Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom

TELEPHONE: **Cambridge (01223) 515010**
INTERNATIONAL: **+44 1223 515010**
FAX: **+44 1223 359779**
E-MAIL: **apm@ansa.co.uk**

Training

ANSAwise - CORBA Infrastructure Engineering

Chris Mayers

Abstract

Organizations that wish to make demanding use of CORBA technology will wish to understand its underlying structure and mechanism, so as to select and make best use of it. Such organizations may wish to be able to configure, customize, and even re-implement portions of the infrastructure for improved performance

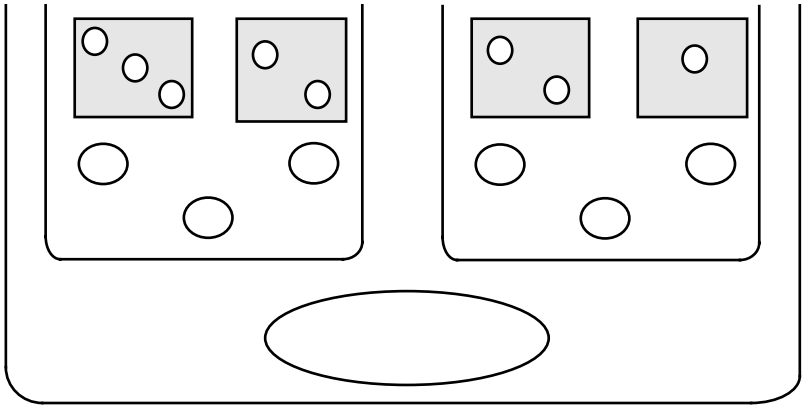
This module of the ANSAwise training programme places the CORBA infrastructure in the context of the ODP Engineering Model (although these concepts are not used directly. Issues with stubs are discussed. Finally some brief mention is made of vendor-specific mechanisms (in IONA's Orbix).

APM.1543.01 **Approved** 31st July 1995

Briefing Note

Distribution:
Supersedes:
Superseded by:

CORBA Infrastructure Engineering



In this session

- *Examine the roles of the elements of an ORB infrastructure*
- *Explain the various types of transparency mechanisms*
- *Explore some vendor-specific mechanisms*

Engineering is concerned with trade-offs

- *For example*
  - flexibility versus performance
  - time versus space
  - ... and many others
- *Using many of these trade-offs requires access to the ORB infrastructure*
  - but some trade-offs can be done entirely within applications

An application trade-off - object placement

- ***Place objects in the same object implementation (process)***
  - **for efficiency of communications**
  - **for efficiency by exploiting shared state**
- ***Place objects in different object implementations***
  - **for robustness**
  - **for security**
  - **for flexibility of configuration**
  - **to avoid competing for same resources**

