible remote) object implementation**
  - **Each stub code implements a set of proxy objects for a specific IDL interface**
- *Unmarshaling:*
  - **The inverse of marshaling. Conversion possibly into a data type in a different language. Done by the skeleton code**

BOA

- Needed because the ORB Core is free to be implemented in a variety of ways, adapter interfaces are defined to provide standard interfaces to servers
- There is currently only one standard Object Adapter defined in CORBA 2:
- Basic Object Adapter
  - **Basic was supposed to mean minimal, but BOAs are quite complicated**
- The BOA is involved with various parts of a CORBA object's lifecycle: creation, destruction, activation and deactivation.
- BOA operations
  - **Object creation and deletion**
  - **Obtain the principal associated with a client request**
  - **Signal whether an implementation is ready or not**
- A BOA has a IDL interface

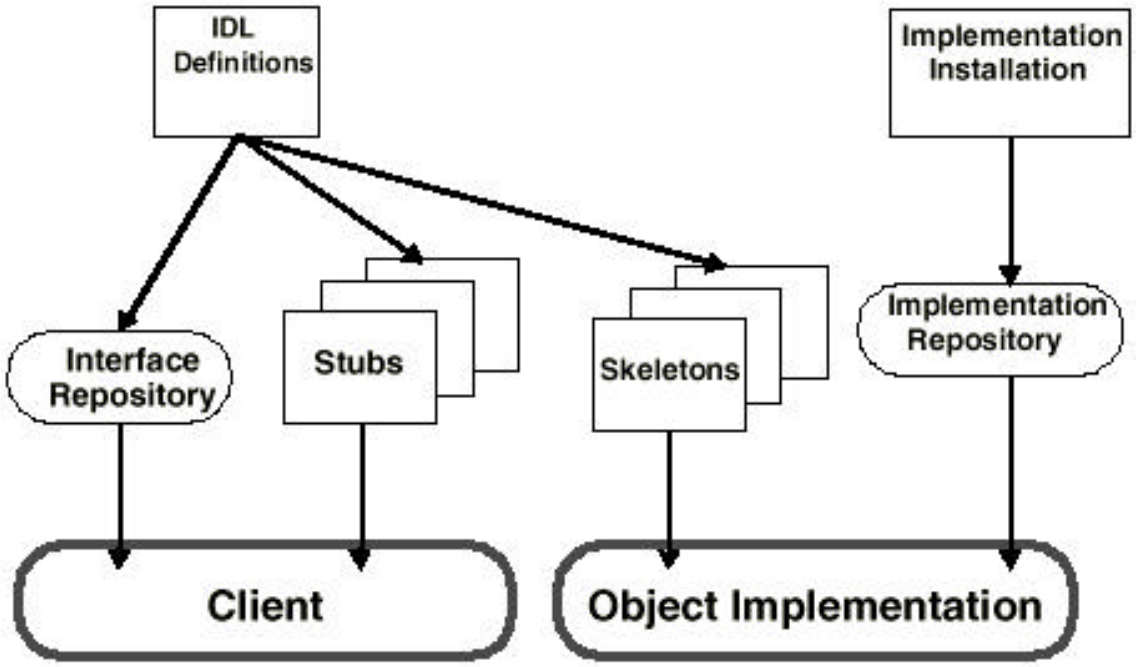
Invoking a CORBA interface

- Two methods: Static (with IDL) or Dynamic (with DII)
- Static; looks like normal method / procedure call
  - **In Java, simply "import" object definition**
  - **In C++, #include header and link with library**
  - **Find object on ORB**
    - **Variety of methods, some vendor-specific**
    - **Naming services provide powerful object lookup**
  - **"bind" to the desired object**
  - **Finally, use object like any other!**
- Dynamic Invocation Interface (DII)
  - **Query Interface repository about arguments and operations**
  - **manually build up argument list**

Dynamic Invocation Interface

- In IDL all objects are defined at compilation time.
  - **Allows good code optimization**
  - **No overhead at run time**
- Sometimes IDL is too restrictive. The dynamic invocation interface (DII) allows to make requests on objects that are unknown at implementation time
- Advantages of DII over IDL:
  - **Ideal for software development based on prototyping**
  - **Takes less than 80% of compiled IDL code**
  - **New objects do not require recompilation of existing code**

