

©2008 Component-based Systems, ICA, Linköping University. Some slides by courtesy of Olaf Adamiak, ICA / TU Dresden. Revised by C. Kessler, 2008-2010.

CORBA

[Szyperski, Chapter 13]

Overview, Goals

Basic interoperability:

- IDL
- ORB
- Object Adapter
- IOR
- GIOP/IOP
- Dynamic Calls

Trader Service

Evaluation of CORBA as a composition system

Following: CCM CORBA Component Model

Appendices:

- CORBA Services and CORBA Facilities
- CORBA, Web and Java

1

©2008 Component-based Systems, ICA, Linköping University. Some slides by courtesy of Olaf Adamiak, ICA / TU Dresden. Revised by C. Kessler, 2008-2010.

Background literature on CORBA

- F. Bolton: *Pure CORBA*. Sams Publishing, 2002.
Java and C++ examples
- M. Aleksy, A. Korthaus, M. Schader: *Implementing Distributed Systems with Java and CORBA*. Springer, 2005.
- Special issue of *Communications of the ACM* **41**(10), Oct. 1998.
All articles. Overview of CORBA 3.0.
- Tanenbaum, van Steen: *Distributed Systems*. Pearson, 2003.
Principles and paradigms.
- OMG: *CORBA 2.2 and CORBA 3.0 Specification*.
<http://www.omg.org>
See also further material from the OMG on the Web
- OMG: *CORBAfacilities: Common Object Facilities Specifications*.
<http://www.omg.org>

2

©2008 Component-based Systems, ICA, Linköping University. Some slides by courtesy of Olaf Adamiak, ICA / TU Dresden. Revised by C. Kessler, 2008-2010.

CORBA




- Common Object Request Broker Architecture®
- Founding year of the OMG (Object Management Group) 1989
- Goal: plug-and-play components everywhere
- CORBA 1.1 1991 (IDL, ORB, BOA)
- ODMG-93 (Standard for OO-databases)
- CORBA 2.0 1995.
Version 2 is a separate line, 2.2 and 2.4 are status quo
- CORBA 3.0 1999 (POA).
Current version (2005) is 3.0.3.

3

©2008 Component-based Systems, ICA, Linköping University. Some slides by courtesy of Olaf Adamiak, ICA / TU Dresden. Revised by C. Kessler, 2008-2010.

Ingredients of CORBA

- Component Model**
 - Components == classes (and objects), i.e., similar to object-oriented software. CORBA components have more component secrets.
- Basic interoperability**
 - Language interoperability by uniform interfaces description
 - Transparent distribution
 - Transparent network protocols
- CORBA Services**
- CORBA Facilities**
 - Horizontal (general-purpose) vs. vertical (domain-specific)
 - CORBA MOF

4

©2008 Component-based Systems, ICA, Linköping University. Some slides by courtesy of Olaf Adamiak, ICA / TU Dresden. Revised by C. Kessler, 2008-2010.

Corba's Hydrocephalus

- Corba is large**
 - Object Request Broker – 2000 pages of specification
 - Object Services – 300 pages
 - Common Facilities – 150 pages
- Technical reasons**
 - Clean detailed solution
 - Sometimes overkill
- Sociologic reasons**
 - OMG is large (over 800 partners) and heterogeneous
 - Standard covers a wide range
- Linguistic reasons**
 - Own language
 - Lots of unintuitive 3-capitals-names (OMG, ORB, IDL, ...)
 - Appears larger than necessary

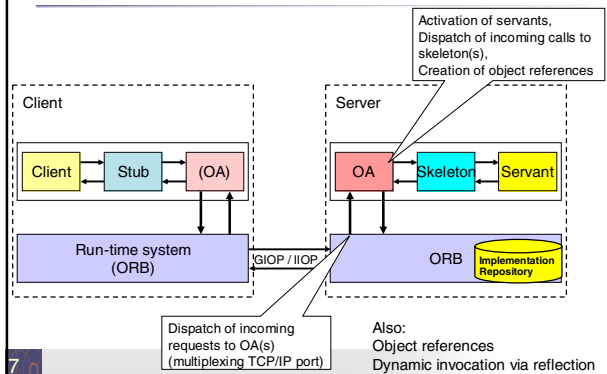
5

©2008 Component-based Systems, ICA, Linköping University. Some slides by courtesy of Olaf Adamiak, ICA / TU Dresden. Revised by C. Kessler, 2008-2010.

Corbas Mechanisms for Composition

(Basic Interoperability)

Recall: Solutions for Language and Location Transparency



7

Language Transparency

- Interface definition language – CORBA IDL
 - CORBA Interface Definition Language describes interfaces
 - From that, glue code is generated (*glue code* is code that glues non-fitting components together)
 - Generate stub and skeletons for language adaptation
 - Powerful type system
 - Standardized (ISO 14750)
- Language bindings for many languages
 - Antique: COBOL
 - Classic: C
 - OO: C++, SmallTalk, Eiffel, Java
 - Scripting: Python

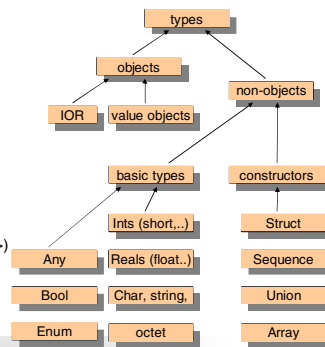
8

Concepts in the CORBA Interface Definition Language (IDL)

```

module <identifier> {
  <type declarations>
  <constant declarations>
  <exception declarations>

  // classes
  interface <identifier> : <inheriting-from> {
    <type declarations>
    <constant declarations>
    <exception declarations>
    // methods
    <optype> <identifier>(<parameters>);
    ...
  }
}
    
```



9

IDL-to-Language Mapping

- Bijjective mapping from Corba IDL types to programming language types**
 - Maps basic types directly
 - Maps type constructors
- Mapping makes transparent**
 - Byte order (big-endian / little-endian)
 - Word length
 - Memory layout
 - References
- One standard for each programming language!**

10

IDL-to-C, Mapping for basic types

Table 1-1 Data Type Mappings

OMG IDL	C
short	CORBA_short
long	CORBA_long
long long	CORBA_long_long
unsigned short	CORBA_unsigned_short
unsigned long	CORBA_unsigned_long
unsigned long long	CORBA_unsigned_long_long
float	CORBA_float
double	CORBA_double
long double	CORBA_long_double
char	CORBA_char
wchar	CORBA_wchar
boolean	CORBA_boolean
any	typedef struct CORBA_any { CORBA_TypeCode _type; void *_value; } CORBA_any;

Source: OMG, www.omg.org

11

IDL-to-Java, mapping of basic types

Table 2-1 Basic Type Mappings

IDL Type	Java type	Exceptions
boolean	boolean	
char	char	CORBA::DATA_CONVERSION
wchar	char	CORBA::DATA_CONVERSION
octet	byte	
string	java.lang.String	CORBA::MARSHAL CORBA::DATA_CONVERSION
wstring	java.lang.String	CORBA::MARSHAL CORBA::DATA_CONVERSION
short	short	
unsigned short	short	
long	int	
unsigned long	int	
long long	long	
unsigned long long	long	
float	float	
double	double	
fixed	java.math.BigDecimal	CORBA::DATA_CONVERSION