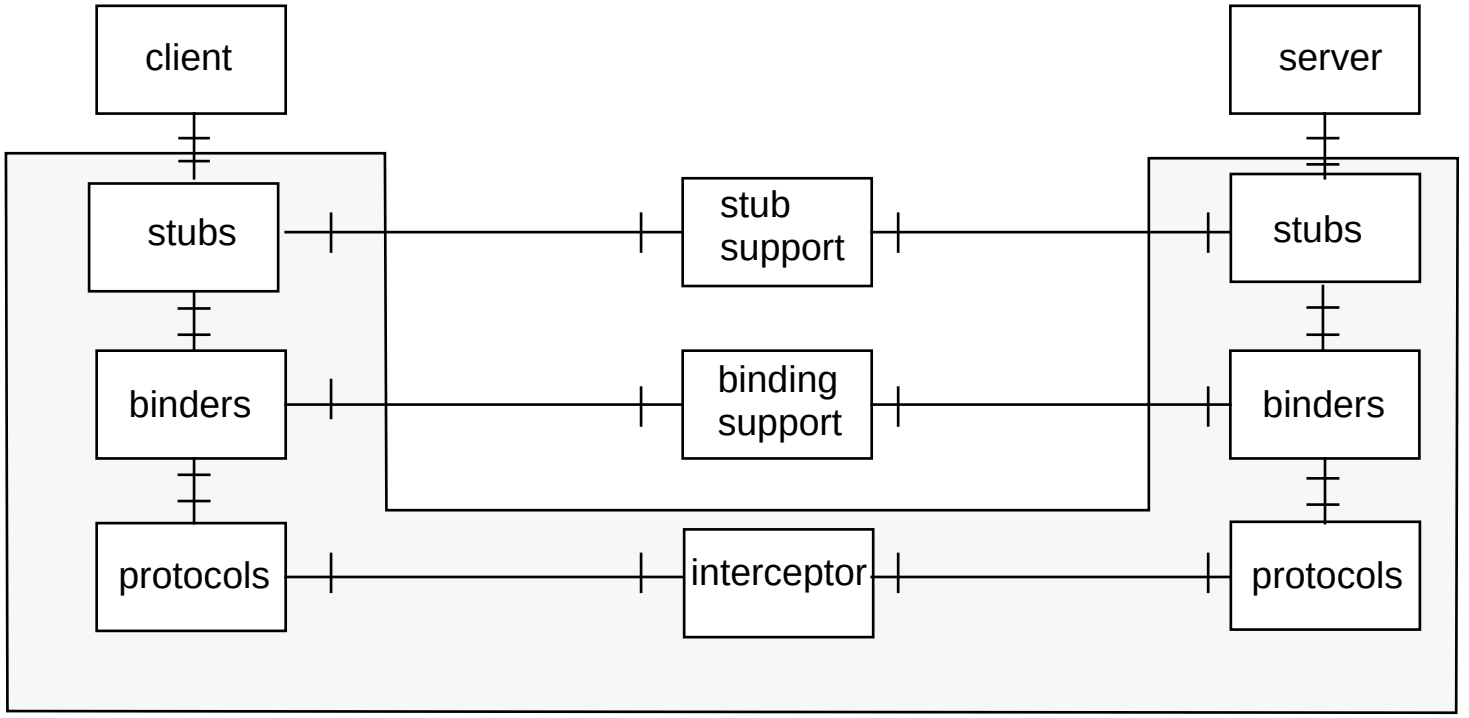
Object Implementations and Interfaces

- ***Objects in the same object implementation can still invoke each other's operations***
  - **you are not compelled to exploit shared state**
- ***Operations are invoked in the same way...***
  - **within a object implementation**
  - **between two object implementations on the same node**
  - **between two nodes**
- ***...the infrastructure should optimize communications between objects on the same node***

A general model for channels

- ***Channels are communication paths between objects***
- ***Channels may be:***
  - **1 to 1 (point-to-point)**
  - **1 to many (point-to-multipoint: not yet supported by CORBA)**
- ***Channels may be***
  - **operational**
  - **stream (not yet supported by CORBA)**
- ***Channels are layered***
  - **built from *stubs*, *binders*, and *protocol objects***
  - **there may be multiple protocols in a particular infrastructure...**
  - **...layering hides the diversity from the application**

Point-to-Point Client-Server Channel



Stubs, Binders, and Protocol Objects

- *Stubs provide data conversion*
  - for example, the GIOP CDR
- *Binders manage end-to-end integrity and quality-of-service*
- *Protocol objects provide communication*
- *... Most application developers will only be aware of stubs*
  - and even these will probably be generated automatically
- *Stubs should be independent of binders and protocols*

Stubs

- ***Typically, there will be one stub per interface***
  - **with separate code for each operation**
- ***Careful design of the stub code is necessary to avoid large amounts of code being generated***
- ***Stubs marshal the invocation parameters in and out of a (linear) buffer***
- ***Important optimizations include***
  - **not copying the data more than once**
  - **using out-of-line marshalling (to share marshalling functions between stubs)**
- ***Stubs must be careful with garbage-collection***

Binding

- ***Binders establish end-to-end connections***
- ***Binding may be either implicit or explicit***
- ***Binding is usually implicit for operational interfaces***
  - **explicit binding may be helpful if you need precise control over resource allocation, and when allocation takes place**
- ***Binding is explicit for streams***

Objectives for the engineering infrastructure

- *Do not allocate resources that are never used*
- *Allocate resources as late as possible*
- *Share resources as much as possible*
- *Release resources as early as possible*
- *Match the distribution of resources to the scale of the demand*

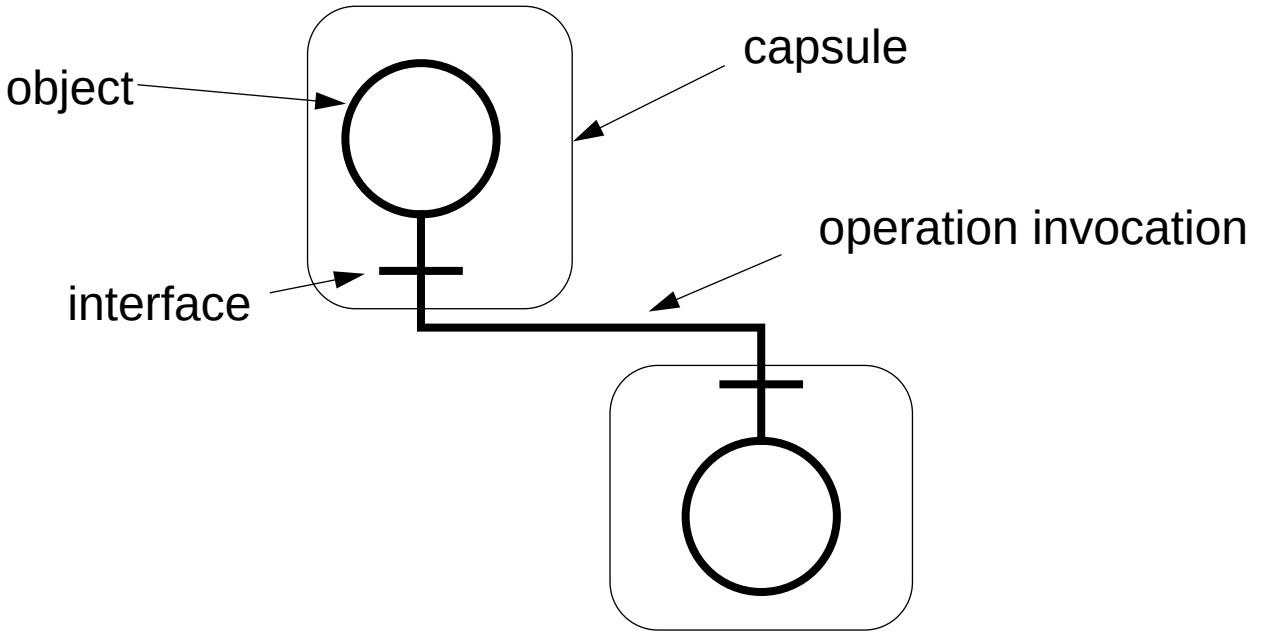
Quality-of-service considerations may constrain them

