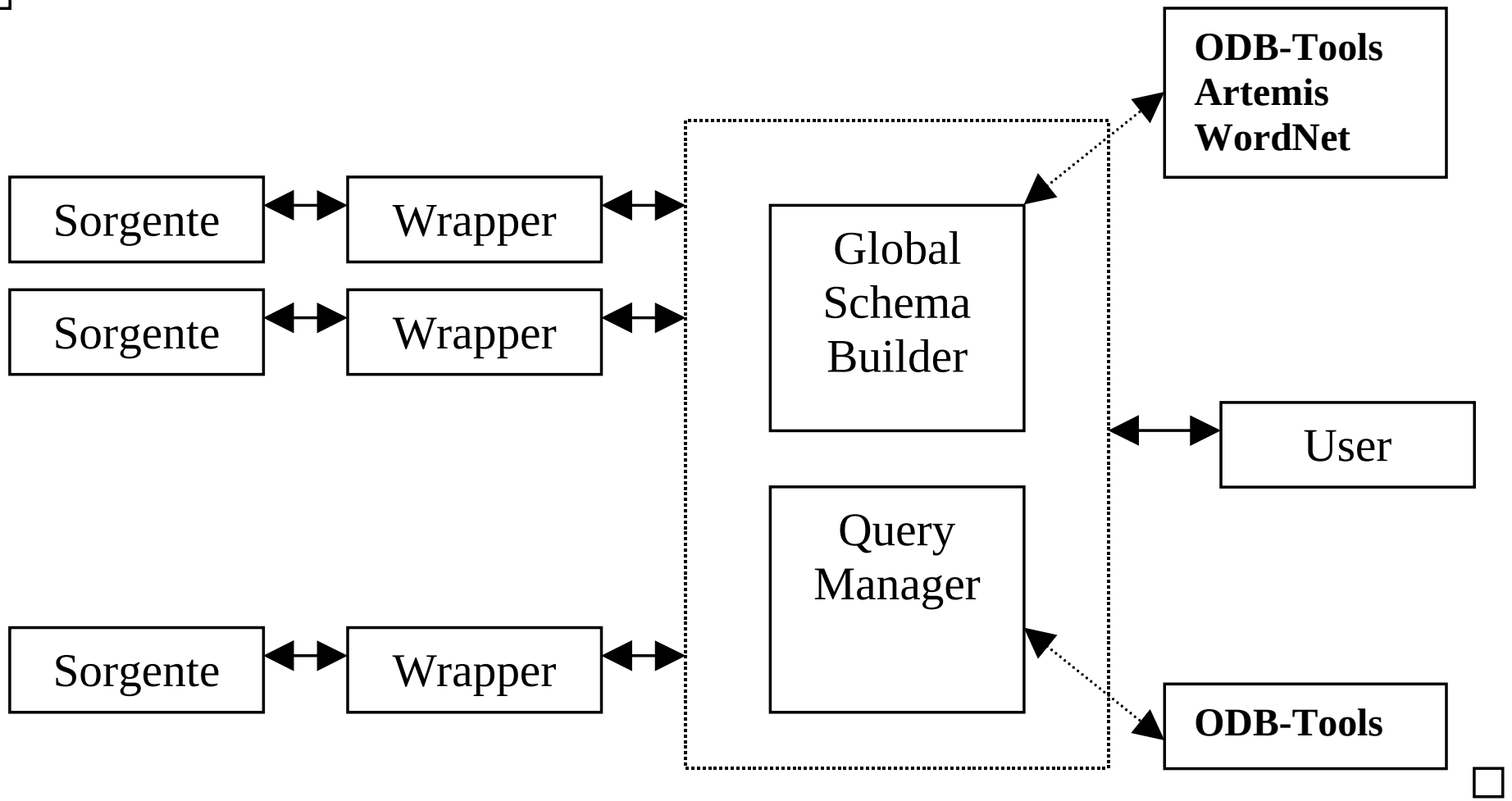
CORBA in MOMIS

MOMIS: Mediator envirOnment for Multiple Information Sources



Perché CORBA in MOMIS

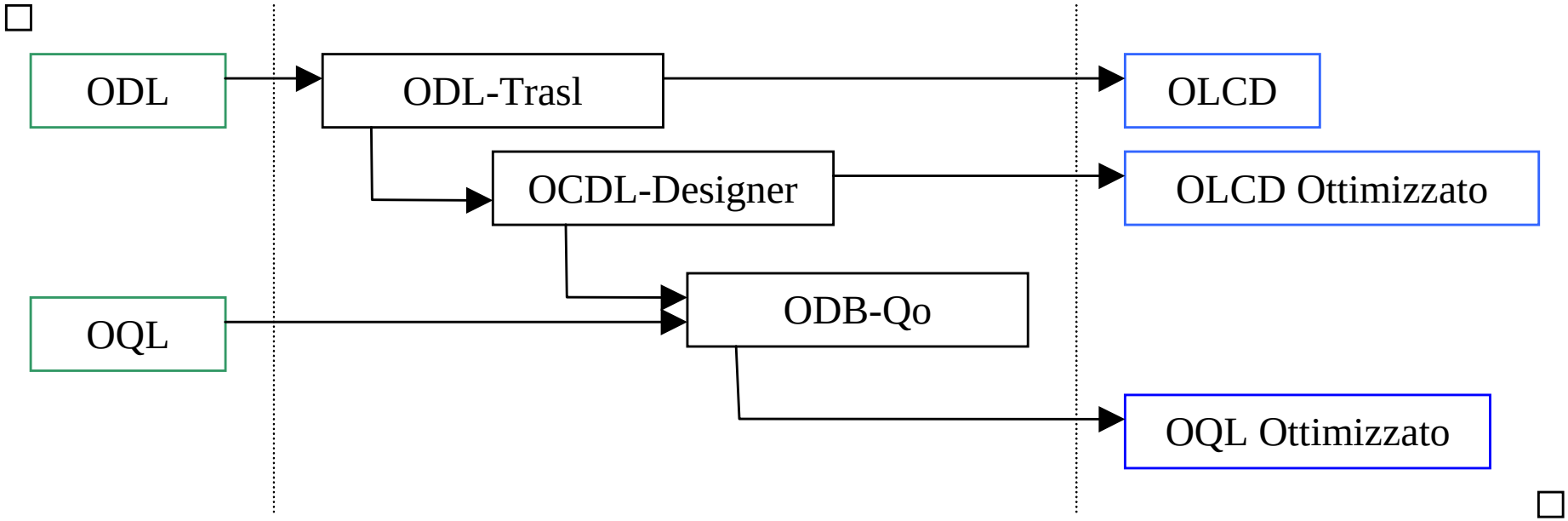
CORBA è una *tecnologia per l'integrazione* e
MOMIS allo stato attuale necessita di essere integrato.