Source: OMG
www.omg.org

12

hello.idl

```
#ifndef _HELLOWORLD_IDL
#define _HELLOWORLD_IDL

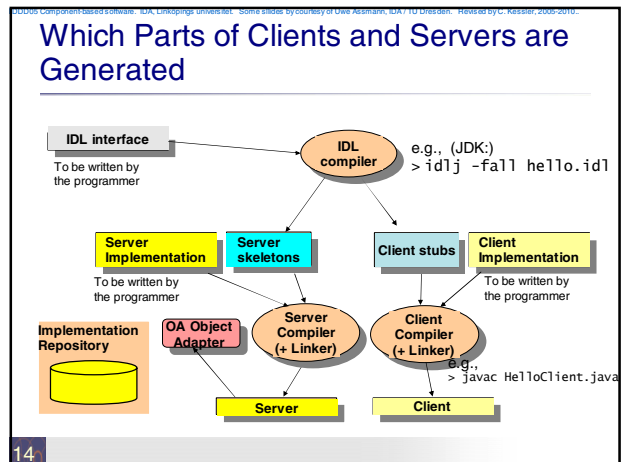
module HelloWorld {
    interface SimpleHelloWorld {
        string sayHello();
    };
};

#endif
```

count.idl

```
module Counter {
    // unbounded sequence of longs:
    typedef sequence<long> oneDimArray;
    // specify interface for a counter:
    interface Count {
        attribute long sum; // counter
        long increment();
        void readCtr ( in oneDimArray X,
                      in long position k );
    }
}
```

13



Example: Counter.idl

```
// IDL

module Counter {
    interface Counter {
        attribute long thecounter;
        void inc( in long k );
        long getcounter ( );
    };
};
```

15

Example (cont.): IDL compiler result

- Example: (for CORBA supplied in JDK 1.2 and later)

idlj -fall Counter.idl

generates the following files:

- Counter.java -- the Java interface for Counter
- CounterOperations.java -- the Java interface for Counter methods
- CounterPOA.java -- servant impl. class should inherit from this one
- CounterPOATie.java -- or delegate to this one (see later)
- CounterHolder.java -- serialization/deser. code for passing Counters
- CounterHelper.java -- type conversion routines for Counters
- _CounterStub.java -- class with the client-side stub code

16

Example (cont.): CounterOperations.java

```
package Counter;

/**
 * Counter/CounterOperations.java .
 * Generated by the IDL-to-Java compiler (portable), version "3.2"
 * from Counter.idl
 * den 23 april 2007 kl 10:02 CEST
 */

public interface CounterOperations
{
    int thecounter (); // getter method for thecounter, created automatically
    void thecounter (int newThecounter); // setter method for thecounter...
    void inc (int k);
    int getcounter ();
}
```

17

Example (cont.): Counter.java

```
package Counter;

/**
 * Counter/Counter.java .
 * Generated by the IDL-to-Java compiler (portable), version "3.2"
 * from Counter.idl
 * den 23 april 2007 kl 10:02 CEST
 */

public interface Counter
    extends CounterOperations,
           org.omg.CORBA.Object,
           org.omg.CORBA.portable.IDLEntity
{
}
```

18

Example (cont.): CounterPOA.java

```
package Counter;
/**
 * Counter/CounterPOA.java
 * Generated by the IDL-to-Java compiler (portable), version "3.2" from Counter.idl
 */
public abstract class CounterPOA extends org.omg.PortableServer
implements Counter.CounterOperations, org.omg.CORBA.portable
{
    // Registry for Counter-methods:
    private static java.util.Hashtable _methods = new java.util.Hashtable ();
    static {
        _methods.put ("_get_thecounter", new java.lang.Integer (0));
        _methods.put ("_set_thecounter", new java.lang.Integer (1));
        _methods.put ("_inc", new java.lang.Integer (2));
        _methods.put ("getcounter", new java.lang.Integer (3));
    }
    public org.omg.CORBA.portable.OutputStream _invoke (String $method,
        org.omg.CORBA.portable.InputStream in,
        org.omg.CORBA.portable.ResponseHandler $rh)
    {
        org.omg.CORBA.portable.OutputStream out = null;
        java.lang.Integer _method = (java.lang.Integer) _methods.get ($method);
        // ...
        switch ( _method.intValue () ) { ... } // call skeleton by method index - see next page
    }
}
```

19

Example (cont.): CounterPOA.java (cont.)

```
switch ( _method.intValue () ) {
    case 0: // Counter/Counter/_get_thecounter
    {
        int $result = (int)0;
        $result = this.thecounter ();
        out = $rh.createReply();
        out.write_long ($result);
        break;
    }
    case 1: // Counter/Counter/_set_thecounter
    {
        ...
    }
    case 2: // Counter/Counter/_inc
    {
        int k = in.read_long ();
        this.inc (k);
        out = $rh.createReply();
        break;
    }
    ...
    default: throw new org.omg.CORBA.BAD_OPERATION (0,
        org.omg.CORBA.CompletionStatus.COMPLETED_MAYBE);
}
return out; // result of _invoke
}
```

20

Example (cont.): _CounterStub.java

```
package Counter;
/**
 * Counter/_CounterStub.java
 * Generated by the IDL-to-Java compiler (portable), version "3.2" from Counter.idl
 */
public class _CounterStub extends org.omg.CORBA.portable.ObjectImpl
implements Counter.Counter
{
    // some other methods omitted ...
    public void inc (int k)
    {
        org.omg.CORBA.portable.InputStream $in = null;
        try {
            org.omg.CORBA.portable.OutputStream $out = _request ("inc", true);
            $out.write_long (k);
            $in = _invoke ($out);
            return;
        } catch (org.omg.CORBA.portable.ApplicationException $ex) {
            $in = $ex.getInputStream ();
            String _id = $ex.getId ();
            throw new org.omg.CORBA.MARSHAL (_id);
        } catch (org.omg.CORBA.portable.RemarshalException $rm) {
            inc (k);
        } finally {
            _releaseReply ($in);
        }
    } // inc
}
```

21

The Top Class: CORBA::Object

CORBA::Object

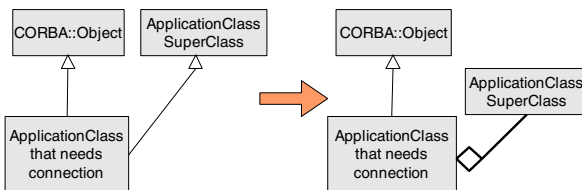
get_implementation
get_interface
is_nil
is_a
create_request
duplicate
release
....

- The class **CORBA::Object** is inherited to all objects
 - supports reflection and introspection
- Reflective functions:
 - get_interface
delivers a reference to the entry in the interface repository
 - get_implementation
a reference to the implementation
- Reflection also by the Interface Repository (list initial references from the CORBA::ORB interface).