Interface and Implementation Repositories



Repository

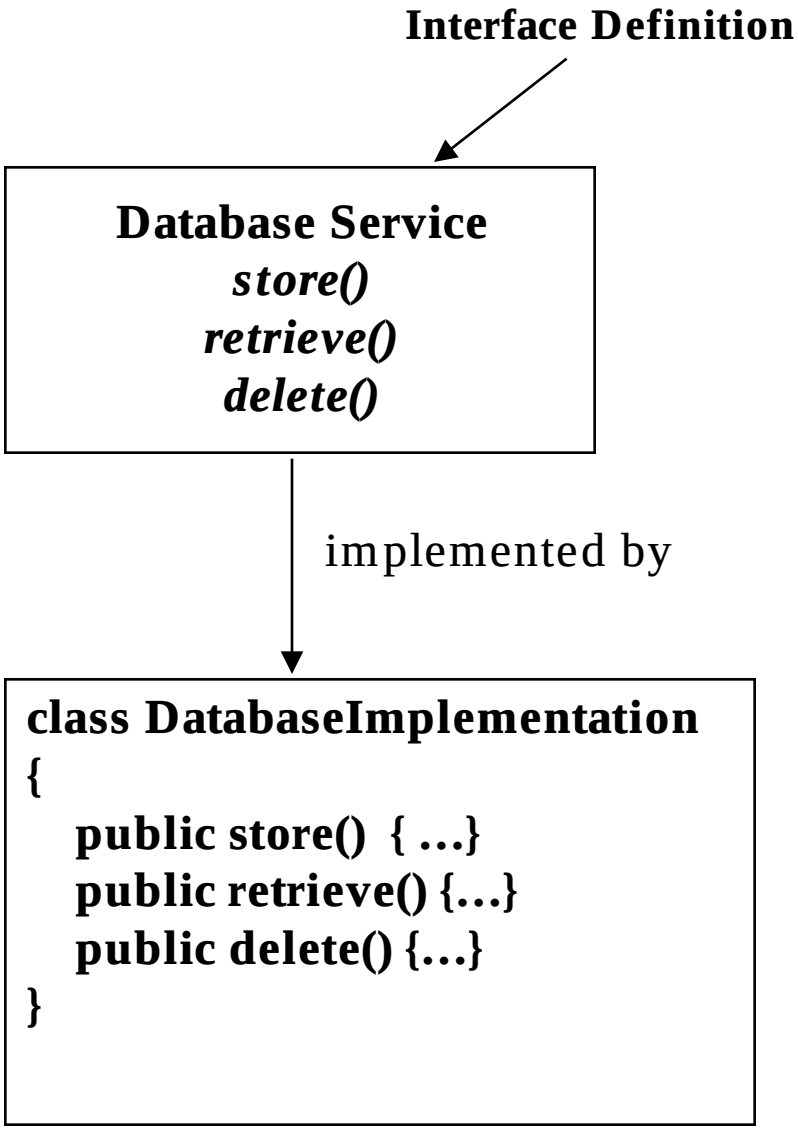
- A repository is a service, that when presented with a query, returns some object of information
- In many aspects, a repository is like a database
  - **lighter-weight, because it needs to support only a specific application**
  - **Flat files or a dynamically linked library could form a CORBA repository**
- CORBA specification mentions two repositories:
  - **Interface repository:**
    - **Registry of fully qualified interface definitions. Can be browsed by the Dynamic Invocation Interface client to construct invocations on an interface**
  - **Implementation repository:**
    - **Currently not well defined. Basic Idea: Implementations provide information such that they can be invoked. Not clear whether more than a path in the local file system is necessary.**

Creating a CORBA Application

- Perform object-oriented analysis and design of system
- Generate IDL from the object model
  - **Define "interfaces" for objects**
  - **Define structures, types, exceptions**
- Map IDL files into client code and server stubs
  - **Generated client code contains mechanism for communicating with servers**
    - **Marshalling ("translating a type to bits on the wire"), transport, conversions, etc.**
  - **Server stubs must be implemented**
    - **Often via inheritance (C++, Java)**

Creating a CORBA object (service, facility, application object)

- Interface defined by IDL
  - **Interface Definition Language**
  - **Defines behavior, not implementation**
- Implemented by…
  - **Java/C++ code**
  - **Mapped to a database**
  - **Linked to a legacy system**
- CORBA service maps:
  - **from client language types**
  - **to CORBA-neutral types**
  - **to implementation types**
    - **e.g. Java classes**
  - **And back again..**
- To the client, the service appears to be written in the native language