CORBA è ad oggetti e
MOMIS allo stato attuale necessita di essere ri-modellato per ridurne la complessità:

- esistono metodologie consolidate per la rappresentazione e progettazione di sistemi ad oggetti (OMT o UML)
- per utilizzare un oggetto è sufficiente conoscerne l'interfaccia *pubblica*

CORBA è uno standard che sta prendendo piede.

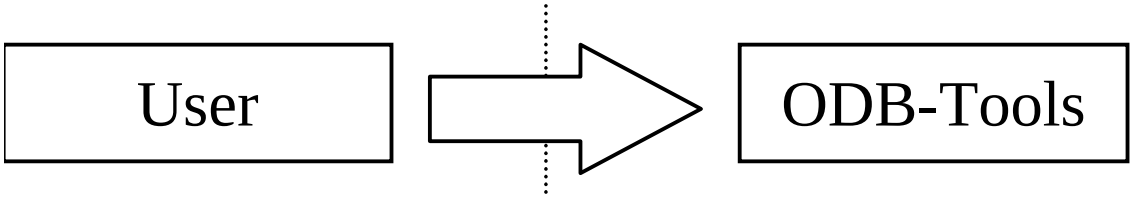
L’esempio ODB-Tools



`s.odl`	`odl_trasl s`	**s.sc** schema in OLCD **s.odl.vf** schema in Visual
`s.sc`	`ocdl-designer s`	**s.fc** schema canonico **s.ft** fattori **s.sb** subsumption **s.vf** visual form

`s.oql`
`s.fc`
`s.ft`
	`odbqo s`	**s.oql.fc, s.oql.ic.html**
		s.oql.out, s.oql.sb, s.oql.vf
		s.oql.vf.last, STD-OUT

ODB-Tools Visione object




```
interface OdbTools {
   string translate_Odl_Olcd ( in string odl, out string olcd );
   string translate_Odl_Olcd_vf ( in string odl, out string olcd,  out string vf );
   string validate_Odl ( in string odl, out string fc );
   string validate_OdlSS ( out string fc );
   string validate_OdlSS_vf ( out string fc, out string vf );
   string optimize_Oql ( in string odl, in string oql, out string stdOut  );
   string optimize_OqlSS ( in string oql );
   string optimize_OqlSS_vf ( in string oql, out string vf );
   long killServer();
 };
```

Utilizzo di *ODB-Tools* CORBA

Invocazione del Query Optmizer da Java

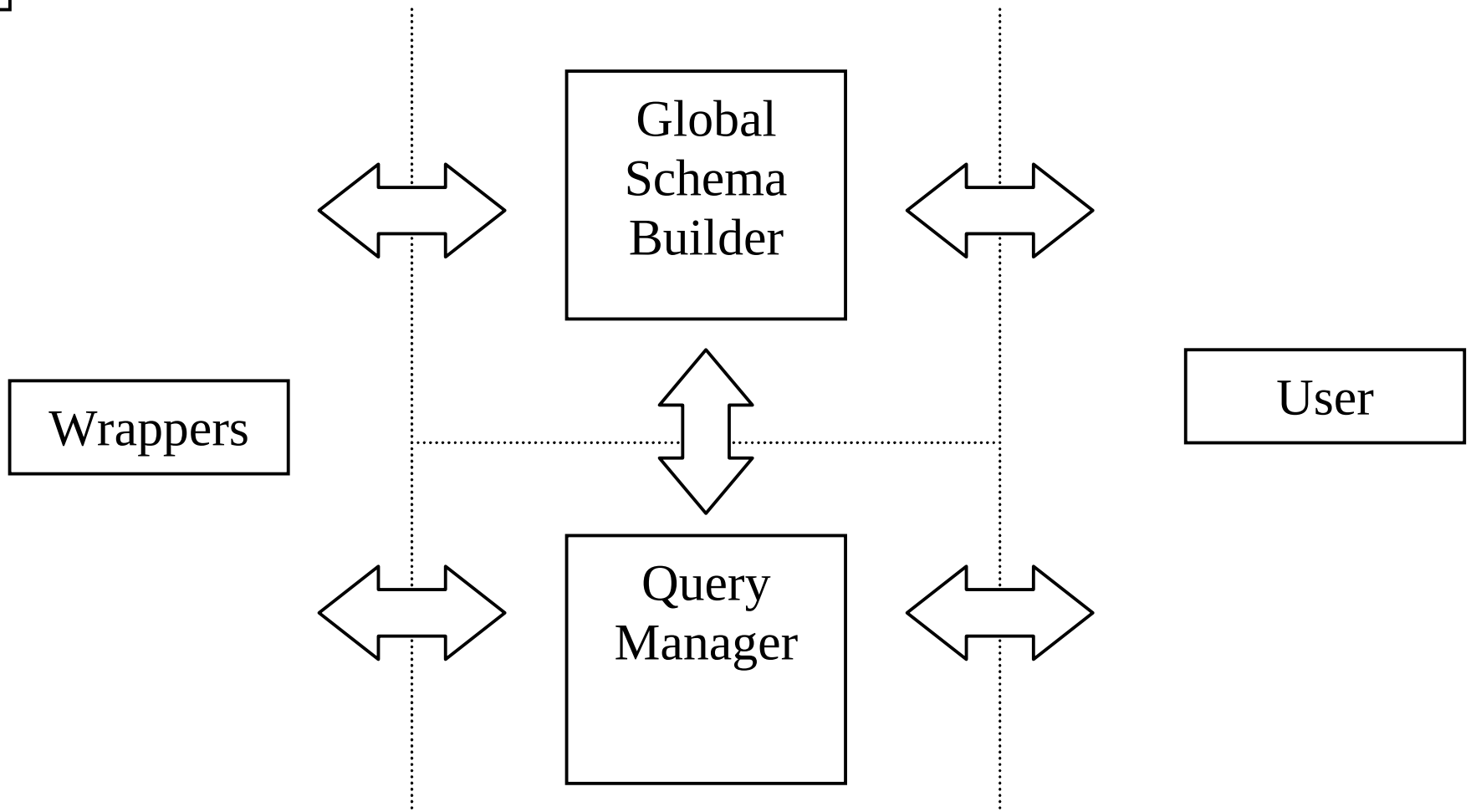
s = myRef.optimize_Oql(stringOdl , stringOql, stringStdOut);

