CORBA

Parte: 4c – CORBA “un Middleware”

Corso di: Sistemi Distribuiti
Lauree in: Ingegneria Informatica,
delle Telecomunicazioni ed Informatica di Scienze

Prof. Paolo Nesi

Department of Systems and Informatics, University of Florence
Via S. Marta 3, 50139, Firenze, Italy
tel: +39-055-2758515, fax: +39-055-2758570

DISIT Lab, Sistemi Distribuiti e Tecnologie Internet

http://www.disit.dinfo.unifi.it/

paolo.nesi@unifi.it
http://www.disit.dinfo.unifi.it/nesi

CORBA

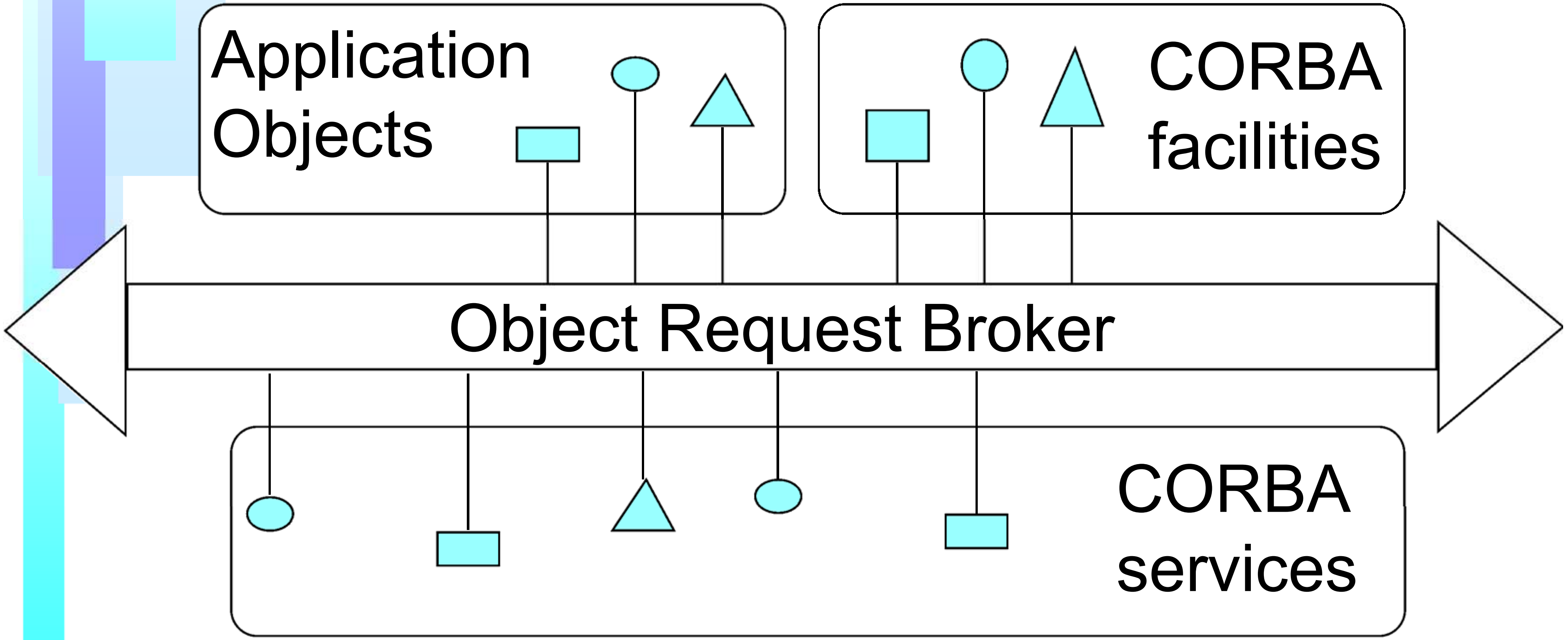
- CORBA Architecture
- General Concepts
- ORB Structure
- Client and Server in CORBA
- Object Adapter
- CORBA for WEB applications
- Usage of CORBA
- Single and Multithread CORBA

CORBA Common Object Request Broker Architecture

- **OMG**'s (Object Management Group) specification for interoperability between distributed computing nodes (1989)
- **ORB**: middleware that establishes requester-provider relationship
- **Goal**:
  - ♣ Usage of OO programming in Distributed Systems
  - ♣ Allow heterogeneous environments communicating at object level
  - ♣ regardless of implementation of the endpoints
    - ➔ Different languages in the applications
    - ➔ Different implementations of the ORB
- CORBA 1 (1990), CORBA 2 (1996)

CORBA, *Common Object Request Broker Architecture*

- Defined by the Object Management Group nel 1991
- Object Management Architecture