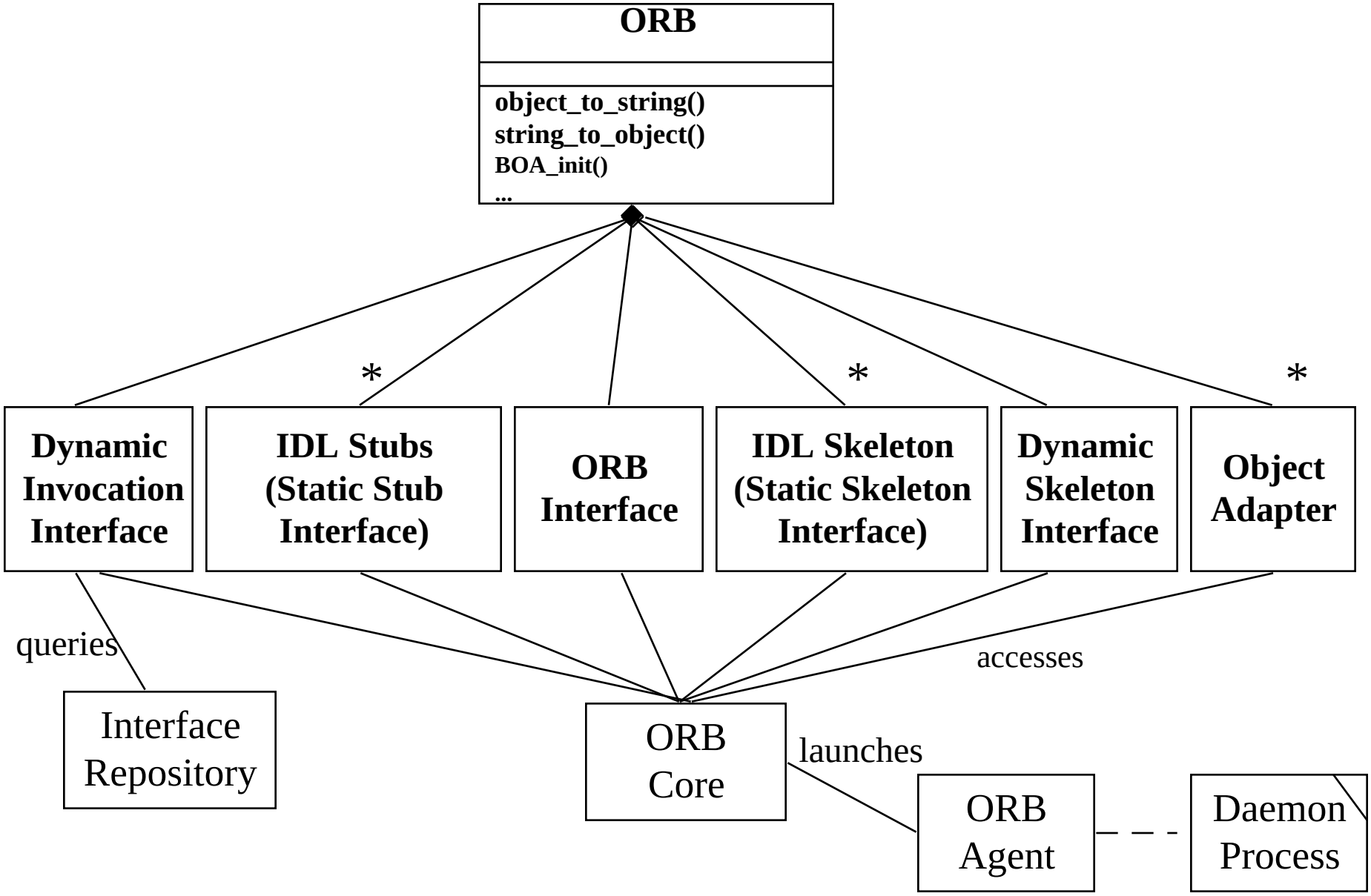
CORBA Development Process

- Write IDL code that describes the interfaces to objects running on different platforms or implemented in different languages
- Compile the interfaces with the IDL compiler
  - **This produces stubs and skeleton code**
- Write code to initialize the ORB and informa it of any CORBA objects that are created
- Compile all the generated code
- Run the distributed application

