Università di Roma “RomaTre”

Introduzione a CORBA

Claudio Morgia

1999

Sommario (1)

- Architetture distribuite
- Object Request Broker
- Contratti
  - IDL
  - IR + DSI + DII
- Strutture Client, Object Implementation and Adapter
- GIOP/IIOP (General/Internet Inter-ORB Pr)

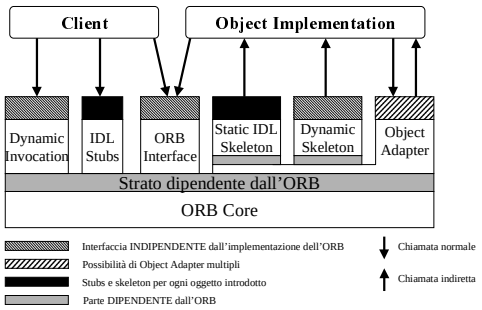
Sommario (2)

- Mapping
  - COM/OLE Automation
  - C/C++
  - Java
- Servizi
- Facilities
- Domini

Architetture distribuite

- Client/Server
- Three-tier
- Java RMI
- (D)COM/OLE Automation
- Agent based

Object Request Broker



Object Request Broker

- L’ORB si fa carico di
  - localizzare l’Object Implementation
  - preparare gli argomenti della chiamata
  - trasmettere la richiesta
  - restituire il/i risultato/i trasmesso/i
- Nessuna differenza tra IDL e Dynamic Invocation (dal punto di vista dell’Object Implementor).

Object Request Broker

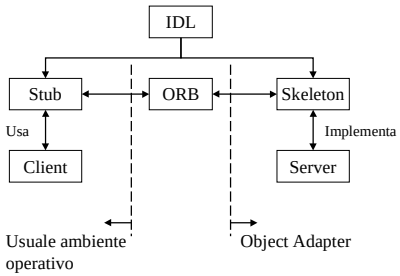
- Interfacce ovunque.
  - Tra ORB Core e moduli
  - Tra Client ed ORB: IDL e Dynamic Stub
  - Tra Object Implementation ed ORB: Static/Dynamic Skeleton ed Object Adapter
- Interfacce come contratti tra:
  - chi richiede un servizio (*stub*)
  - chi eroga un servizio (*skeleton*)

IDL

- Interface Definition Language di OMG
- Mapping verso altri linguaggi
- IDL come generatore di contratti (*stub* e *skeleton*)
- Mascheramento della trasmissione e della distribuzione (marshalling, serialization, transmission, location, ecc.)

IR+DII+DSI

- Lato client
  - Interface Repository
  - Dynamic Invocation Interface
- Lato object implementor
  - Dynamic Skeleton Interface
  - Implementation Repository
  - Object Adapter (immagine dell'ORB)



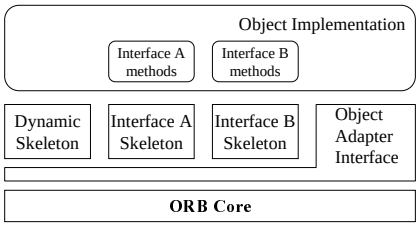
Uso di IDL

- Generazione di stub e skeleton da file IDL.
- Il Server *implementa* l'interfaccia.
- Il Client *ottiene riferimenti* ad oggetti che implementano l'interfaccia (tramite l'ORB).
- La trasmissione è globalmente gestita dall'ORB in maniera trasparente.

IR+DSI+DII

- Reperimento descrizione oggetto tramite IR.
- Localizzazione metodo da invocare.
- Generazione di un oggetto *Request* generico.
- Generazione argomenti in base ad analisi descrizione.
- Invocazione del metodo.

Object Adapter



Services

- Naming: localizzazione per nome
- Trading: localizzazione per caratteristiche
- Event: modelli ad eventi
- Externalization: serializzazione
- Persistent storage: storage su DB distribuiti
- Transaction: transazioni generalizzate
- Time: sincronizzazione temporale e trigger

Services

- Concurrency: gestione concorrenza distribuita.
- Relationship: modelli E-R distribuiti ed estesi ad oggetti.
- Life Cycle: ciclo di vita distribuito.
- Query: interrogazioni su basi di oggetti.
- Licensing: gestione licenze.
- Property: attributi esterni associati agli oggetti.
- Security: identificazione, autenticazione, autorizzazione, auditing, canali criptati.