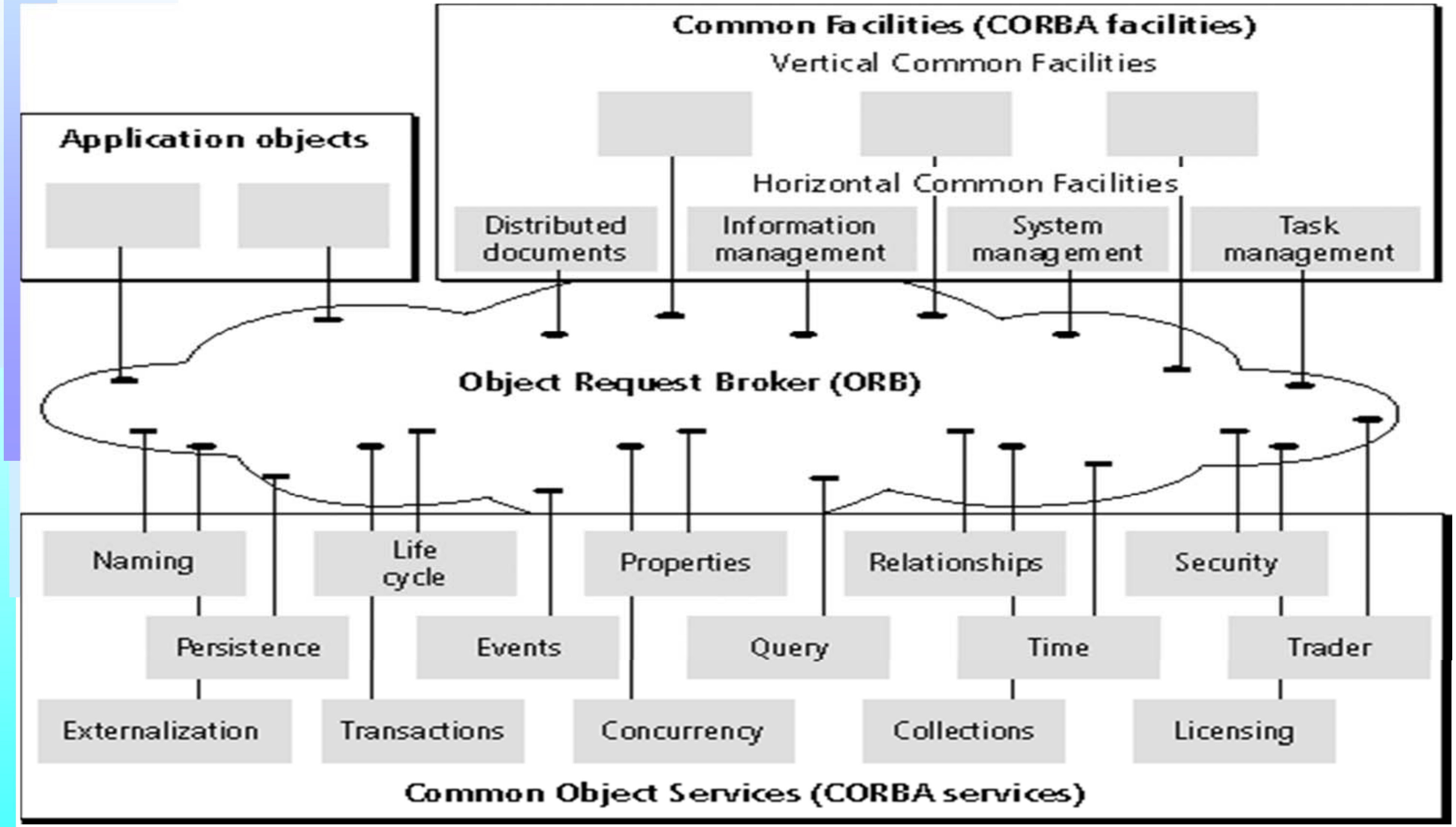


CORBA

- **Object Request Broker (ORB)**
  - The libraries, processes, and other infrastructure in a distributed environment that enable CORBA objects to communicate with each other.
  - The ORB connects objects requesting services to the objects providing them.

- **Naming service**
  - to allow CORBA objects to be named by binding a name to an object reference.
  - The name binding may be stored in the naming service, and
    - a client may supply the name to obtain the desired object reference.

CORBA detailed architecture

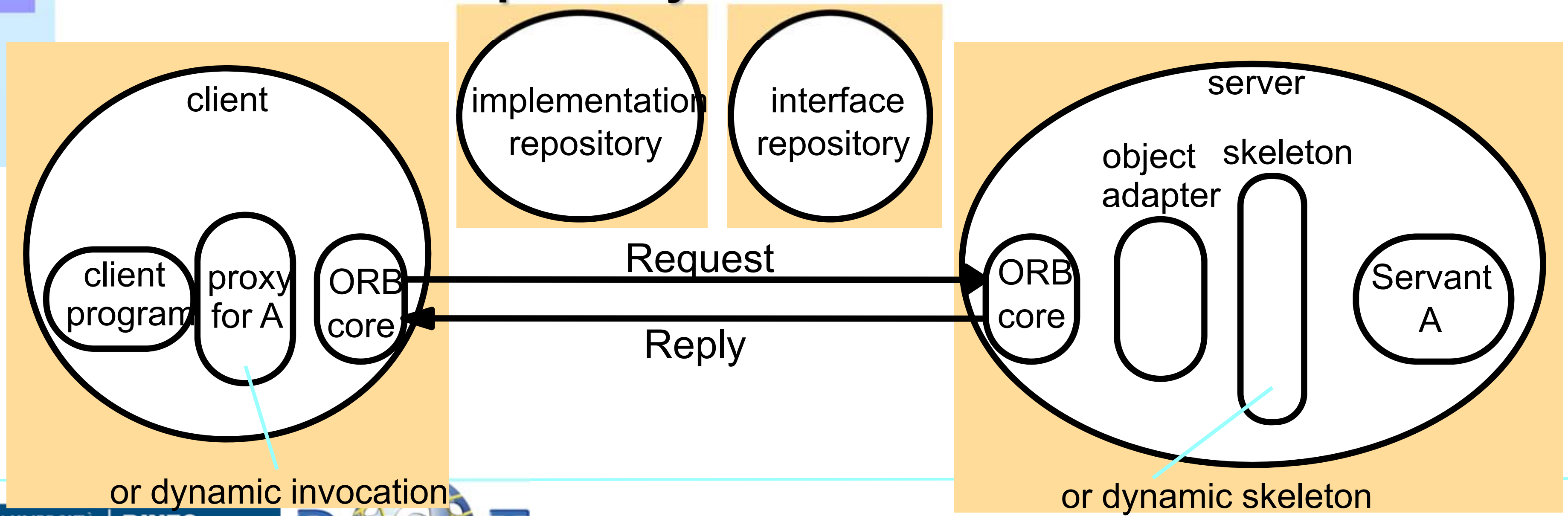


4 Componenti di CORBA

- **ORB, Object Request Broker**
  - Distributed application
  - rende trasparente la locazione fisica degli oggetti, naming
  - unmarshal-marshal, e invocazione dei metodi
- **CORBA Services**
  - Security, time, etc..
  - persistency, events, transactions, etc..
- **CORBA Facilities**
  - Servizi di base condivisi da molte applicazioni
  - Non vitali come i CORBA Services, OS esteso….
  - E.g.: amministrazione sistema, mail, etc.
- **Application Objects**
  - Objects basati su CORBA

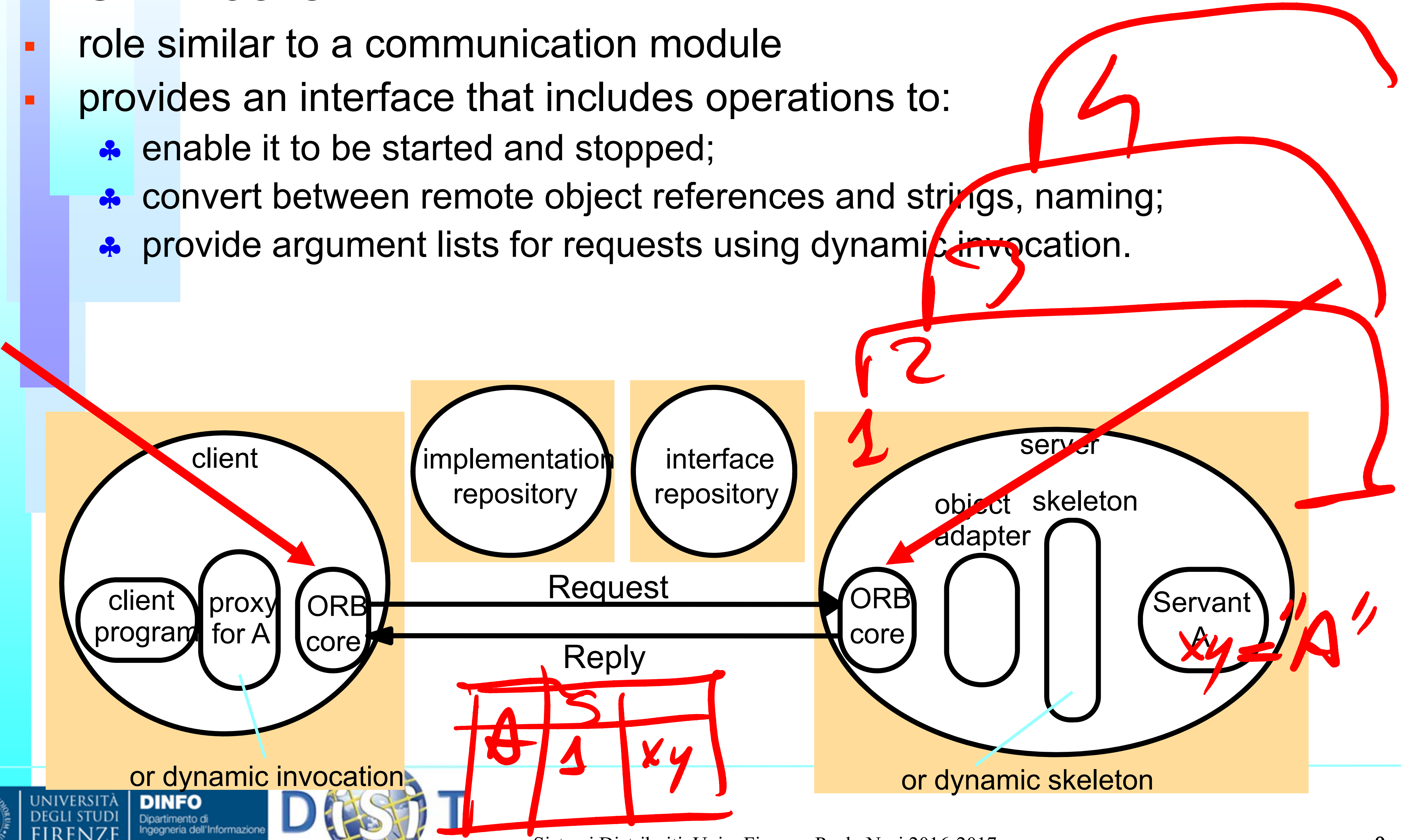
The main components of the CORBA architecture