Oppure

myRef.translate_Odl_Olcdql(stringOdl , stringOlcd);
myRef.validate_OdlSS(stringFc);
myRef.optimize_OqlSS (stringOql);

Dove l'oggetto *myRef* è il riferimento ad un oggetto CORBA di tipo *OdbTools* residente su *sparc20.dsi.unimo.it* che si preoccupa di lanciare i comandi *odl_trasl*, *ocdl-designer* e *odbqo* e gestisce anche i file temporanei necessari alla corretta esecuzione di ODB-Tools.

MOMIS Visione Object



Significa isolare per ogni *entità* metodi d'interfaccia che corrispondono alle funzionalità fornite.

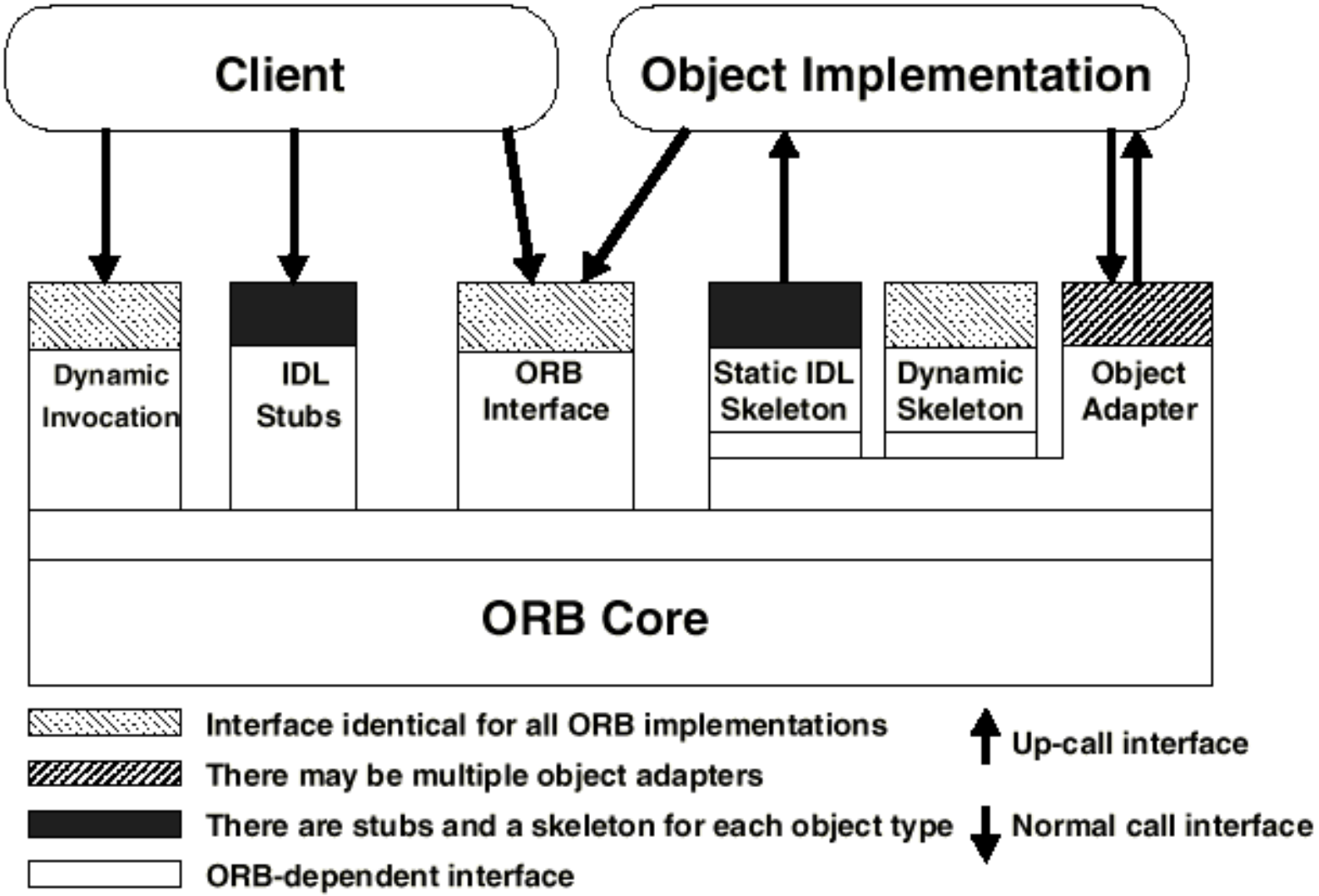
Interfaccia Momis OO

```
interface Wrapper{
    string getOdlDescription();
    MomisCollection executeQuery(string oql);
};

interface MomisFactory{
    GlobalSchema newQueryManager(
        MomisList, GlobalSchema schema);
};

interface QueryManager{
    MomisCollection executeQuery(string oql);
};
```

