Figure 17.1 IDL interfaces *Shape* and *ShapeList*.

```
struct Rectangle{                                                          1
        long width;
        long height;
        long x;
        long y;
} ;
struct GraphicalObject {                                                   2
        string type;
        Rectangle enclosing;
        boolean isFilled;
};
interface Shape {                                                          3
        long getVersion() ;
        GraphicalObject getAllState() ;    // returns state of the GraphicalObject
};
typedef sequence <Shape, 100> All;                                         4
interface ShapeList {                                                      5
    exception FullException{ };                                            6
    Shape newShape(in GraphicalObject g) raises (FullException);           7
    All allShapes();              // returns sequence of remote object references  8
    long getVersion() ;
};
```

Figure 17.2 Java interface *ShapeList* generated by *idltojava* from CORBA interface *ShapeList*.

```
public interface ShapeList extends org.omg.CORBA.Object {
    Shape newShape(GraphicalObject g) throws ShapeListPackage.FullException;
    Shape[] allShapes();
    int getVersion();
}
```

Figure 17.3 *ShapeListServant* class of the Java server program for CORBA interface *ShapeList*

```
import org.omg.CORBA.*;
class ShapeListServant extends _ShapeListImplBase {
    ORB theOrb;
    private Shape theList[];
    private int version;
    private static int n=0;
    public ShapeListServant(ORB orb){
        theOrb = orb;
        // initialize the other instance variables
    }
    public Shape newShape(GraphicalObject g) throws ShapeListPackage.FullException {1
        version++;
        Shape s = new ShapeServant( g, version);
        if(n >=100) throw new ShapeListPackage.FullException();
        theList[n++] = s;
        theOrb.connect(s);                                                    2
        return s;
    }
    public  Shape[] allShapes(){ ... }
    public int getVersion() { ... }
}
```

Figure 17.4 Java class *ShapeListServer*

```
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
public class ShapeListServer {
    public static void main(String args[]) {
        try{
            ORB orb = ORB.init(args, null);                                  1
            ShapeListServant shapeRef = new ShapeListServant(orb);           2
            orb.connect(shapeRef);                                           3
            org.omg.CORBA.Object objRef =
                orb.resolve_initial_references("NameService");               4
            NamingContext ncRef = NamingContextHelper.narrow(objRef);
            NameComponent nc = new NameComponent("ShapeList", "");           5
            NameComponent path[] = {nc};                                     6
            ncRef.rebind(path, shapeRef);                                    7
            java.lang.Object sync = new java.lang.Object();
            synchronized (sync) {   sync.wait();}
        } catch (Exception e) { ... }
    }
}
```

Figure 17.5 Java client program for CORBA interfaces *Shape* and *ShapeList*

```
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;
import org.omg.CORBA.*;
public class ShapeListClient{
    public static void main(String args[]) {
        try{
            ORB orb = ORB.init(args, null);                                  1
            org.omg.CORBA.Object objRef =
                orb.resolve_initial_references("NameService");
            NamingContext ncRef = NamingContextHelper.narrow(objRef);
            NameComponent nc = new NameComponent("ShapeList", "");
            NameComponent path [] = { nc };
            ShapeList shapeListRef =
                ShapeListHelper.narrow(ncRef.resolve(path));                 2
            Shape[] sList = shapeListRef.allShapes();                        3
            GraphicalObject g = sList[0].getAllState();                      4
        } catch(org.omg.CORBA.SystemException e) {...}
    }
```