22

Problem: Multiple Inheritance

- CORBA::Object includes *code* into a class
- Many languages only offer single inheritance
 - Application superclass must be a delegatee



23

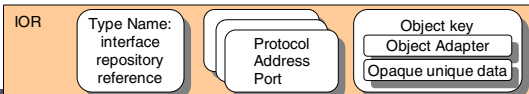
Interoperable Object Reference (IOR)

- An **object reference** provides information to uniquely specify an object within a distributed ORB system
- Unique name or identifier
- Language-transparent:
Mapped to client's normal source language references (unique mapping for each supported language)
- Implementation in CORBA:
Object reference to a server object is given out by the server's OA, shipped to clients as **IOR object** and stored there in a proxy object.
ORB supports **stringification / dstringification** of IOR's.
Retrieval of references by client: supported by naming service
- All referencing goes via the server's ORB
-> enables distributed reference counting

24

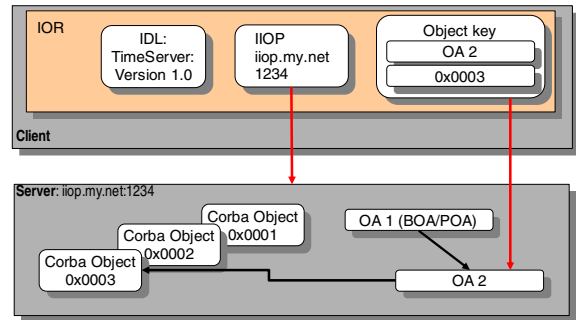
Interoperable Object Reference (IOR) - cont.

- Transient (terminates with server) or persistent
- IOR is larger, more time-consuming than language-bound reference
- Consists of:
 - Type name** (code), i.e. index into Interface Repository
 - Protocol and address information** (e.g. TCP/IP, port #, host name), could support more than one protocol
 - Object key:**
 - Object adapter name (for OA)
 - Opaque data only readable by the generating ORB (local reference)



25

IOR Example



26

How to get an IOR?

- Object references originate in servers.**
 - If client needs a reference, a server must create it.
 - Chicken-and-egg problem...
- Solutions:**
 - Server write stringified IOR to a file (e.g., stdout)**
 - Ok for tests, but not for realistic distributed systems
 - Use the CORBA naming service**
 - Naming service stores (name, IOR) bindings in central location
 - Only location of naming service needs to be known to client
 - Use the CORBA trading service**
 - Look up IOR for objects by reg. properties, instead of by name

27

Example: Time Service

- Call provides current time (on server)
- Code to write:
 - Interface in IDL →
 - Server
 - Starts ORB
 - Initializes Service
 - Gives IOR to the output
 - Client
 - Takes IOR
 - Calls service

```
//TestTimeServer.idl
module TestTimeServer{
    interface ObjTimeServer{
        string getTime();
    };
};
```

28

Time Service Component as part of the server implementation (Java)

```
//TestTimeServerImpl.java
import CORBA.*;

class ObjTestTimeServerImpl
    extends TestTimeServer.ObjTimeServer_Skeleton
    //which is generated from IDL
{
    //Variables
    //Constructor
    //Method (Service) Implementation
    public String getTime() throws CORBA.SystemException
    {
        return "Time: " + currentTime;
    }
};
```

29

Time Service The other part of the server implementation

```
// TimeServer_Server.java
import CORBA.*;
public class TimeServer_Server {
    public static void main( String[] argv ) {
        try {
            CORBA.ORB orb = CORBA.ORB.init();
            ...
            ObjTestTimeServerImpl obj
                = new ObjTestTimeServerImpl(...);
            ...
            // print stringified object reference:
            System.out.println( orb.object_to_string(obj));
        }
        catch (CORBA.SystemException e){
            System.err.println(e);
        }
    }
};
```

30

Time Service Client Implementation

```
//TimeServer_Client.java
import CORBA.*;

public class TimeServer_Client{
    public static void main( String[] argv ) {
        // pass stringified object reference as argv[0]
        try {
            CORBA.ORB orb = CORBA.ORB.init();
            ...
            CORBA.Object obj = orb.string_to_object( argv[0] ); //IOR
            ...
            TestTimeServer.ObjTimeServer timeServer = // downcast
                TestTimeServerImpl.ObjTimeServer_var.narrow(obj);
            ...
            System.out.println( timeServer.getTime() ); // invoke
        } catch (CORBA.SystemException e) { System.err.println(e); }
    }
}
```

31

Time Service Execution

```
C:\> java TimeServer_Server
```

```
IOR:000000000000122342435 ...
```

```
C:\> java TimeServer_Client
```

```
IOR:000000000000122342435 ...
```

```
Time: 14:35:44
```

32

GIOP / IIOP

OSI Networking Model layers CORBA GIOP / IIOP layers

- | | |
|------------------|-------------|
| ■ 7 Application | ■ ORB |
| ■ 6 Presentation | ■ GIOP |
| ■ 5 Session | ■ IIOP |
| ■ 4 Transport | ■ TCP |
| ■ 3 Network | ■ IP |
| ■ 2 Data Link | ■ Data Link |
| ■ 1 Physical | ■ Physical |

33

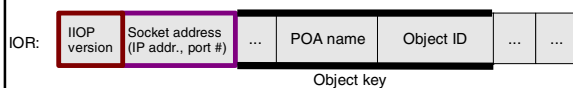
GIOP General Inter-ORB Protocol

- **General protocol, simple, abstract**
 - Independent of any particular transport protocol (IIOP: over TCP/IP)
- **Asymmetric (client-server) connections**
 - Client creates connection
 - Server receives requests and replies (without knowing client)
- **Connection-oriented transport, no packet size restrictions**
- **Common data representation (CDR): octet (8-bit bytes) stream**
 - Sender endianness information in header
 - Sender alignment information (1, 2, or 4 bytes) in header
 - Sender sends natively aligned data, receiver adapts if necessary
 - Encoding of the IDL datatypes
- **Message formats:**
 - Request, LocateRequest, CancelRequest (client)
 - Reply, LocateReply (server)
- MessageError, Fragment, CloseConnection (both)

34

IIOP (Internet Inter-ORB Protocol)

- Implementation of GIOP on top of TCP/IP
- TCP/IP Socket communication
- Adds socket address information to IOR contents



35

Basic CORBA Connections

Basic Connections in CORBA

- **Static method call with static stubs and skeletons**
 - Local or remote
- **Polymorphic call**
 - Local or remote
- **Event transmission**
- **Callback**
- **Dynamic invocation (DII, request broking)**
 - Searching services dynamically in the web (location transparency of a service)
- **Trading**
 - Find services in a yellow pages service, based on properties

37

Static CORBA Call

- **Advantage: the participants (methods) are statically known**
 - Call by stub and skeletons, without involvement of an ORB
 - Supports distribution:
 - Exchange of local call in one address space to remote call is very easy:
 - Inherit from a CORBA class
 - Write an IDL spec
 - No search for service objects → rather fast
 - Better type check, since the compiler knows the involved types
- **The call goes through the server object adapter**
 - This hides the detail whether the server is transient or persistent