- The CORBA architecture is designed to allow clients to invoke methods in CORBA objects
  - clients and objects can be implemented in a variety of programming languages
  - it has additional components with respect to a generic MiddleWare →
    - **object adapter, implementation repository and interface repository**



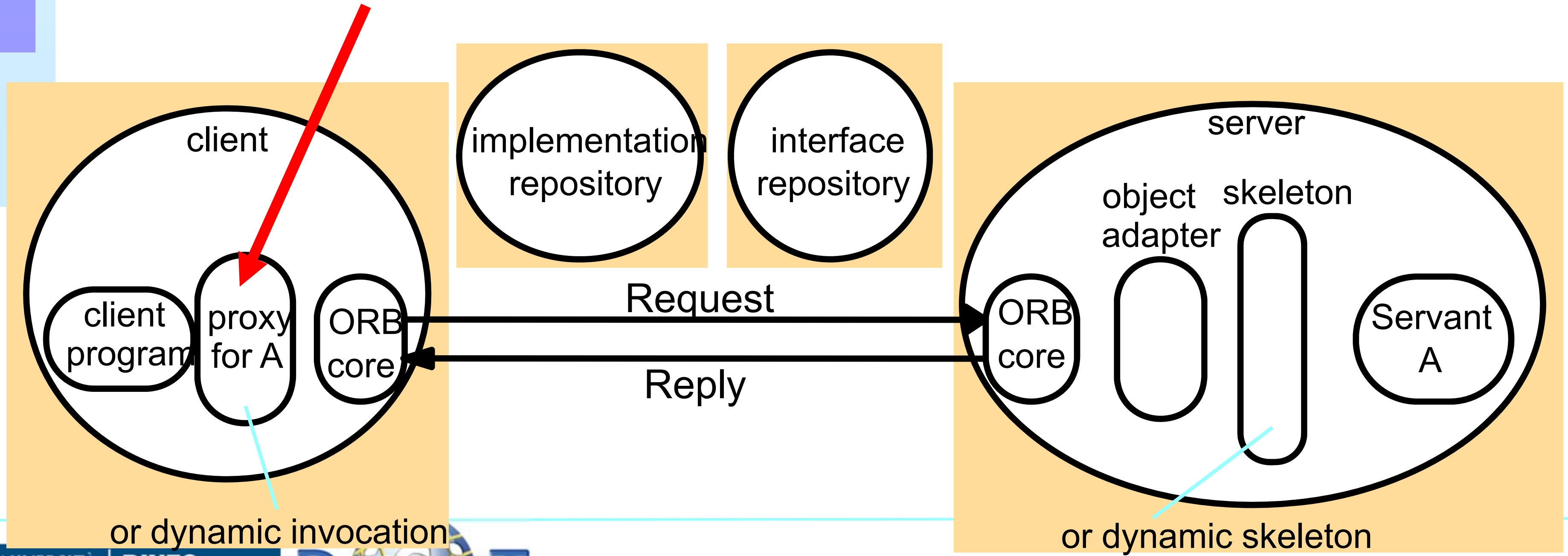
The main components of the CORBA architecture

- ORB core
- role similar to a communication module
- provides an interface that includes operations to:
  - enable it to be started and stopped;
  - convert between remote object references and strings, naming;
  - provide argument lists for requests using dynamic invocation.



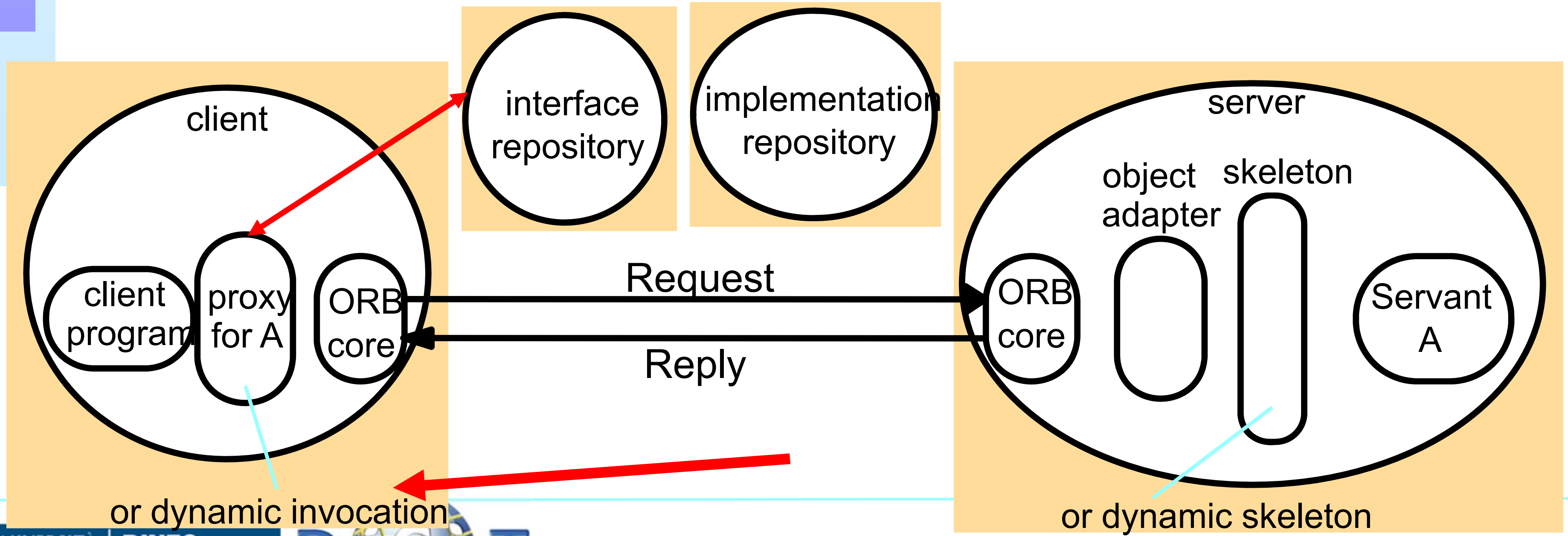
The main components of the CORBA architecture