Questo è un punto da studiare approfonditamente

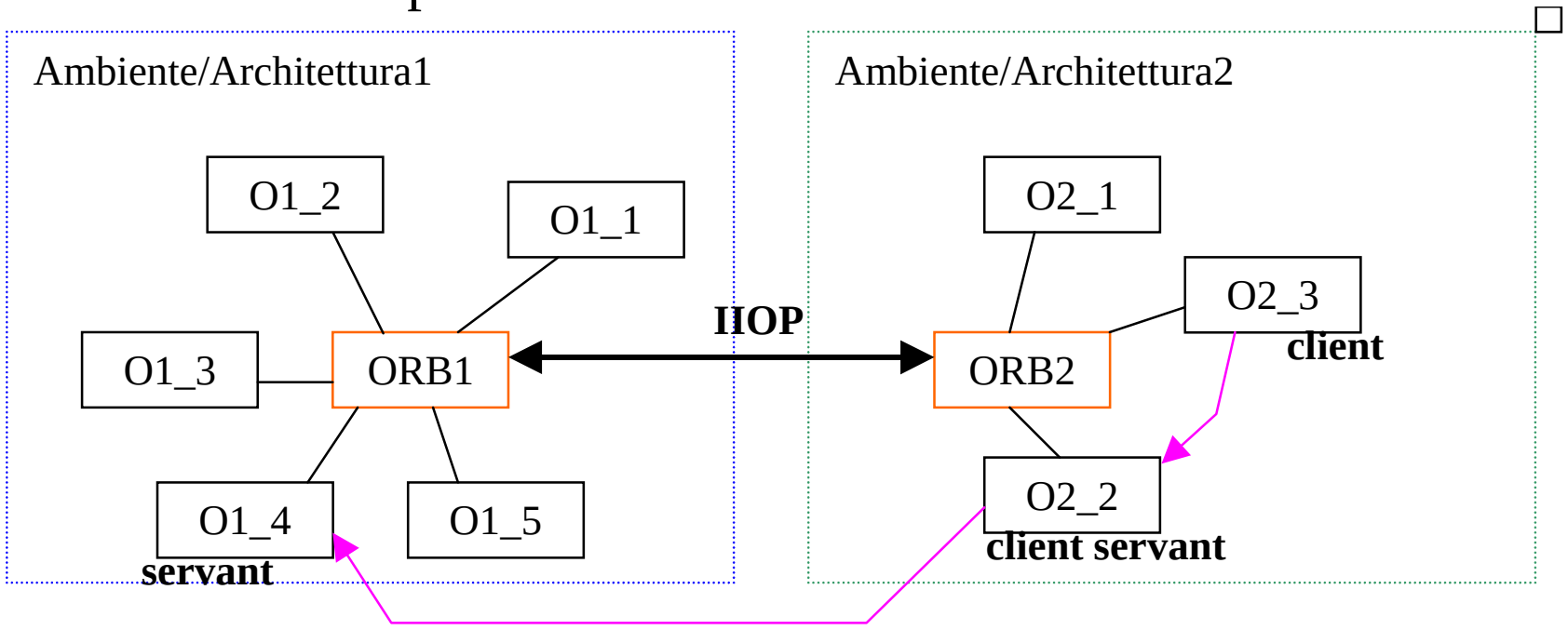
CORBA: Common ORB Architecture

Purtroppo CORBA è piuttosto complesso...



CORBA: Common ORB Architecture (2)