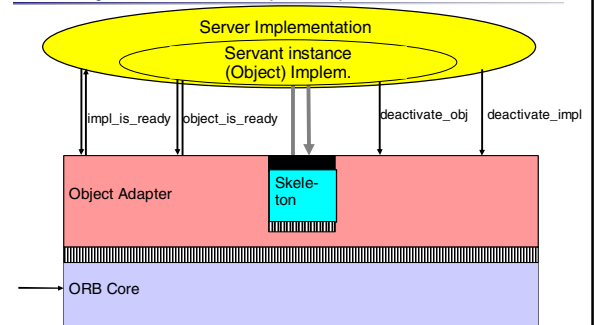
38

Client side protocol for static calls

- **Step 1: Initialize the ORB**
 - `global_orb = CORBA::ORB_init ( argc, argv );`
- **Step 2: Obtain an object reference (here: from file)**
 - `CORBA::Object obj = global_orb -> string_to_object( read_refstring("filename.ref") );`
 - **and narrow it to expected object type (dynamic downcast)**
 - `Counter::Counter ctr = Counter::Counter::_narrow( obj );`
- **Step 3: Invoke on Count object**
 - `ctr->increment();`
 - ...
- **Step 4: Shut down the ORB**
 - `global_orb->shutdown(1); global_orb->destroy();`

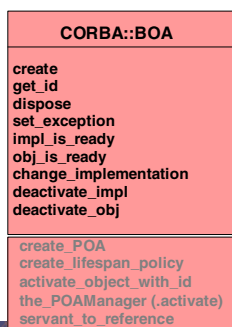
39

Server Side, Old-style Protocol (BOA)



40

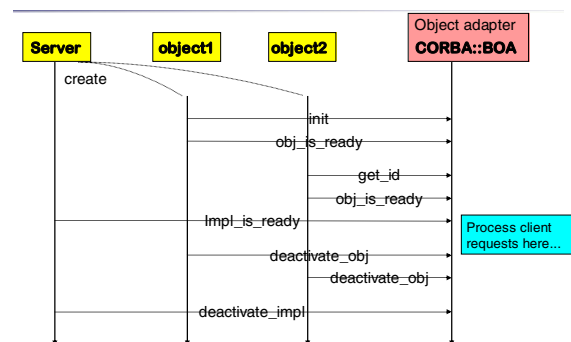
Basic Object Adapter BOA



- The BOA hides
 - Life time of the server object (activation: start, stop)
 - Persistency
- The BOA is implemented in every ORB, for minimal service provision
- The BOA maintains the implementation repository (component registry).
- It supports non-object-oriented code
- In CORBA 3.0 replaced by POA (Portable Object Adapter)

41

Object Activation on the Server (BOA version)



42

Object Adapters Support Different Server Models

- Common server process
 - Several objects reside in one process on the server; the OA initializes them as threads with common address space ("common apartment")
 - e.g. BOA: deactivate_impl, impl_is_ready, obj_is_ready are mapped directly to thread functions
- Separate server processes
 - For every object an own process
- Server-per-request
 - Every request generates a new process
- Persistent server
 - Here starts another application the objects (e.g., a data base).
 - The BOA passes on the queries

43

POA Portable Object Adapter

- The POA is an evolution of the BOA
- Nested POAs possible, with nested name spaces
 - Root POA (one per server) started/accessed by ORB.
 - A POA can create new POAs.
 - A POA may serve a group of objects and handle references to them.
- POAs can be named
 - ORB maintains a registry of named POAs, e.g. for reactivation as needed.
- Policies for object management
 - e.g. Lifespan: transient / persistent

CORBA::BOA

```
create
get_id
dispose
set_exception
impl_is_ready
obj_is_ready
change_implementation
deactivate_impl
deactivate_obj
```

CORBA::POA

```
create_POA
create_lifespan_policy
activate_object_with_id
the_POAManager (.activate)
servant_to_reference
...
```

44

Towards Dynamic Call (DII, Request Broking)

- Dynamic call via the ORB's DII (Dynamic Invocation Interface)
 - Services can be dynamically exchanged, or brought into the play a posteriori
 - Without recompilation of clients
 - Slower than static invocations
- Requires introspection
- Requires descriptions of semantics of service components...
 - For identification of services
 - Metadata (descriptive data): catalogs of components (interface repository, implem. repository)
 - Naming service, Trading service, Property service (later)
- ... and a mediator that looks up for services: the ORB

45

Object Request Broker ORB

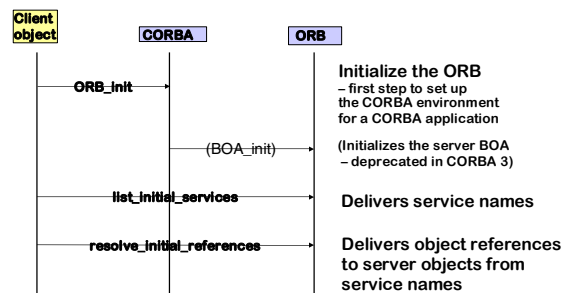
CORBA::ORB

```
init
object_to_string
string_to_object
create_list
create_operation_list
get_default_context
create_environment
list_initial_services
resolve_initial_references
....
```

- ORB is a Mediator
 - Hides the the environment from clients
- List_initial_services:
 - yields list of names of initial services e.g. Naming Service
- Resolve_initial_references:
 - uses the naming service e.g. to get an IOR to "NameService" or the "RootPOA"
- ORB is responsible for managing all communication:
 - Can talk to other ORBs on the network (IIOP Internet Inter-ORB protocol)

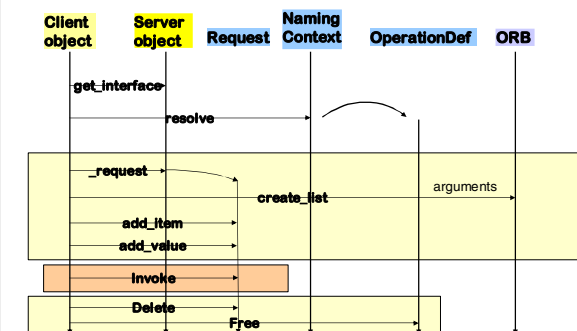
46

ORB Activation



47

Protocol Dynamic Call (DII)



48

Example for Dynamic Call

```
// Ship.idl

module Ship {

    interface Aircraft {
        string codeNumber();
    };

    interface AircraftCarrier {
        Aircraft launch (in string name);
    };
};
```

Source: Infowave, Building distributed applications...,
www.waveman.com/etac/corba/page13.html, 1998