- Client stubs/proxies
- Written in the **client language**.
- The IDL compiler for the client language uses an IDL interface to generate one of the following:
  - **for object-oriented languages** the class of a proxy
  - **for procedural languages** a set of stub procedures.
- the client stubs/proxies **marshal** the arguments in invocation requests and unmarshal exceptions and results in replies



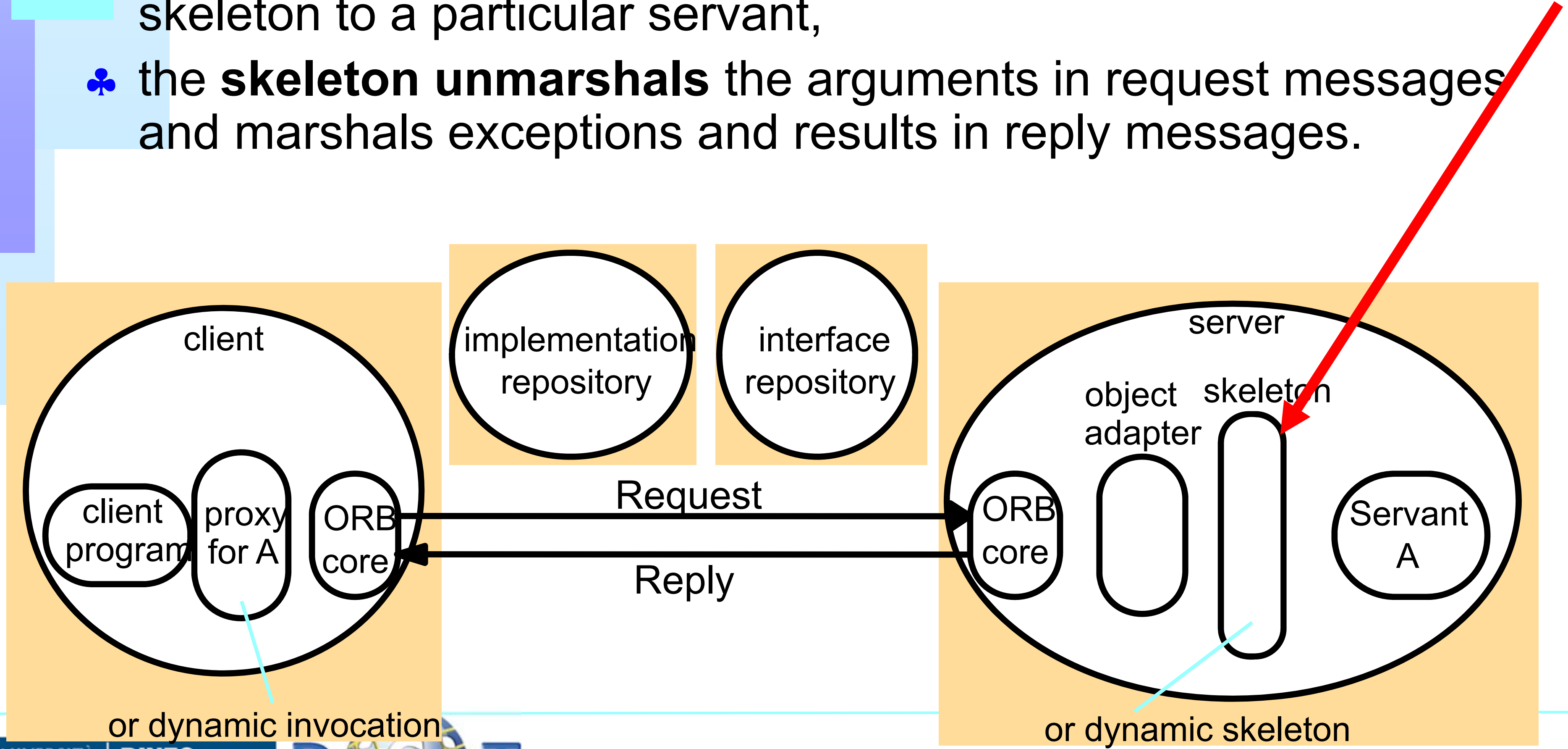
The main components of the CORBA architecture

- ***Dynamic invocation interface***
- In some applications (e.g., browsers), a client without the appropriate proxy class may need to invoke a method in a remote object.
- CORBA does not allow classes for proxies to be downloaded at run time as in Java RMI.
- The **dynamic invocation interface is CORBA's alternative**. (see the **Interface Repository**)



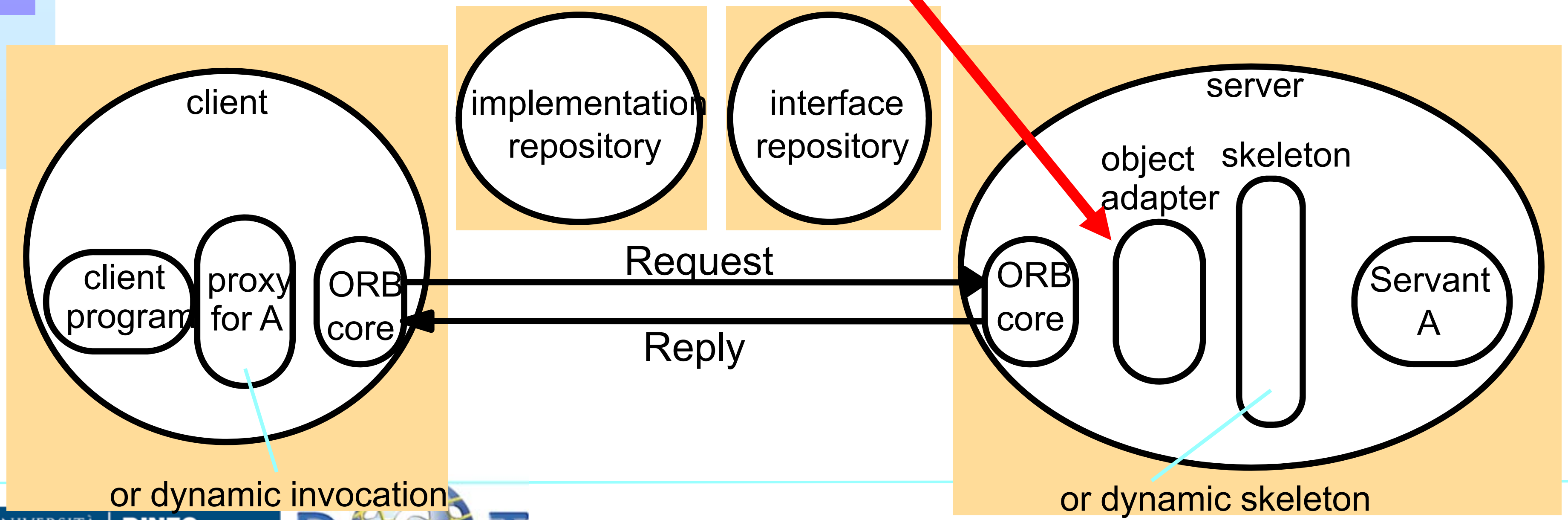
The main components of the CORBA architecture

- ***Skeletons***
  - skeleton classes (for OO languages) are **generated in the language of the server** by the IDL compiler.
  - remote method invocations are **dispatched** via the appropriate skeleton to a particular servant,
  - the **skeleton unmarshals** the arguments in request messages and marshals exceptions and results in reply messages.