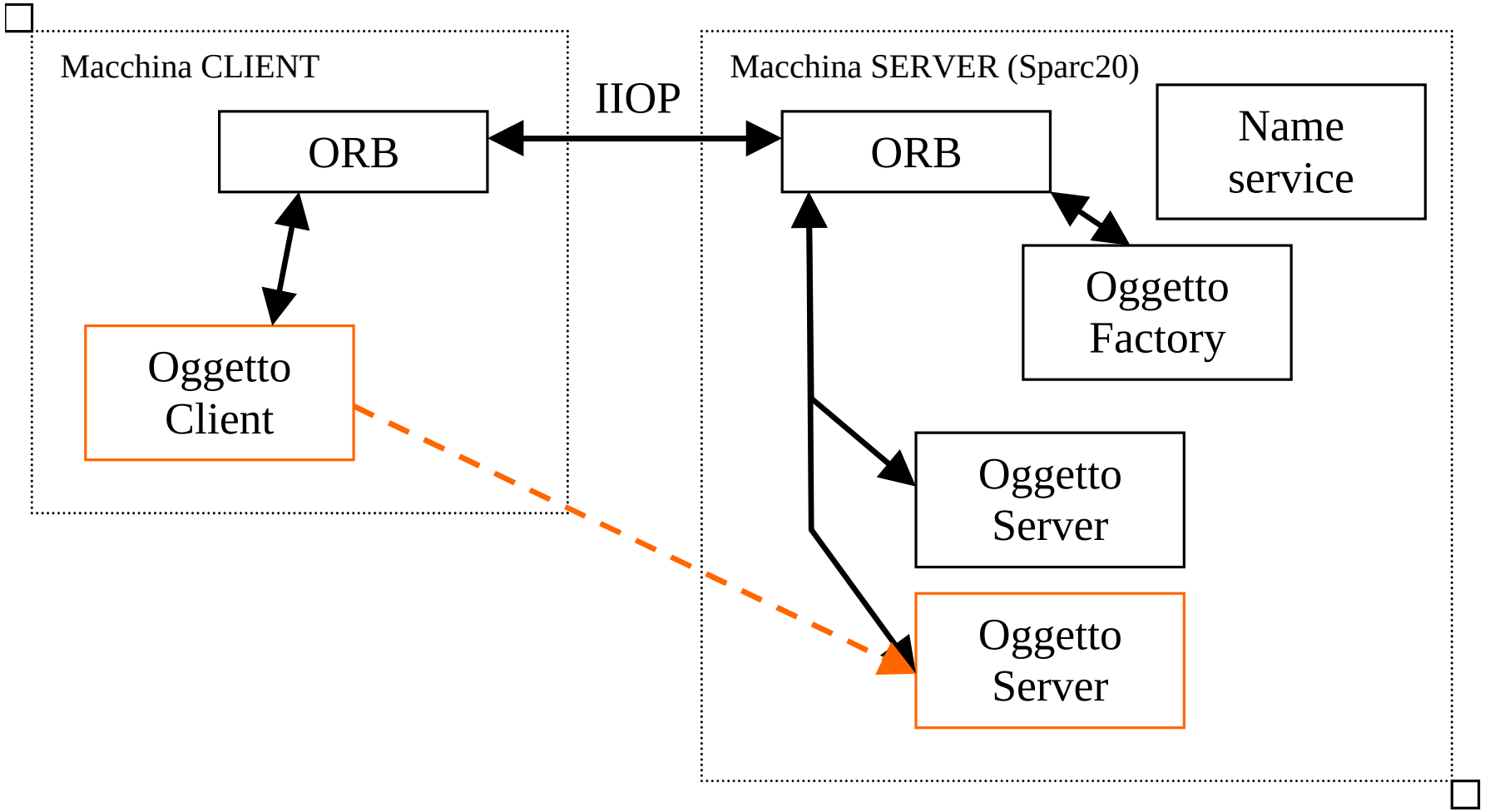
ORB sta per Object Request Broker ed è un *oggetto* in grado di gestire le chiamate di metodi tra oggetti registrati presso tale ORB oppure registrati presso altri ORB. La comunicazione tra ORB è standardizzata, si utilizza il IIOP (Internet Inter Orb Protocol) parlare con altri ORB.
Vi possono essere diverse implementazioni di ORB.
Un OBB *deve* implementare un numero limitato di servizi.



Servizi CORBA previsti dallo standard

Naming Service
Event Service
Life Cycle Service
Persistent Object Service
Transaction Service
Concurrency Control Service
Relationship Service
Externalization Service
Query Service
Licensing Service
Property Service
Time Service
Security Service
Object Trader Service
Object Collections Service

Connessione CORBA ODB-Tools



Client e Server devono condividere la medesima interfaccia (IDL)

Nomenclatura oggetti CORBA

---------- *oggetti client* ----------

CLIENT: È l'oggetto che utilizza i servizi offerti dagli oggetti CORBA *server*.

---------- *oggetti server* ----------

SERVANT: Si tratta della categoria di oggetti in grado di fornire il servizio richiesto. Ad esempio tutte le funzionalità di ODB-Tools sono dichiarate come metodi di un oggetto servant.
Normalmente per ogni oggetto *client* è creato un oggetto *servant*.