Figure 17.6 The main components of the CORBA architecture

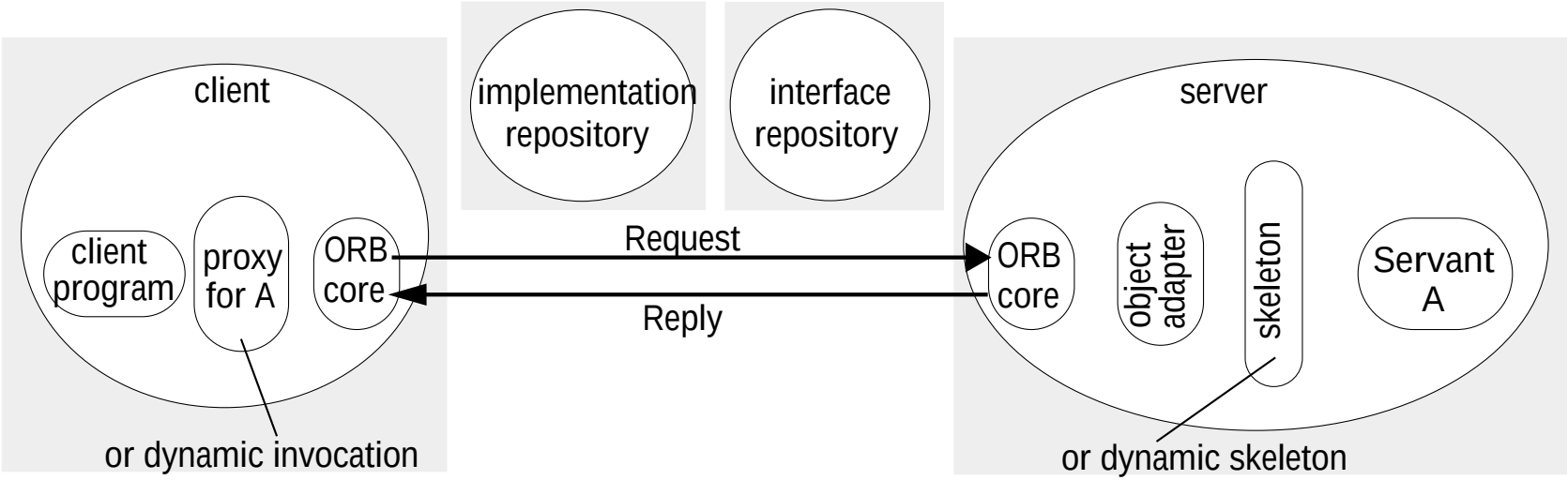

Figure 17.7 IDL module *Whiteboard*.

```
module Whiteboard {
    struct Rectangle{
    ...} ;
    struct GraphicalObject {
    ...};
    interface Shape {
    ...};
    typedef sequence <Shape, 100> All;
    interface ShapeList {
    ...};
};
```

Figure 17.8 IDL constructed types - slide 1.

Type	*Examples*	*Use*
sequence	*typedef sequence <Shape, 100> All;* *typedef sequence <Shape> All* bounded and unbounded sequences of *Shapes*	Defines a type for a variable-length sequence of elements of a specified IDL type. An upper bound on the length may be specified.
string	*string name;* *typedef string<8> SmallString;* unbounded and bounded sequences of characters	Defines a sequences of characters, terminated by the null character. An upper bound on the length may be specified.
array	*typedef octet uniqueId[12];* *typedef GraphicalObject GO[10][8]*	Defines a type for a multi-dimensional fixed-length sequence of elements of a specified IDL type.

this figure continues on the next slide

IDL constructed types - slide 2

Type	*Examples*	*Use*
record	*struct GraphicalObject {* *string type;* *Rectangle enclosing;* *boolean isFilled;* *};*	Defines a type for a record containing a group of related entities. *Structs* are passed by value in arguments and results.
enumerated	*enum Rand* *(Exp, Number, Name);*	The enumerated type in IDL maps a type name onto a small set of integer values.
union	*union Exp switch (Rand) {* *case Exp: string vote;* *case Number: long n;* *case Name: string s;* *};*	The IDL discriminated union allows one of a given set of types to be passed as an argument. The header is parameterized by an *enum*, which specifies which member is in use.