Using CORBA: Setting up a Service

```
public static void main(String[] args) {


  // initialize the Object Request Broker
  ORB orb = ORB.init();

  // initialize the Basic Object Adapter (BOA)
  BOA boa = orb.BOA_init();


  // instantiate the CORBA Object “MyServer”
  MyServer srvr = new MyServerImpl(”MyServer");

  // tell the BOA (ORB) that the CORBA object is ready
  boa.obj_is_ready(srvr);


  // tell the BOA (ORB) that  “MyServer” is ready to accept requests
  boa.impl_is_ready();
}
```

UML Model of the Object Request Broker



Data Dictionary for ORB object model

- **IDL Stubs-** The code generated for a specific IDL interface to allow static invocation of that interface. Linked into a CORBA client
- **IDL Skeleton -** The code generated for a specific IDL interface. Linked into a CORBA object implementation
- **Dynamic Invocation Interface -** Allows invocations of CORBA operations without IDL stubs
- **Dynamic Skeleton Interface -** Interface that allows interpretation of requests to a server for types that were not known at compile time
- **ORB Interface -** Interface offering miscellaneous services from the ORB to clients and servers
- **ORB agent -** Locates and launches servers; facilitates client communication with servers
- **Object Adapter -** Capable of activating servers whose objects are required by invocations. (After a server is ready it must inform the Object Adapter that its objects can receive requests).
- **Interface Repository -** Stores operations and parameter types of CORBA objects for discovery.

CORBA Pitfalls

- Object instances are *never* passed across the network
  - **Instances contain platform-specific code**
  - **Only references are passed**
    - **Current discussion in OMG to pass objects**
  - **How do I transmit data?**
    - **Use structures for transmitting data types**
- Don't break encapsulation!
  - **Don't pass pointers, directory names, etc.**
  - **Don't pass any language or platform specific information**
- Making interfaces generic is difficult
  - **Design, rethink, redesign...**

More CORBA Pitfalls

- Integration with other tools problematic
  - **CORBA types automatically generated**
    - **If another tool needs to process the types, you're out of luck**
- Data types are scalars and sequences of scalars *only*
  - **No references to other structures**
    - **That is, no C++ pointers or Java object references**
- Build environment for CORBA application is complex
  - **Lots of classes, directories, processes, etc**
  - **Need "make", powerful editor, decent shell**

Other Observations on CORBA

- CORBA concepts take getting used to
  - **Developing distributed systems is not easy the first time**
  - **CORBA *will* make things much easier once you learn it**
  - **Good documentation is freely available**
- CORBA vs DCOM?
  - **Andreesen says IIOP (CORBA protocol) will replace HTTP**
  - **Many companies moving to distributed systems are starting to use CORBA**
  - **CORBA has widespread industry support**
  - **Microsoft has agreement with IONA to provide CORBA interface to DCOM**

Java Applets as CORBA clients

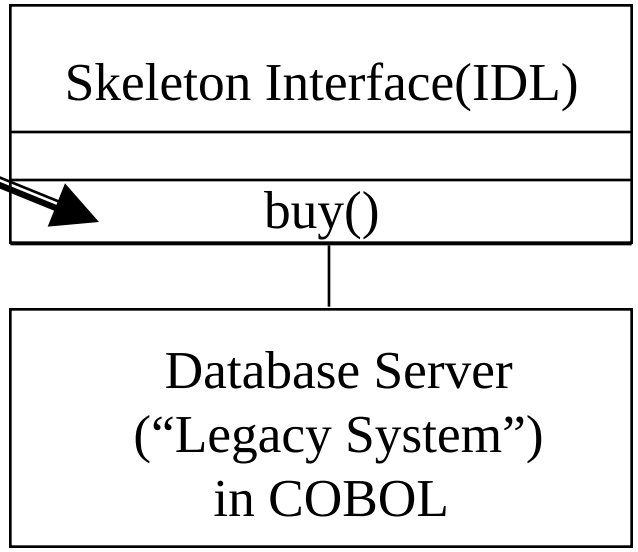
- Interface engineering exercise:
  - **Take an existing standalone application and make it available on the Web**
- Currently a hot topic
- Major Problems:
  - **Java's security model: Applets are only allowed to open network connections to the host from which they have been downloaded (the identification is based on IP numbers)**
    - **Goal: Prevent untrusted applets, prevent viruses, ensure privacy,...**
    - **Java's security model is in conflict with CORBA's goal to allow clients to invoke operations on objects regardless of their physical location (location transparency)**
      - **Security model currently under discussion**
  - **Firewalls: Restrict the communication between an intranet and the Internet.**
    - **CORBA's 2.0 IIOP protocol used for intra-ORB communication is TCP/IP based**