49

Example 1: Dynamic Call in C++ Client program

```
CORBA::ORB_ptr orb;

main( int argc, char* argv[]) {
    orb = CORBA::ORB_init(argc,argv, ORBID);
    // alternative description of service
    CosNaming::NamingContext_ptr naming =
        CosNaming::NamingContext::_narrow(
            ::resolve_initial_reference("NameService"));
    CORBA::Object_ptr obj;
    try {
        obj = naming->resolve( mk_name("dl_smp1"));
    } catch (CORBA::Exception) {
        cerr << "not registered" << endl; exit(1);
    }

    // Build request (short form)
    CORBA::Request_ptr rq = obj->_request("op");
    // Create argument list
    rq->arguments() = orb->create_list();
    rq->arguments()->add_value("arg1",val1,CORBA::ARG_IN);
    rq->arguments()->add_value("arg2",val2,CORBA::ARG_OUT);
    rq->arguments()->add_value("arg3",val3,CORBA::ARG_INOUT);

    // Invoke request
    rq->invoke();

    // Analyze result
    CORBA::Short rslt;
    if ("(rq->result()->value()) >= rslt" ) {
        // Analyze the out/inout-parameters (arg1 has index 0)
        CORBA::Short _arg2, _arg3;
        *(rq->arguments()->item(1)->value()) >= _arg2;
        *(rq->arguments()->item(2)->value()) >= _arg3;
        cout << "arg2=" << _arg2 << " arg3=" << _arg3
            << endl;
    } else {
        cout << "result has unexpected type" << endl;
    }
}
```

50

Example 2: DII Invocation in Java Client program (1)

```
// Client.java
// Adapted from: Building Distributed Object Applications with CORBA
// Infowave (Thailand) Co., Ltd.
// Jan 1998

public class Client {
    public static void main(String[] args) {
        if (args.length != 2) {
            System.out.println("Usage: vbj Client <carrier-name> <aircraft-name>\n");
            return;
        }
        String carrierName = args[0];
        String aircraftName = args[1];
        org.omg.CORBA.Object carrier = null;
        org.omg.CORBA.Object aircraft = null;
        org.omg.CORBA.ORB orb = null;

        try {
            orb = org.omg.CORBA.ORB.init( args, null);
        } catch (org.omg.CORBA.SystemException se) {
            System.err.println("ORB init failure " + se);
            System.exit(1);
        }
    }
}
```

51

Example 2: DII Invocation in Java Client code (2)

```
{ // scope of request object:
    try { ... (simplified)
        carrier = intf_rep.lookup("IDL:Ship/AircraftCarrier:1.0");
    }
    catch (...) {
        System.err.println("..." + se);
        System.exit(1);
    }

    org.omg.CORBA.Request request = carrier._request("launch");
    request.add_in_arg().insert_string( aircraftName );
    request.set_return_type( orb.get_primitive_tc( org.omg.CORBA.TCKind.tk_objref ) );
    // Step 4: Invoke request:
    request.invoke();
    // Step 5: Read result value:
    aircraft = request.result().value().extract_Object();
}

{ // scope of another DII call ( use a fresh request object):
    org.omg.CORBA.Request request = aircraft._request("codeNumber" );
    request.set_return_type( orb.get_primitive_tc ( org.omg.CORBA.TCKind.tk_string ) );
    request.invoke();
    String designation = request.result().value().extract_string();
    System.out.println ("Aircraft " + designation + " is coming your way");
}
```

52

Example 2 Server code (Java, POA version)

```
// Building Distributed Object Applications with CORBA
// Infowave (Thailand) Co., Ltd. http://www.waveman.com, Sep 2000

import java.io.*;
import org.omg.CosNaming.*;

public class Server {
    public static void main( String[] args ) {
        // Step 1: Initialize server ORB
        org.omg.CORBA.ORB orb =
            org.omg.CORBA.ORB.init( args, null );

        org.omg.CORBA.Object objPOA = null;
        try { Step 2: Get RootPOA ref from naming
            objPOA = orb.resolve_initial_references("RootPOA");
        } catch (org.omg.CORBA.ORBPackage.InvalidName
            ex) { ... }
        org.omg.PortableServer.POA rootPOA = null;
        rootPOA = (org.omg.PortableServer.POA) objPOA;
        // Step 3: Narrow it to a RootPOA object
        (downcast)

        // Step 4: Create new POA
        with specific policies:
        org.omg.PortableServer.POA myPOA = null;
        try {
            myPOA = rootPOA.create_POA(
                "personalPOA",
                rootPOA.the_POAManager(),
                new org.omg.CORBA.Policy[] {
                    rootPOA.create_id_assignment_policy (
                        org.omg.PortableServer.
                            IdAssignmentPolicyValue.USER_ID ) });
        } catch (java.lang.Exception ex) {
            System.err.println("Create POA Exception " + ex);
            System.exit(1);
        }

        // Step 5: Create new servant object:
        org.omg.PortableServer.Servant carrier = null;
        try { ...pass the POA to its constructor
            carrier = new AircraftCarrierImpl(myPOA);
            myPOA.activate_object_with_id
                ("Nimitz".getBytes(), carrier); and activate it
        } catch (org.omg.CORBA.SystemException se) { ... }
        catch (org.omg.CORBA.UserException ue) { ... }
    }
}
```

53

Example 2 Server code (Java, POA version) - continued

```
....

// Write object reference to an IOR file
org.omg.CORBA.Object initRef = null;
try {
    initRef = myPOA.servant_to_reference( carrier );

    FileWriter output = new FileWriter("ns.ior");
    output.write( orb.object_to_string( initRef ) );
    output.close();
    System.out.println("Wrote IOR to file: ns.ior");

    // Step 6: Activate the POA manager:
    myPOA.the_POAManager().activate();
    System.out.println( carrier + " ready for launch !!!");
    orb.run(); // Step 7: Hand over application control to the ORB
    to service incoming calls
} catch (java.lang.Exception exb) {
    System.err.println("Exception Last deep in here " + exb);
    System.exit(1);
}
```

54

Example 2 Servant implementation (Java, POA version)

// Adapted from: Building Distributed Object Applications with CORBA
// Infowave (Thailand) Co., Ltd. http://www.waveman.com, Sep 2000

```
public class AircraftCarrierImpl extends Ship.AircraftCarrierPOA
{
    private org.omg.PortableServer.POA _myPOA;

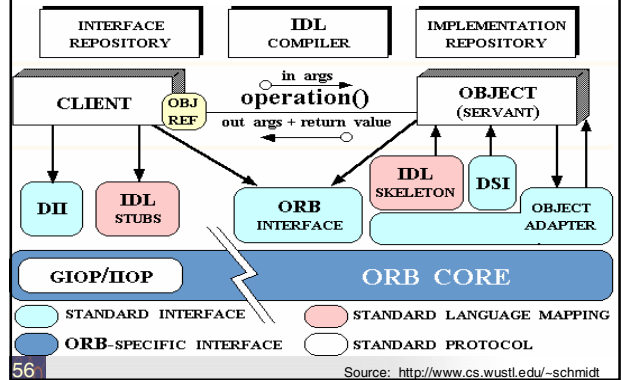
    // Constructor:
    public AircraftCarrierImpl (
        org.omg.PortableServer.POA myPOA ) {
        _myPOA = myPOA;
        // Record a ref. to my POA
        // (here, in constructor)
    }

    public Ship.Aircraft launch ( String name ) {
        org.omg.PortableServer.Servant aircraft
        = new AircraftImpl( name );
        // Can register created objects ...
        _myPOA.activate_object_with_idl(
            "name".getBytes(), aircraft );
        // ... as CORBA objects with my POA
        catch ( java.lang.Exception ex )
        {
            System.err.println("Exception 2 " + ex);
            System.exit(1);
        }
    }

    ....
    System.out.println( name + " on Catapult 2");
    Ship.Aircraft _aircraft = null;
    try {
        _aircraft = Ship.AircraftHelper.narrow(
            _myPOA.create_reference_with_idl(
                "name".getBytes(),
                aircraft_all_interfaces(null, null)[0]);
        )
    } catch ( java.lang.Exception ex )
    {
        System.err.println("Exception 3 " + ex);
        System.exit(1);
    }
    return _aircraft;
}
```