The main components of the CORBA architecture

- ***Object adapter***
- bridges the gap between
  - CORBA objects with IDL interfaces and
  - the **programming language interfaces** of the corresponding servant classes.
- it does the **work of the remote reference and dispatcher modules**

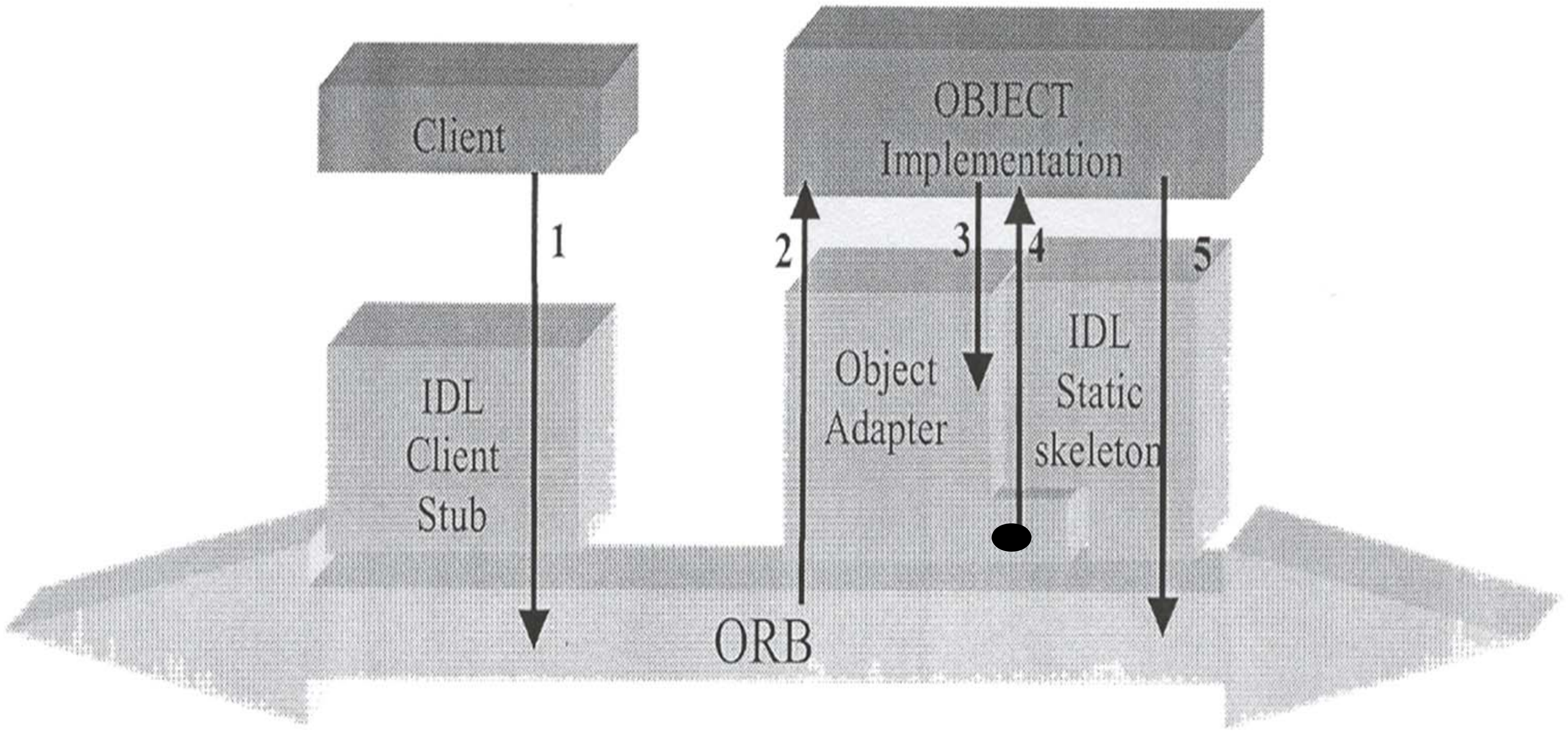


CORBA

Object Adapter

- **Object Adapter (Portable object adapter)**
  - provides ORB services to particular groups of object implementations
- **Services and duties/activities**
  - generation and interpretation of object references, mapping object references to
    - Specific implementations, and
    - registration of implementations.
  - method invocation (dispatching)
    - via a skeleton, object and implementation activation and deactivation
  - security of interactions (method access control, etc.)

OA, Object Adapter



OA, Object Adapter

1. The client via the stub calls the method at the ORB.
2. The ORB notifies the calls the OA which activates in turn the correct implementation
3. L'implementazione si registra e si dichiara pronta.
4. L'OA passa l'invocazione allo skeleton che spacchetta i parametri e li fornisce all'implementazione.
5. L'implementazione esegue il metodo e ritorna parametri di ritorno al client.

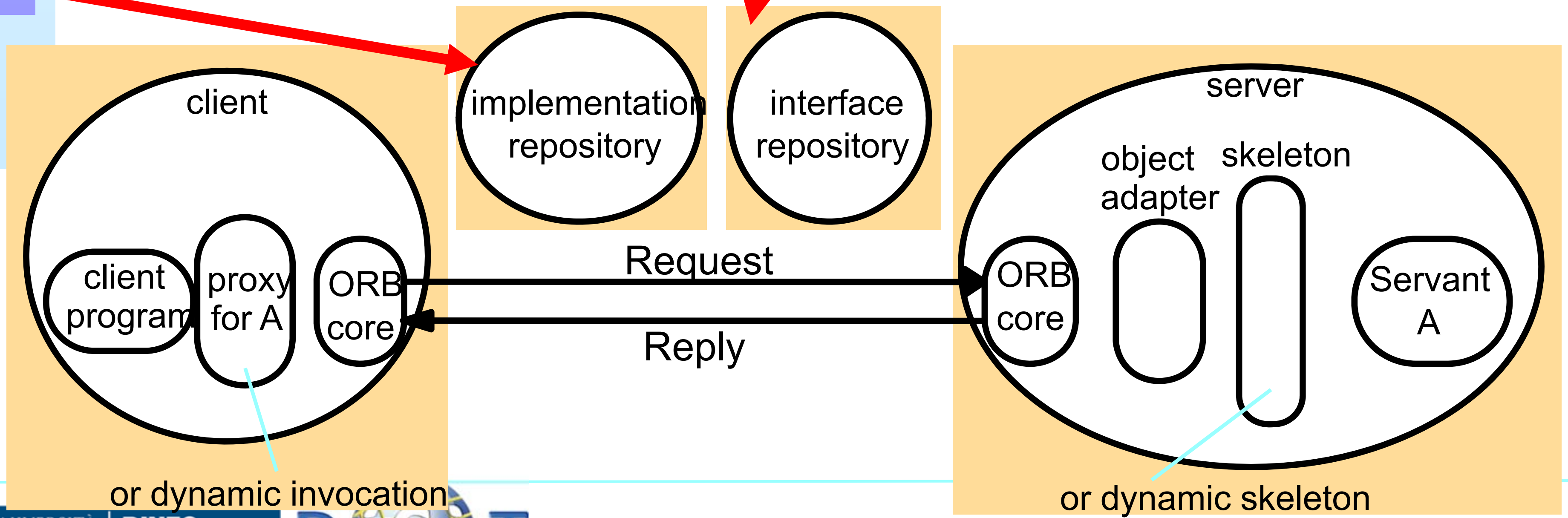
6. *Si noti che il ritorno dei parametri passa sempre dallo skeleton che gestisce anche le eventuali condizioni di eccezione e il loro marshalling verso il client.*