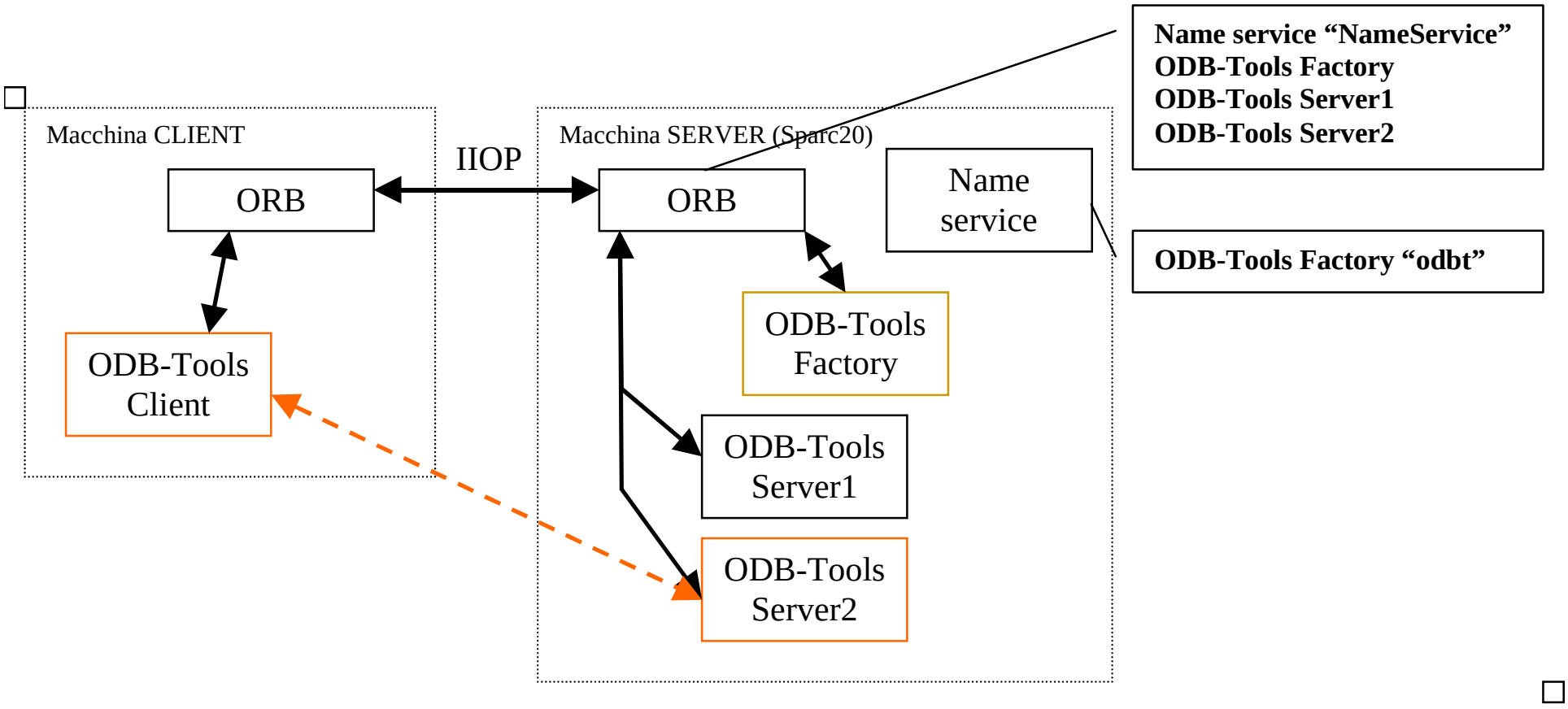
FACTORY: Si tratta di un oggetto di riferimento attraverso il quale è possibile creare nuovi oggetti *servant*.
Questo oggetto è visibile tramite il naming service

Nomenclatura oggetti CORBA(2)

CLIENT: Qualunque oggetto può essere client di altri oggetti corba. Normalmente l'applicazione attraverso la quale l'utente utilizza il sistema è un tipico esempio di applicazione *solo client*.

FACTORY: Normalmente si tratta di oggetti *residenti*, di *demoni* fatti partire una volta e si tratta di *punti di rifermento* per gli oggetti client.
Questo oggetto è visibile tramite il naming service e si tratta di una sorta di punto di accesso al servizio. Oggetti *Factory* solitamente generano nuovi oggetti *servant* che sono quelli che forniscono il servizio o sono oggetti di *comodo* per il trasporto di informazioni (es: liste o resultSet). Per questi ultimi occorre gestire un eventuale Time-out.

Compiti degli oggetti ODB-Tools



Interfaccia IDL ODB-Tools

`OdbToolApplication.idl`

```
module OdbToolsApplic {
  interface OdbTools {
     string translate_Odl_Olcd ( in string odl, out string olcd );
     ...
     string optimize_OqlSS_vf ( in string oql, out string vf );
     long killServer();
  };

  interface OdbToolsFactory
  {
     OdbTools newServant ( in string description );
  };
};
```

Inizializzazione Factory

```
try{
   orb = ORB.init({"-ORBInitialHost”, orbServerName,
                        "-ORBInitialPort";  orbPort}, null);
   OdbToolsFactory_server myRef =
      new OdbToolsFactory_server(“odbt”);
   orb.connect(myRef);
   org.omg.CORBA.Object objRef =
      orb.resolve_initial_references("NameService");
   NamingContext ncRef = NamingContextHelper.narrow(objRef);
   NameComponent nc = new NameComponent(“odbt”, "");
   NameComponent path[] = {nc}; ncRef.rebind(path, myRef);
   java.lang.Object sync = new java.lang.Object();
   synchronized(sync){
      while (true) {
            sync.wait(timeBetweenCeckTimeOut);
            myRef.checkTimedOut();
      }   } }
```