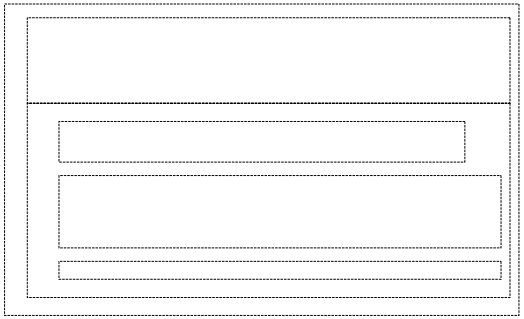
Facilities

- User interface
  - windowing and rendering
  - compound representation (MDI)
  - user support (sessioni)
  - desktop management
  - scripting
- Information management
  - information exchange
  - data encoding
- Task management (workflow, agent, automation)

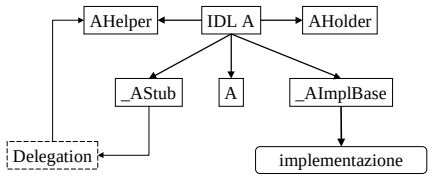
Esempio di IDL (Java)

```
module modulo {

   exception Ex {
        string reason;
   };

   interface A {

        attribute string sAttr;

        int metodo(in long arg1,
                   inout long arg2,
                   out any result)
        raises(Ex);

        void semplice(in int arg);
   };
};
```

```
package modulo;
final public class Ex extends
org.omg.CORBA.UserException {
        public String reason;
        public Ex() {..};
        public Ex(String r) {..};
}
public interface A extends
org.omg.CORBA.Object {
        String sAttr();
        void sAttr(String);

        int metodo(
                long arg1,
                IntHolder arg2,
                AnyHolder result
        ) throws(modulo.Ex);

        void semplice(long arg);
}
```



Schema implementativo



Implementazione Servant (1)

```
import java.io.*;
import org.omg.CORBA.*;

package modulo;

public class AImplementation extends _AImplBase {
 protected String sAttr;

 public AImplementation(String name) {
  super(name);
 }

 public void sAttr(String s) {
  sAttr=s;
 }

 public String sAttr() {
  return sAttr;
 }

 public void semplice(int arg) {
  // fai qualcosa
 }
 public int metodo(int arg1,
                   org.omg.CORBA.IntHolder arg2,
                   org.omg.CORBA.AnyHolder result)
                  throws modulo.Ex {
  if ((arg1>40) && arg2.value<20)) {
    arg2.value+=arg1;
    result.value.insert_string(“stringa”);
  }
 }
```

Implementazione Servant (2)

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

            Inizializzazione ORB e NameService
  ORB orb=ORB.init(args,null);
  BOA boa=orb.BOA_init();
  org.omg.CORBA.Object obj;
  obj=orb.resolve_initial_reference("NameService");
  NamingContext nameServer=NamingContextHelper.narrow(obj);

              Creazione esplicita servente
  AImplementation server=new AImplementation("A Server");

                      Registrazione
  NameComponent[] nome={new NameComponent("A Server","")};
  nameServer.rebind(nome,server);

              Preparazione Object Adapter
  boa.obj_is_ready(server);

                 Attivazione servente
  boa.impl_is_ready();
 }
}
```

Implementazione Client

```
import java.lang.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;

public class AClient {
 public static void main(String[] args) {

                  Inizializzazione
  ORB orb=ORB.init(args,null);
  org.omg.CORBA.Object obj;
  obj=orb.resolve_initial_references("NameService");
  NamingContext nameServer=NamingContextHelper.narrow(obj);

                   Localizzazione
  NameComponent[] nome={new NameComponent("A Server","")};
  A server=AHelper.narrow(nameServer.resolve(nome));

                    Uso statico
  server.semplice(5);

                  Uso degli holder
  IntHolder arg2=new IntHolder(100);

                   Any come short
  Any subarg3=new Any();
  subarg3.insert_short(3);
  AnyHolder arg3=new AnyHolder(subarg3);
  int result=server.metodo(3,arg2,arg3);
  try {

             Any trasformato in stringa
    String str=arg3.extract_string();
    System.out.println("Il risultato è la stringa "+str);
  } catch(BAD_OPERATION bo) {
    // non e' una stringa
  }
 }
}
```

Implementazione Client tramite DII

```
import java.lang.*;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;

public class AClient {
 public static void main(String[] args) {

                  Inizializzazione
  ORB orb=ORB.init(args,null);
  org.omg.CORBA.Object obj;
  obj=orb.resolve_initial_references("NameService");
  NamingContext nameServer=NamingContextHelper.narrow(obj);

                   Localizzazione
  NameComponent[] nome={new NameComponent("A Server","")};
  obj=nameServer.resolve(nome);

          Generazione richiesta con introspezione
  Request req=obj._request("semplice");
  NVList args=req.arguments();
  Any arg=new Any();
  arg.insert_short(5);
  args.add_value("arg",arg,org.omg.CORBA.ARG_IN.value);

                        Invocazione
  req.send_oneway();
  }
 }
}
```