55

CORBA interoperability mechanisms Summary and further components



56

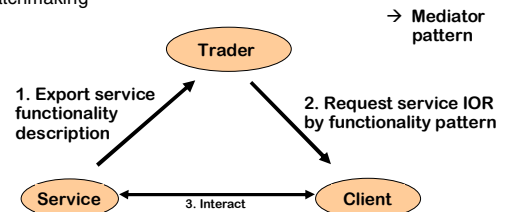
Available ORBs

- **Java-based**
 - IBM WebSphere
 - SUN NEO, Joe: own protocol. the Java Transaction Service JTS is the JOE Corba Object Transaction Service OTS.
 - IONA Orbix: developed in Java, i.e., ORBlets possible, C++, Java-applications
 - BEA WebLogic
 - Borland Visibroker (in Netscape Communicator), IOP based. Also for C++.
 - **free:** JacORB, ILU, Jorba, DynaORB, OpenORB, JDK1.4+
- **C-based**
 - ACE ORB TAO, University Washington (with trader)
 - Linux ORBIT (gnome) (also for Cygwin).
 - Linux MICO (kde 1.0 used it)
- **Python-based**
 - fnorb
- <http://www.omg.org>
- [Szyperski CS 13.4]

57

Beyond Dynamic Call: The Trader Service

- Trader mediates services, based on published properties ("yellow page service")
- Matchmaking



58

ORBs and Traders

- The ORB resolves operations still based on naming (with the *Naming service* = "White pages")
- The **Trader service**, however, resolves operations (services) without names, only based on *properties* and *policies* = "Yellow pages"
- The trader gets *offers* from servers, containing new services

59

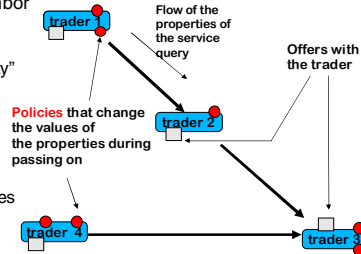
Service offers for the Trader service

- **Service offer (IOR, properties)**
 - Properties describe services
 - Are used by traders to match services to queries
- **Dynamic property**
 - A property can be queried dynamically by the trader of service
 - The service-object can determine the value of a dynamic property anew
- **Matching with the standard constraint language**
 - Boolean expressions about properties
 - Numeric and string comparisons

60

Traders Provide Service Hopping

- If a trader does not find a service, it can ask neighbor traders
 - Design pattern "Chain of Responsibility"
- Graph of traders
 - Links to neighbors via TraderLink
 - TraderLink filters and manipulates queries via policies
- A distributed search algorithm (also used in P2P)

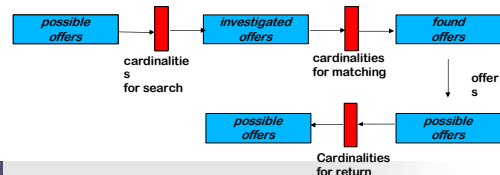


61

Modification of Queries

Policies parameterize the behavior of the traders and the TraderLinks

- Filters, i.e., values, limiting / modifying the queries:
 - max_search_card: maximal cardinality for the ongoing searches
 - max_match_card: maximal cardinality for matchings
 - max_hop_count: maximal search depth in the graph



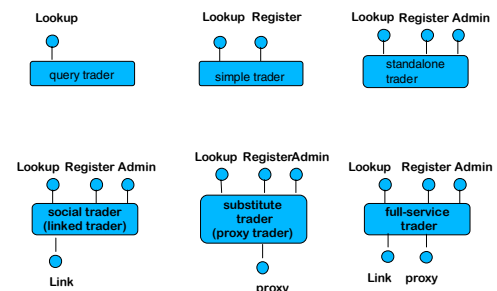
62

Interfaces Trading Service

- Basic interfaces**
 - Lookup (query)
 - Register (for export, retract, import of services)
 - Admin (info about services)
 - Link (construction of trader graph)
- How does a query look like?**
 - Lookup.Query(in ServicetypeName, in Constraint, in PolicySeq, in SpecifiedProperties, in how_to_y, out OfferSequence, offerIterator)

63

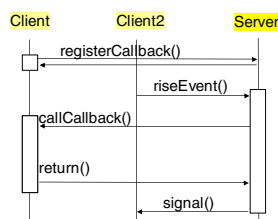
CORBA Trader Types



64

Callbacks with the Callback Service

- Callback function registration**
 - Procedure variable, closure (procedure variable with arguments) or reference to an object
- Callback works for all languages**
- Callback reverses roles of client and server**



65

Events

- Send event objects from event suppliers to event consumers**
unidirectional event channels decouple supplier and consumer
- Event objects (also called messages) are immutable once sent**
 - Asynchronous communication; order of events is not respected
 - No return values (except with references to collector objects)
- Unicast:** one receiver
- Multicast:** many receivers
- Dynamically varying receivers**
(register at channels as supplier / consumer; event type filtering)
- Works for every CORBA language**

66

CORBA Event Service

- **Push model:**
Supplier sends event object by calling *push* operation on channel, which calls *push* to deliver event object to all registered consumers
- **Pull model:**
Consumer calls *pull* operation on channel (polling for arriving events) which triggers calls to *pull* to registered suppliers
- **As intermediate instances, an event channel can be allocated**
 - They buffer, filter, and map pull to push
- **Untyped generic events, or typed by IDL**
- **Advantage:**
 - Asynchronous working in the Web (with IIOP and dynamic Call)
 - Attachment of legacy systems interesting for user interfaces, network computing etc.

67 **Disadvantage:** Very general interface

Corba 3.0 since 1999

- Provides the well-defined packaging for producing components
- Messaging
- Language mappings that avoid hand-writing of IDL
 - Generating IDL from language specific type definitions
 - C++2IDL, Java2IDL, ...
- XML integration (SOAP)
- Quality of Service management
- Real-time and small footprint versions
- CORBA Component Model (CCM)
 - similar to EJB, see later
- Scripting (CORBA script), a composition language

68

Corba 3.0 (cont.)