OA, Object Adapter

- **The Object Adapter (OA)**
  - to interface an object's implementation with its ORB.
- Three **Object Adapters**.
  - **Basic Object Adapter (BOA)**
    - Provides CORBA objects with a **common set of methods**.
      - CORBA object's interface to the ORB
      - available in every ORB implementation
    - Includes user authentication, object activation, object persistence ,etc...
  - **Library Object Adapter (LOA)**
  - **Object-Oriented Database Adapter (OODA)**
    - Both LOA and OODA are useful for accessing objects in persistent storage

The main components of the CORBA architecture

- **Interface repository**
  - the interface repository provides information about registered IDL interfaces to clients and servers that require it.
- **Implementation repository**
  - activates registered servers on demand and locates running servers
  - uses the object adapter name to register and activate servers.
  - more about this later



Interface repository

- provides information about registered IDL interfaces
  - **for each interface gives:**
    - method names and the names and types of the arguments and exceptions. (a short version of "Manifesto")
  - It is a facility for **reflection** in CORBA
    - Having a remote reference to a CORBA object, it is possible to ask at the interface repository about its methods and their parameter types
    - the client can use the dynamic invocation interface to dynamically construct an invocation with suitable arguments and send it to the server
- the IDL compiler gives a Type ID to each IDL type, which is
  - included in remote object references
  - used also as a Type repository ID
- Applications that use static invocation with client proxies and IDL skeletons do not require an interface repository.
  - Not all ORBs provide an interface repository.

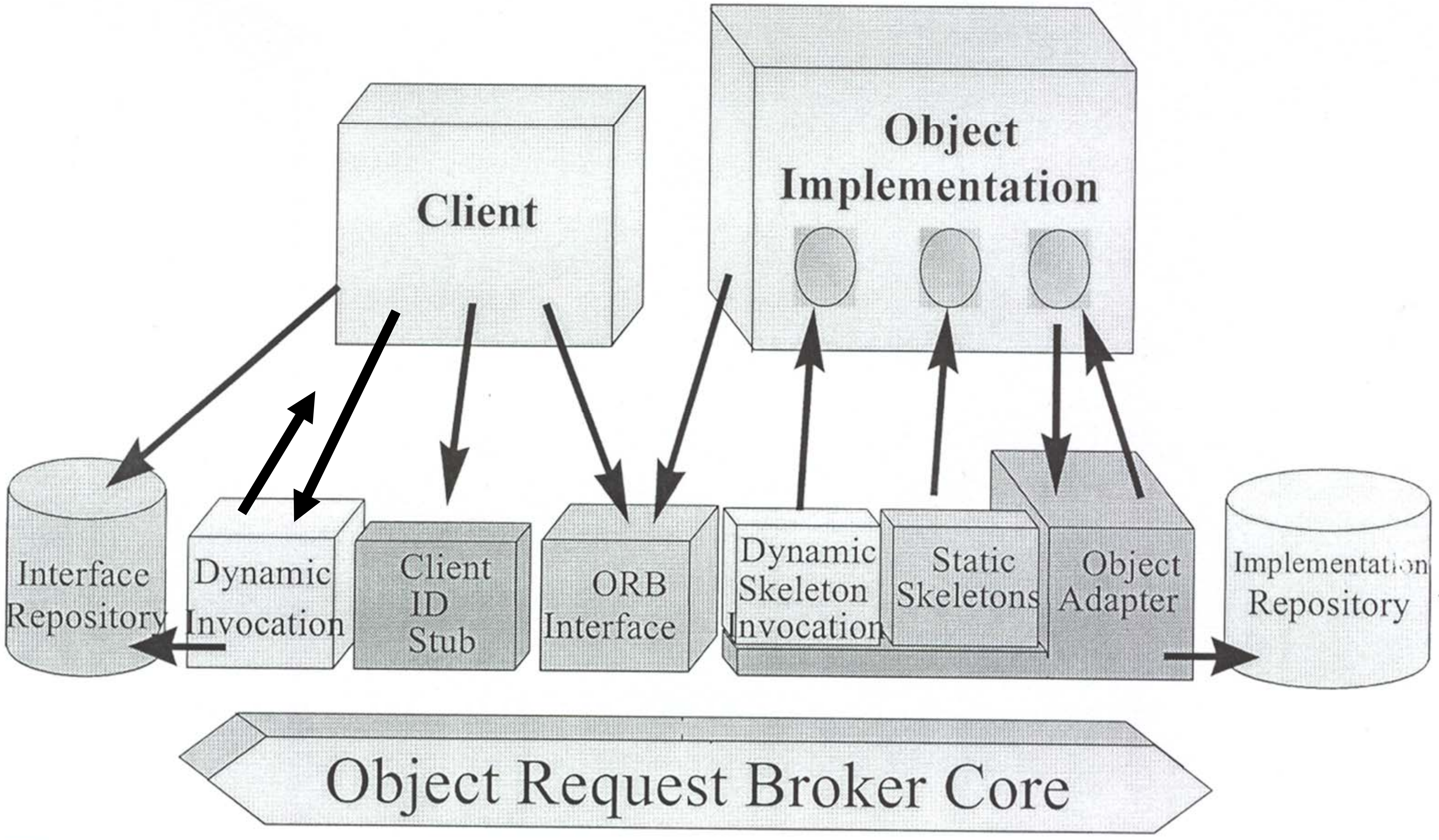
Concetto di Manifesto

- Descrizione della classe/“component software”

- Dovrebbe includere:
  - Nome, descrizione, creatore, produttore, data, versione, sistema operativo, etc.
  - Interfaccia di uso, o interfacce per l’uso, metodi e loro signature, etc.
  - Informazioni di trading: costo, DRM, location per il download di aggiornamenti, scadenze, etc.
  - dipendendenze da altri componenti, lib, dll, etc.

- In CORBA in concetto di manifesto e’ limitato:
  - L’ORB o chi per lui non e’ in grado di prendere decisioni su quali diverse implementazioni scegliere, etc.. Sulla base di informazioni disponibili
  - Le interfacce/implementazioni devono essere note agli ORB non possono arrivare dall’esterno del sistema, non possono essere caricate dinamicamente nel middleware.