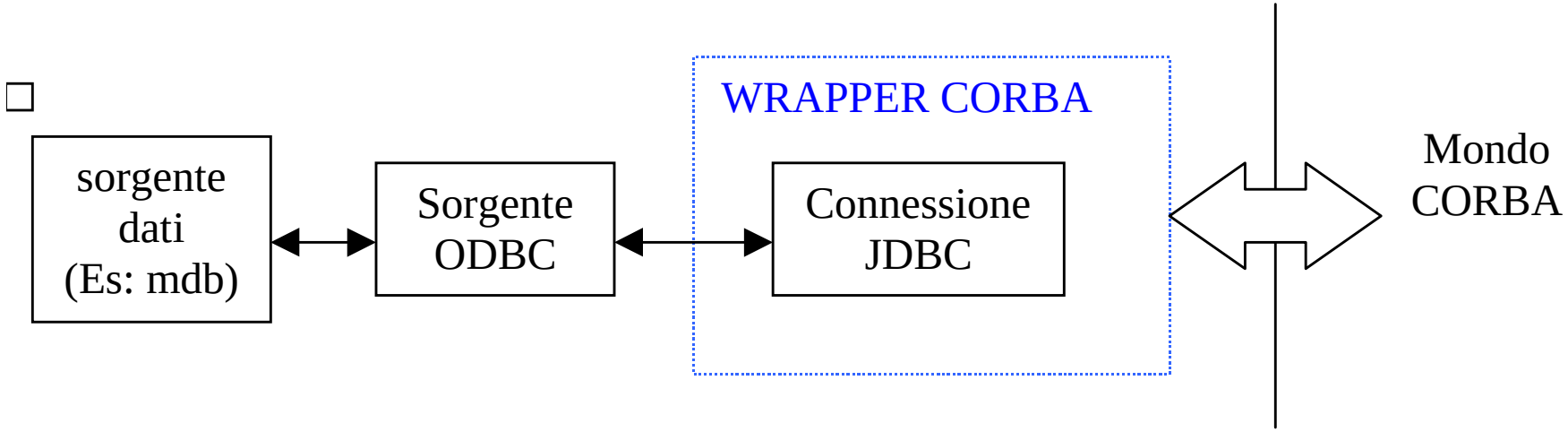
Inizializzazione Client

```
ORB orb;
org.omg.CORBA.Object objRef;  NamingContext ncRef;
NameComponent nc;             OdbToolsFactory factoryObj;
OdbTools myRef;               String serverName = "odbt";
try{
   orb = ORB.init(args, null);
   objRef = orb.resolve_initial_references("NameService");
   ncRef = NamingContextHelper.narrow(objRef);
   nc = new NameComponent(serverName, "");
   NameComponent path[] = {nc};
   factoryObj = OdbToolsFactoryHelper.narrow(ncRef.resolve(path));
   try {
      hn = "" + InetAddress.getLocalHost();
   } catch(Exception e1) { }
   myRef = factoryObj.newServant("DemoClient " + hn);
   …
   s = myRef.optimize_Oql(stingOdl, stringOql, stdOut);
   ...
```

Inizializzazione Servant

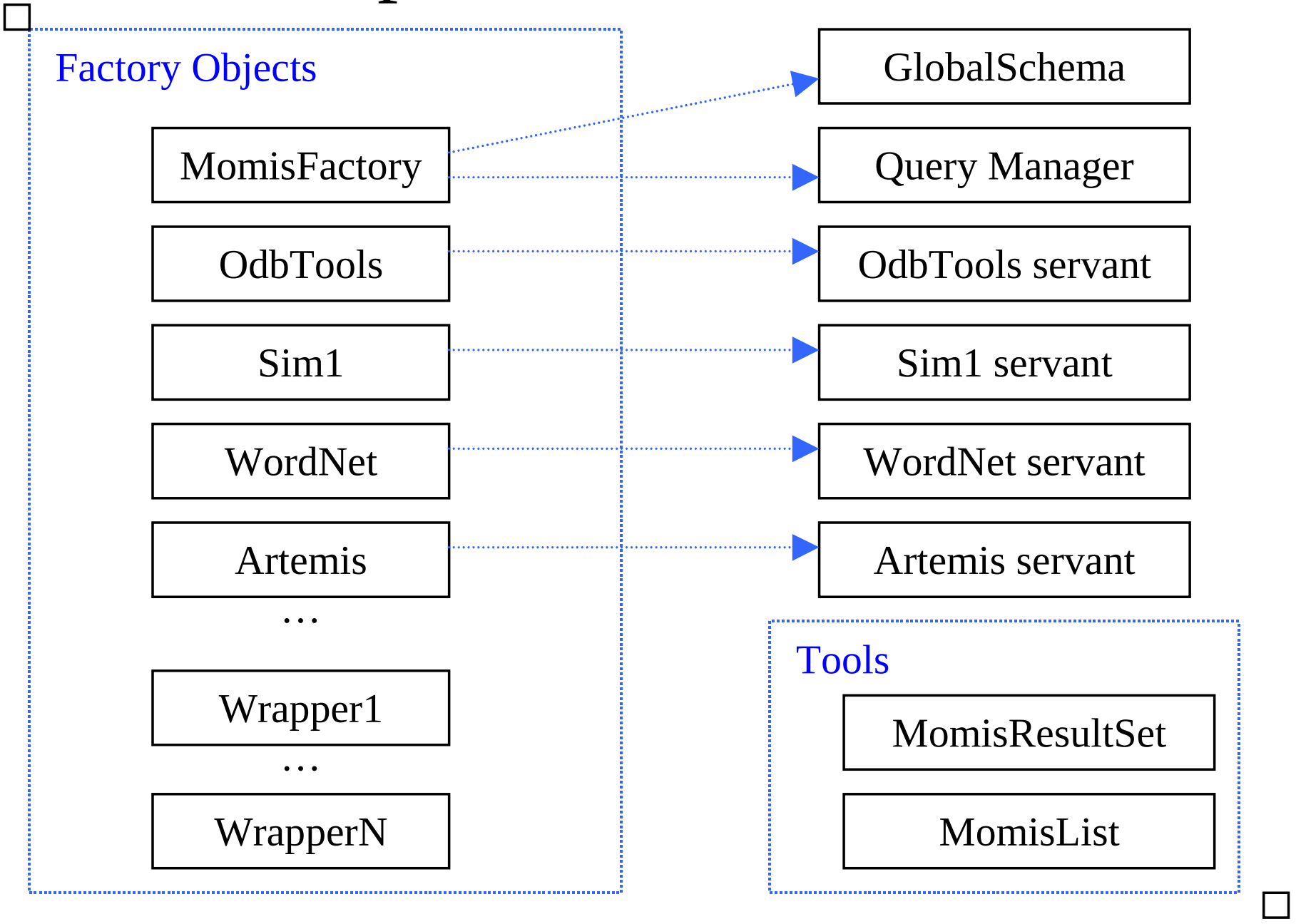
```
public OdbToolsApplic.OdbTools newServant (String description)   {
   OdbTools_server rv;
   long currentNumber = _servantId++;
   String filePrefixName = _name + "_" + currentNumber;
   // -----
   //
   // Creo un nuovo Servant !
   logScrivi("Creating new ODB-Tools servant [" + description +
            "], prefix: [" + filePrefixName + "]");
   rv = new OdbTools_server(filePrefixName, this);
   //
   // Inserisco il servant nella lista di time-out
   _timeOutManager.add(rv, timeBeforeTimeOut);
   //
   // Collego il servant all'ORB
   OdbToolsFactory_server.orb.connect(rv);
   return rv;
}
```