- **New Basic services:**
 - POA, the Portable Object Adapter, replaces BOA
 - SFA, Server Framework Adapter
 - Value objects
- **Services:**
 - Message Service MOM:
Objects as asynchronous buffered messages
 - Corba Beans-components
 - Script language
- **Facilities:**
compound documents, Mobile Agents, BOF (business object facility)

69

Evaluation of CORBA

as composition system

Evaluation: Component Model

- **Mechanisms for secrets and transparency: very good**
 - Interface and Implementation repository
 - Component language hidden (interoperability)
 - Life-time of service hidden
 - Identity of services hidden
 - Location hidden
- **No parameterization**
- **Many standards (see following subchapters)**

71

Evaluation: Standardization

- **Quite good!**
 - Services, application services
 - On the other hand, some standards are FAT
- **Technical vs. application specific vs business components:**
 - Corba has standards for technical and application specific components
 - ... but for business objects, standards must be extended (vertical facilities)

72

Evaluation: Composition Technique

- **Mechanisms for connection**
 - Mechanisms for adaptation: stubs, skeletons, server adapters
 - Mechanisms for glueing: marshalling based on IDL
- **Mechanisms for aspect separation**
 - Multiple interfaces per object
- **Nothing for extensions**
- **Mechanisms for Meta-modeling**
 - Interface Repositories with type codes, implementation repositories
- **Scalability**
 - Connections cannot easily be exchanged (except static local and remote call)

73

Evaluation: Composition Language

- **Weak**
 - CORBA scripting provides a facility to write glue code, but only black-box composition

74

What Have We Learned (1)

- **CORBA is big, but universal:**
 - The Corba-interfaces are very flexible, work, and can be used in practice
 - ... but also complex and fat, maybe too flexible
 - If you have to connect to legacy systems, CORBA works
- CORBA has the advantage of an **open standard**
- **Trading** and **dynamic call** are advanced communication mechanisms
- CORBA was probably only the first step, web services might be taking over

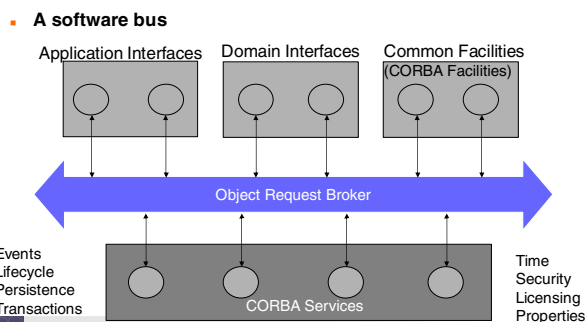
75

APPENDIX

- **Advanced material on CORBA (for self-study)**
 - CORBA services
 - CORBA facilities
 - CORBA and the web, ORBlets

76

OMA (Object Management Architecture)



77

Corba Services

OMG: *CORBA services: Common Object Service Specifications.*
<http://www.omg.org>.

OMG: *CORBA facilities: Common Object Facilities Specifications.*

Overview on Corba Services

- **16+ standardized service interfaces** (*i.e.*, a library)
 - Standardized, but status of implementation different depending on producer
- **Object services**
 - Deal with features and management of objects
- **Collaboration services**
 - Deal with collaboration, *i.e.*, object contexts
- **Business services**
 - Deal with business applications
- The services serve for standardization.
They are very important to increase reuse.
 - Available for every language, and on distributed systems!

79

Object Services

- **Name service** (directory service)
 - Records server objects in a simple tree-like name space
 - (Is a simple component system itself)
- **Lifecycle service** (allocation service)
 - Not automatic; semantics of deallocation undefined
- **Property service** (feature service for objects)
- **Persistency service** (storing objects in data bases)
- **Relationship service** to build interoperable relations and graphs
 - Support of standard relations: reference, containment
 - Divided in standard roles: contains, containedIn, references, referenced
- **Container service** (collection service)

80

Collaboration Services

- **Communication services**
 - Resemble connectors in architecture systems, but cannot be exchanged to each other
 - Event service
 - push model: the components push events into the event channel
 - pull model: the components wait at the channel and empty it
 - Callback service
- **Concurrency service**
 - Distributed locks
- **Object transaction service, OTS**
 - Flat transactions on object graphs

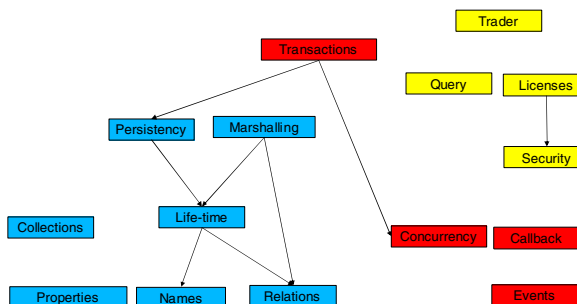
81

Business Services

- **Trader service**
 - Yellow Pages, localization of services
- **Query service**
 - Search for objects with attributes and the OQL, SQL (ODMG-93)
- **Licensing service**
 - For application providers (application servers)
 - License managers
- **Security service**
 - Use of SSL and other basic services

82

Dependencies Between the Services



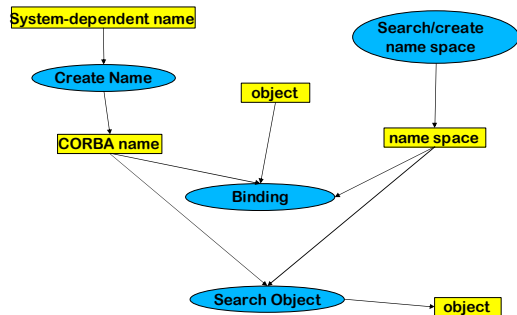
83

Object Services: Names

- **Binding of a name creates an object in a name space (directory, scope, naming context).**
 - A *name space* is a container with a set of bindings of names to values.
 - They can reference each other and build name graphs
- **The representation of a name is based on abstract syntax, not on the concrete syntax of a operating system or URL.**
 - A name consists of a tuple (Identifier, Kind).
 - The Identifier is the real name, the Kind tells how the name is represented (e.g., c_source, object_code, executable, postscript,...).
 - For creation of names there is a library (design pattern Abstract Factory).