ORB-Structure



ORB-invocation, Client

- **ORB Interface**
  - Identification of the objects
  - String to objects and viceversa (Marshalling/unmarshalling)
- **Client IDL Stubs**
  - Static interface to object services, precompiled stubs
- **Dynamic Invocation Interface, DII**
  - Permette di identificare i metodi che possono essere chiamati a run-time.
  - CORBA permette di identificare i metadati e l'interfaccia dei servizi degli oggetti non noti al client ma noti al MW
- **Interface Repository**
  - Database con tutte le interfacce possibili e registrate in base agli oggetti disponibili

ORB-provider part, Server

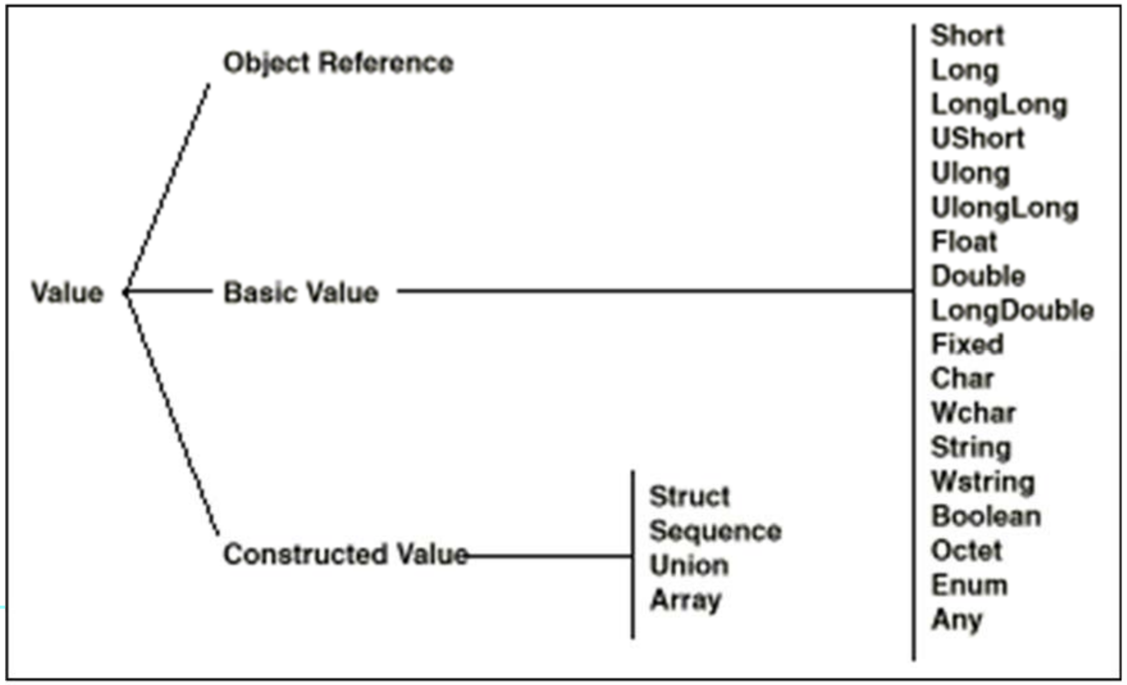
- **ORB Interface verso il Server**
  - Come quella da lato client
- **Static Skeleton equivalente al Server IDL Stubs**
  - Interfaccia statica dei servizi esportati dal server con IDL
- **Dynamic Skeleton Interface, DSI**
  - Interfaccia dynamica per la pubblicazione di servizi dinamici
  - Interface repository
- **Object Adapter / Dispatcher**
  - Accetta richieste di servizio per il Server
  - Istanzia oggetti distribuiti e gli assegna richieste
  - Fa uso del Implementation Repository
- **IR, Implementation Repository**
  - Tabelle di classi e loro ID

CORBA Characteristics

- Object-Oriented Programming
- Support multiple languages
  - Official: JAVA, C, C++, Smalltalk, COBOL
  - Also: eiffel, modula, perl, TCL, Python, etc.

What are Objects in CORBA !!

- **Objects are abstract**: not realized by any particular technology
  - An object system is a collection of objects that isolates the requestor of services (clients) from the providers of services by a well-defined **encapsulating interface**
- **Objects "talk" through requests**: operation, target object, zero or more parameters, optional request context
- **Objects are described** with interfaces
  - operations (methods)
  - attributes (properties)
  - Standard data types are supported
    - object references
    - Any



CORBA and IDL

- **Interface Definition Language (IDL)**
  - The OMG-standard language for defining the interfaces for all CORBA objects.
  - An IDL interface declares a set of operations, exceptions, and attributes.
  - Each operation has a signature, which defines its name, parameters, result and exceptions.
  - Format of messages, external data representation in CDR

- **Below there is the**
  - **Internet InterORB Protocol (IIOP)**
    The OMG-specified network protocol for communicating between ORBs fo different vendors. Based on TCP.

Interface Definition Language

- Language neutral specification

```
interface Polynomial : MathObject {
   sequence<Monomial> monomials;
   int rank;
   Polynomial add(in Polynomial p);
};
```

- Mappings to several languages
- Tools (compilers) generate stubs and skeletons in various languages

Note. No way to know at run-time which interfaces an objects provides: IDL is compiled away

Dynamic taking of an interface, but it has to be created in advance

CORBA

Interface Definition Language (IDL)

- General Properties of IDL
  - Case sensitive
  - Definition syntax is the same as C++ definition syntax.
  - Assumes the existence of a C processor to process constructs such as macro definitions and conditional complication
  - An example; The Module:

```
module Bank {
        interface Customer {
        ….
        };
        interface Account {
        ….
        };
        ….
};
```

CORBA

Interface Definition Language (IDL)

- Primitive types: void, Boolean, char, wchar,
- Floating point types: float, double and long double
- Integer types: long, long long, unsigned long etc.
- Constructed types: enum, struct, union, etc.
- The interface type: in, out, inout
- Attributes: readonly..
- Other IDL constructs
  - typedef
  - forward declaration
- Container types: sequence, array.
- The Exception type
- The Any type
- The TypeCode Pseudotype