Proposta di *wrapper* sorgenti JDBC




```
interface Wrapper {
    /* Get the description of teh source in ODLi3. */
    string getDescription();
    /* Executes a Query OQL on the GlobalSchema specified */
    MomisResultSet runQuery( in string oql );
};
```

Proposta MOMIS OO



Operazioni da compiere

I compiti da svolgere sono:

- progettazione interfaccia **SchemaBuilder** (+interfaccia utente)
- progettazione interfaccia **Sim1**
- progettazione interfaccia **Artemis**

- tuning interfaccia **OdbTools** (solo se ritenuto utile)

- progettazione interfaccia **QueryManager**
- tuning del **Wrapper JDBC**

- progettazione interfaccia **WordNet** e wrapping in CORBA

Documentazione Progetto

Home Page di MOMIS
http://sparc20.dsi.unimo.it/Momis
http://sparc20.dsi.unimo.it/Momis/prototipo

Directory su sparc20
/export/home/httpd/htdocs/Momis/prototipo
/export/home/httpd/htdocs/Momis/prototipo/wrappers
/export/home/httpd/htdocs/Momis/prototipo/schemaBuilder
/export/home/httpd/htdocs/Momis/prototipo/queryManager
/export/home/httpd/htdocs/Momis/prototipo/wordnetInteraction

File indice:
index.html

Directory per lo sviluppo

Directory su sparc20
/export/home/progetti.comuni/Momis/CORBA_Momis
/export/home/progetti.comuni/Momis/CORBA_ODBTools

ODB-Tools Files:
Tools
TimeOut_able, TimeOutManager
UnixCommand
createDoc
Makefile (, doc, clean)

Il direttorio di sviluppo

/export/home/progetti.comuni/Momis/prototype

0setVarsCsh	Inizializza le variabili d'ambiente
0setVarsSh	Inizializza le variabili d'ambiente
CORBA_Momis/	Direttorio oggetti Momis (GlobalSchema, QM)
CORBA_ODBTools/	Direttorio interfaccia Odb-Tools
CORBA_wrapperJDBC/	Modulo wrapper JDBC
doc/	Documentazione generata con JavaDoc
Makefile	script per compilare i sorgenti
momis.conf	File di configurazione di tutti i moduli
MomisApplic.idl	Interfaccia per i vari oggetti MOMIS
MomisApplic/	Prodotto da IdlToJava da MomisApplic.idl
setMomisCLASSPATH	Sistema il classpath per lavorare al sistema MOMIS
startMomis	Script per far partire/fermare i vari moduli
tools/	Contiene classi di utilita' varia
trash/	Contiene file "inutili" o da cancellare
var/	Contiene i file di log dei vari moduli

Il direttorio di sviluppo Wrapper

/export/home/progetti.comuni/Momis/prototype/CORBA_wrapperJDBC

0note.txt	Note varie sul codice
createDoc	Script per la creazione della documentazione
Makefile	Script per compilare i sorgenti
WMomisResultSet.java	Implementazione di un oggetto MomisResultSet JDBC
Wrapper_server.java	Implementazione wrapper JDBC
WrapperClient.java	Esempio di client per il wrapper

```
java -cp $CLASSPATH Wrapper_server ../momis.conf

echo 'd
x
' | java -cp $CLASSPATH WrapperClient -ORBInitialHost
    sparc20.dsi.unimo.it -ORBInitialPort 1050

java -cp .;.. Wrapper_server ..\momis.conf
```

Il Makefile per il Wrapper

```
JC = javac -classpath $(CLASSPATH)
CC = gcc
JAVA_HOME = /usr/java

all:    server client
clean:
        rm -rf doc
        find . -name "*.class" | xargs rm -f
        find . -name "*~" | xargs rm -f

server:   Wrapper_server__compile
client:   WrapperClient__compile
Wrapper_server__compile:
        $(JC) Wrapper_server.java
WrapperClient__compile:
        $(JC) WrapperClient.java
```

Riferimenti CORBA e Java

Java Api
http://java.sun.com/products/jdk/1.2/docs/api/index.html

Java Tutorial
http://java.sun.com/docs/books/tutorial/index.html
http://java.sun.com/docs/books/tutorial/idl/index.html
Java IDL Documentation page (dalla pagina delle API)

Introduction to CORBA in Java
http://developer.java.sun.com/developer/onlineTraining/corba/corba.html

Object management Group
http://www.omg.org/

Corba su Sparc20 (specifica completa + services anche se un pò datati)
http://sparc20.dsi.unimo.it/documenti/omg

Consigli pratici

Configurazione del forward della posta in sistemi unix.
È sufficient utilizzare il file $HOME/.forward
vedere *http://sparc20.dsi.unimo.it/sparc20/usoSparc20.html*