CORBA interoperable object references

IOR format

IDL interface type name	Protocol and address details			Object key	
interface repository identifier	IIOP	host domain name	port number	adapter name	object name

Figure 17.9 Naming graph in CORBA Naming Service

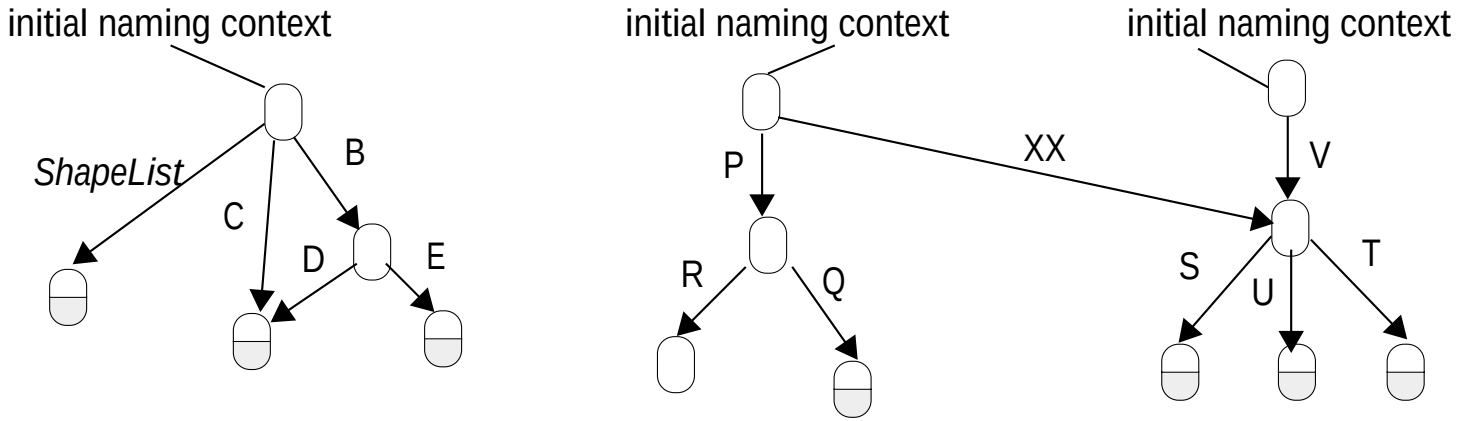

Figure 17.10 Part of the CORBA Naming Service *NamingContext* interface in IDL

struct NameComponent { string id; string kind; };

typedef sequence <NameComponent> Name;

interface NamingContext {

void bind (in Name n, in Object obj);
binds the given name and remote object reference in my context.

void unbind (in Name n);
removes an existing binding with the given name.

void bind_new_context(in Name n);
creates a new naming context and binds it to a given name in my context.

Object resolve (in Name n);
looks up the name in my context and returns its remote object reference.

void list (in unsigned long how_many, out BindingList bl, out BindingIterator bi);
returns the names in the bindings in my context.

};
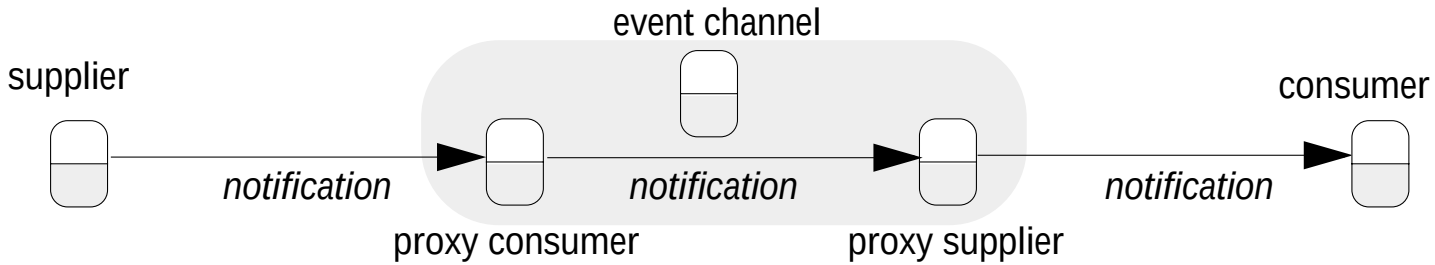
Figure 17.11 CORBA event channels