Problem: Applets accessing Legacy Systems

3. The buy() method cannot directly be invoked unless the applet is downloaded from the Database Server

1. Customer wants to buy Laptop

2. Java Applet tries to access Company's Database



Company Database
as CORBA object

HTTP Tunneling

- The applet issues a method call on a CORBA object residing on a different host
- The applet request is put into an HTTP envelope (HTTP can go through firewalls)
- The applet request is sent to an object called Gatekeeper residing on the host from which the applet was downloaded.
- The Gatekeeper forwards the request to the host nominated in the object reference.

Result: HTTP tunneling reestablishes CORBA location transparency for Java applets

Result: HTTP tunneling reestablishes CORBA location transparency for Java applets

Solution: Web-Based Applets using HTTP Tunneling

0. Download Applet

HTTP

WWW Server

Gatekeeper

3. Send Buy() method wrapped in HTTP packet

4. Forward Request to Database

IIOP

Skeleton Interface(IDL)

buy()

1. Customer wants to buy Laptop

2. Java Applet tries to access Company's Database

Database Server ("Legacy System") in COBOL

Company Database as CORBA object

CORBA vs RMI

RMI vs. CORBA

- CORBA presumes a heterogeneous, multilanguage environment.
- RMI lets pass objects by value
- RMI uses Java as both an interface definition language and as an implementation language
- RMI uses URL-based naming scheme

Caffeine

Caffeine

Caffeine

- Offered by Visigenic's Visibroker
- A Java solution for CORBA developers
- Developers write ordinary Java classes using RMI-like semantics to make them remote
- Java2IIOP takes Java interface files and produces IIOP-compliant stubs and skeletons
- Java2IDL takes Java code and produces CORBA IDL

CORBA Implementations

- **Visibroker** by Visigenic
  - **http://www.visigenic.com/**
  - **Licensed for Netscape 4**
- **Object-Broker** by Digital
  - **http://www.digital.com/info/objectbroker/**
- **Orbix** by IONA
  - **http://www.iona.com/**
- **SOM** by IBM
  - **http://www.software.ibm.com/ad/somobjects/**
- Sun's **NEO**
  - **http://www.sun.com/software/neo**
- **Orb Plus** by HP
  - **http://www.hp.com/gsy/orbplus.html**
- **Power-Broker** by Expersoft
  - **http://www.expersoft.com/**
- **DAIS** by ICL
  - **http://www.iclsoft.com/sbs/daismenu.html**

JAVA ORB Products

- **Visigenic's Visibroker for Java**
  - **Released April 1996, CORBA 2.0 compliant, Provides CORBA Naming and Event Services in Java**
  - **CORBA location transparency via HTTP tunneling**
- **Iona's OrbixWeb**
  - **Released July 1996**
  - **CORBA location transparency via the "Wonderwall"**
- **Sun's Joe**
  - **Released July 1996, supports a number of protocols (Door, NEO as well as IIOP)**
  - **CORBA location transparency via a patch to the Apache HTTP server.**
  - **IBM has licenses Joe**
- **CORBAnet**
  - **Internet-based showcase demonstrating ORB interoperability**
  - **Java ORBs in action: http://www.corba.net**

Recommended Readings

- Client-Server Programming with Java and Corba, Authors: Orfali and Harkey, John Wiley & Sons, 1997
- Instant CORBA, Authors: Orfali, Harkey and Edwards, John Wiley & Sons, 1997
- The CORBA Reference Guide, Author: Alan Pope, Addison Wesley, 1998
- CORBA Design Patterns, Authors: Mowbray and Malveau, John Wiley & Sons, 1998
- Java Programming with CORBA, Authors: Andreas Vogel, Keith Duddy, John Wiley & Sons, 1998
- Java Network Programming. Author: Elliotte Rusty Harold. 1st edition. O'Reilly, 1997.
- Other CORBA-related books and magazines
  - **http://www.omg.org/news/begin.htm#books**

References

- Object Management Group (OMG)
  - **Official documentation, links**
  - **http://www.omg.org**
- Distributed Systems Technology Center (DSTC)
  - **General distributed-OO technology info, plus CORBA**
  - **http://www.dstc.edu.au**
- Good CORBA papers, tutorials, examples
  - **http://siesta.cs.wustl.edu/~schmidt/corba.html**