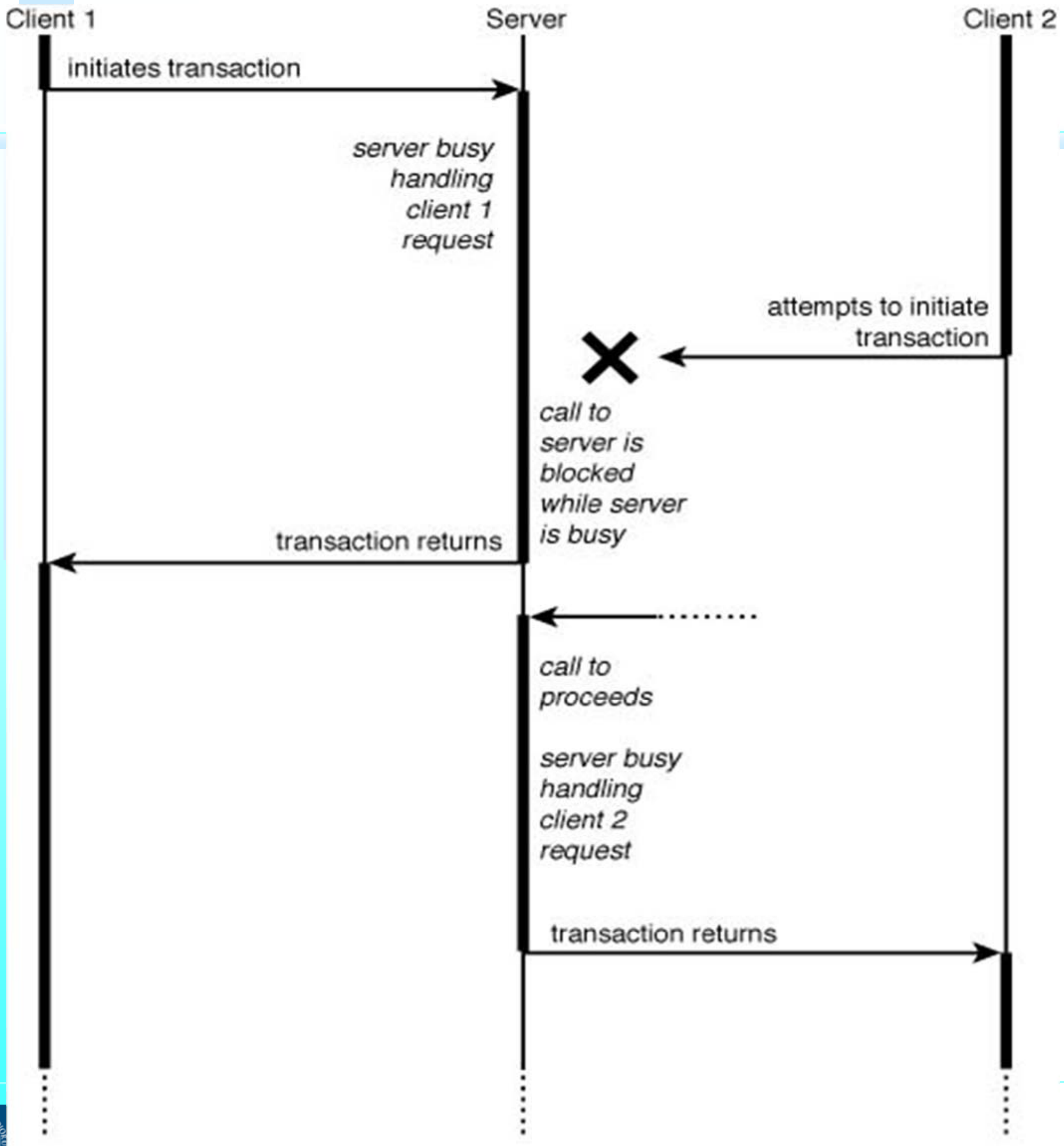
Building a CORBA Application

- Step1: Write IDL interfaces for Server: Server.idl
- Step2: Compile IDL file and generate Server_c.cpp and Server_s.ccp
- Step3: Write server implementation in C++: ServerMain.ccp
- Step4: Compile the ServerMain.cpp with the files created by IDL
- Step5: Write IDL interface for Client: Client.idl (se diversa da Server.idl)
- Step6: Compile client.idl and generate associated java files such as ServerSymbolHelper.java and ServerSymbolListHelper.java etc.
- Step7: Write client implementation in Java
- Step8: Compile client implementation and helper files together
- Step9: Run server and client programs together

CORBA Design Issues

- **Single-Threaded** Applications:
  - Common and Easy.
- **Multi-Threaded** Applications: Limited
  - Not all the Operating systems supports it.
  - Not all the developers are using the later versions of OSs.
  - Introduces new issues: the need to manage concurrent access to objects.

- Server Applications
- Client Applications
- Mixed Server/Client Applications
- Object Life Time


Client 1
Server
Client 2
initiates transaction
server busy handling client 1 request
attempts to initiate transaction
call to server is blocked while server is busy
transaction returns
call to proceeds
server busy handling client 2 request
transaction returns

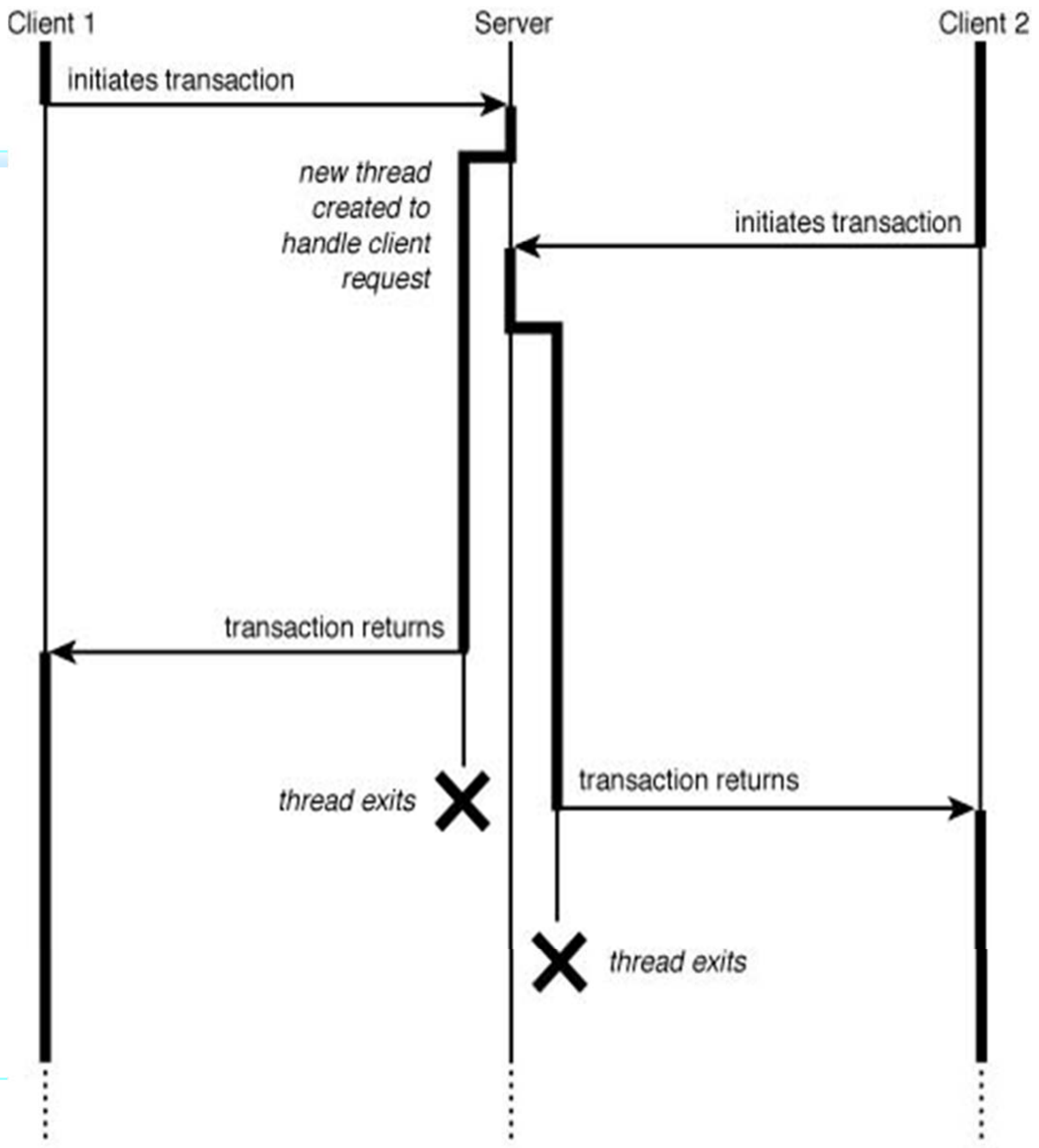
Single Thread

Serialize

CORBA

Multi Thread

Concurrent



UNIVERSITÀ DEGLI STUDI FIRENZE DINFO Dipartimento di Ingegneria dell'Informazione DISIT

Reference

- Visibroker, from Visigenic
  - ♣ http://www.cse.cuhk.edu.hk/~csc5340/material/vbj40java-reference.pdf
- Orbix from Iona
  - ♣ http://www.cse.cuhk.edu.hk/~csc5340/material/OrbixProgrammersGuide.pdf
- CORBA FAQ
  - ♣ http://www.omg.org/gettingstarted/corbafaq.htm