84

Use of Names



85

Naming Service

CosNaming::NamingContext

```

bind ( in Name n, in Object obj)
rebind ( in Name n, in Object obj)
bind_context
rebind_context
mk_name(String s)
Object resolve
unbind ( in Name n)
NamingContext new_context;
NamingContext bind_new_context ( in Name n)
void destroy
void list (...)
_narrow()
  
```

86

Naming Service

```

void bind( in Name n, in Object obj)
  raises( NotFound, CannotProceed, InvalidName, AlreadyBound );
void rebind( in Name n, in Object obj)
  raises( NotFound, CannotProceed, InvalidName );
void bind_context( in Name n, in NamingContext nc)
  raises( NotFound, CannotProceed, InvalidName, AlreadyBound );
void rebind_context( in Name n, in NamingContext nc )
  raises( NotFound, CannotProceed, InvalidName );
Name mk_name( String s );
Object resolve( in Name n )
  raises( NotFound, CannotProceed, InvalidName );
void unbind( in Name n )
  raises( NotFound, CannotProceed, InvalidName );
NamingContext new_context();
NamingContext bind_new_context( in Name n )
  raises( NotFound, AlreadyBound, CannotProceed, InvalidName );
void destroy()
  raises( NotEmpty );
void list( in unsigned long how_many,
          out BindingList bl, out BindingIterator bi );
  
```

87

Naming Service in IDL

```

module CosNaming {
  typedef string lstring;
  struct NameComponent {
    lstring id;
    lstring kind;
  };
  typedef sequence<NameComponent> Name;
  enum BindingType { noobject, nocontext };
  struct Binding {
    Name binding_name;
    BindingType binding_type;
  };
  typedef sequence<Binding> BindingList;
  interface BindingIterator;
  interface NamingContext;
}

...

interface NamingContext {
  enum NotFoundReason { missing_node,
                        not_context, not_object };
  exception NotFound {
    NotFoundReason why;
    Name rest_of_name;
  };
  exception CannotProceed {
    NamingContext cbt;
    Name rest_of_name;
  };
  exception InvalidName {};
  exception AlreadyBound {};
  exception NotEmpty {};

  void bind ( in Name n, in Object obj )
    raises ( NotFound, CannotProceed, ... )
    ... // other operations omitted, see previous page

  interface BindingIterator {
    boolean next_one( out Binding b);
    boolean next_n( in unsigned long how_many,
                   out BindingList bl);
    void destroy();
  };
  ....
}
  
```

88

Naming Service: Example

```

// From: Redlich
import java.io.*;
import java.awt.*;
import IE.Iona.Orbix2.CORBA.SystemException; // OrbixWeb
import CosNaming.NamingContext; // name service/context
import CosNaming.NamingContext.*; // name service/exceptions
import Calc5.calc.complex; // type 'complex' from Calc5

class MyNaming extends CosNaming {
  ...
}

public class client extends Frame {
  private Calc5.calc.Ref calc;
  private TextField inR, inI,
  private Button setB, addB, multB,
  private Button quitB, zeroB;

  public static void main( String argv[]) {
    CosNaming.NamingContext.Ref cbt;
    Calc5.calc_factory.Ref cf;
    Frame f;

    try {
      cbt = NamingContext._narrow( MyNaming.
        resolve_initial_references( MyNaming.NameService ));

      cf = Calc5.calc_factory._narrow(
        cbt.resolve( MyNaming.mk_name("calcfac")));

      f = new client( cf.create_new_calc() );
      f.pack();
      f.show();
    }
    catch (Exception ex) {
      System.out.println("Calc-5!init" + ex.toString());
    }
  }
}
  
```

89

Object Services: Persistency

- Definition of a Persistent Object Identifier (PID)
 - references the *value* of a CORBA object (in contrast to a CORBA object)
- Interface
 - connect, disconnect, store, restore, delete
- Attachment to data bases possible

90

Object Services: Property Service

- **Management of lists of features (properties) for objects**
 - Properties are strings
 - Dynamically extensible
- **Concept well-known as**
 - LISP property lists, associative arrays, Java property classes
- **Iterators for properties**
- **Interface:**
 - `define_property`, `define_properties`, `get_property_value`, `get_properties`, `delete_property`

91

Collaboration Services: Transactions

- **What a dream: the Web as data base with nested transactions.**
Scenarios:
 - Accounts as Web-objects.
Transfers as transaction on the objects of several banks
 - Parallel working on web sites: how to make consistent?
- **Standard 2-phase commit protocol:**
 - `begin_ta`, `rollback`, `commit`
- **Nested transactions**
 - `begin_subtransaction`, `rollback_subtransaction`, `commit_subtransaction`

92

CORBA Facilities (Standards for Application Domains)

Application-domain-specific interfaces

Horizontal Facilities (applicable in many domains)

- **User interfaces**
 - Printing, Scripting
 - Compound documents
e.g. OpenDoc (since 1996 accepted as standard format. Source code has been released of IBM. Now obsolete.)
- **Information management**
 - Metadata (meta object facility, MOF)
 - Tool interchange:
a text- and stream-based exchange format for UML (XMI)
 - Common Warehouse Model (CWM):
MOF-based metamodel for database applications

94

Vertical Facilities (Domain-Specific Facilities)

The Domain technology committee (DTC) creates domain task forces DTF for an application domain

- **Business objects**
- **Finance/insurance**
 - Currency facility
- **Electronic commerce**
- **Manufacturing**
 - Product data management enablers (PDM)
- **Medicine (healthcare CorbaMed)**
 - Lexicon Query Service
 - Person Identifier Service PIDS
- **Telecommunications**
 - Audio/visual stream control object
 - Notification service
- **Transportation**

95

CORBA, Web and Java

Corba and the Web

- **HTML solves many of the CORBA problems**
- **HTTP only for data transport**
 - HTTP cannot call methods, except by CGI-gateway-functionality (CGI = common gateway interface)
 - Behind the CGI-interface is a general program, communicating with HTTP via untyped environment variables (HACK!)
 - HTTP servers are simple ORBs, pages are objects
 - The URI/URL-name schema can be integrated into CORBA
- **IIOB becomes a standard internet protocol**
 - Standard ports, URL-mappings and standard-proxies for firewalls will be available
- **CORBA is an extension of HTTP of data to code**

97

CORBA and Java

- **Java is an ideal partner for CORBA :**
 - Bytecode is mobile
 - Applets: move calculations to clients (thin/thick client problem)
 - can be used for migration of objects, ORBs, and agents
 - Since 1999 direct CORBA support in JDK 1.2
 - IDL-to-Java mapping, IDL compiler, Java-to-IDL compiler, name service, ORB
 - Corba supports for Java a distributed interoperable infrastructure
- **Java imitates functionality of CORBA**
 - Basic services: Remote Method Invocation RMI, Java Native code Interface JNI
 - Services: serialization, events
 - Application-specific services (facilities): reflection, properties of JavaBeans

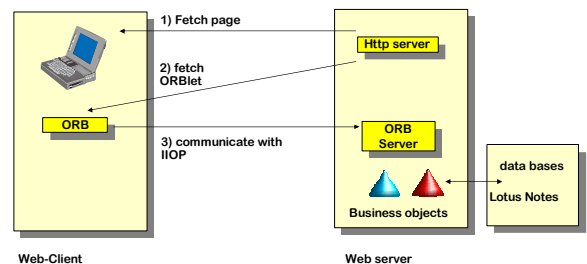
98

Corba and the Web (Orblets)

- ORBs can be written as bytecode applets if they are written in Java (*ORBllet*)
- Coupling of HTTP and IIOB:
 - Download of an ORBllet with HTTP
 - Talk to this ORB to get contact to server
- Replaces CGI hacks!
- Will be realized in web services (see later).

99

ORBllets



100