Transparencies - Simplifying distribution

- ***Remember that in a distributed system, traditional design assumptions must be reversed***
  - **for example, mobility: objects do not stay in one place, they can migrate**
- ***Must isolate the specification of transparencies from their design***

Transparency examples

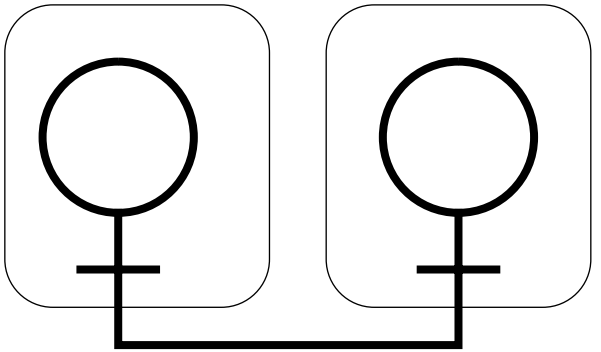
- ***In these examples the diagrams are slightly simplified***
- ***This shows an object invoking an operation from another capsule***



Selective Transparency Engineering - Location

- ***Location Transparency***

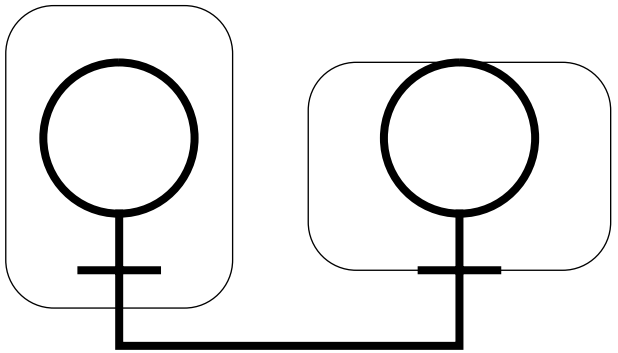
  - **application need not know where object is to use it**



  - **objects may be in the same capsule, different capsules, or different nodes**

Selective Transparency Engineering - Access

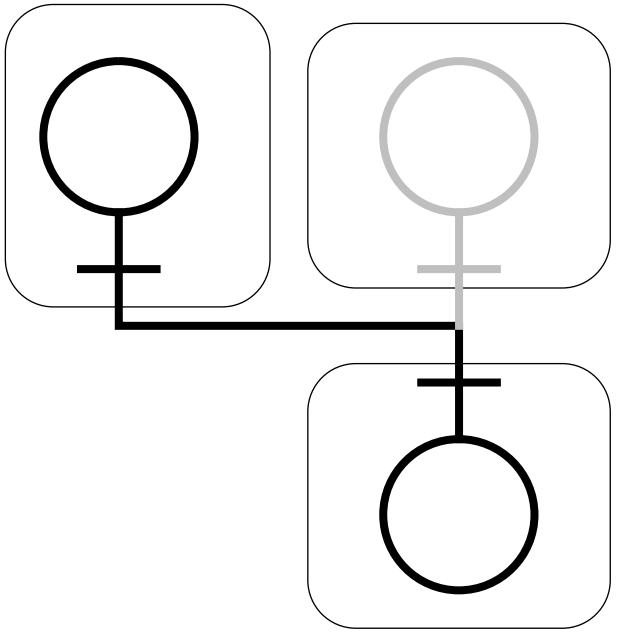
- ***Access Transparency***
  - **application need not know the type of machine where the object is executing**



  - **objects may be in capsules on different operating systems, on different processor types (mainframe, workstation, or PC),...**

Selective Transparency Engineering - Migration

- ***Migration Transparency***
  - **application need not know where the object has moved to**

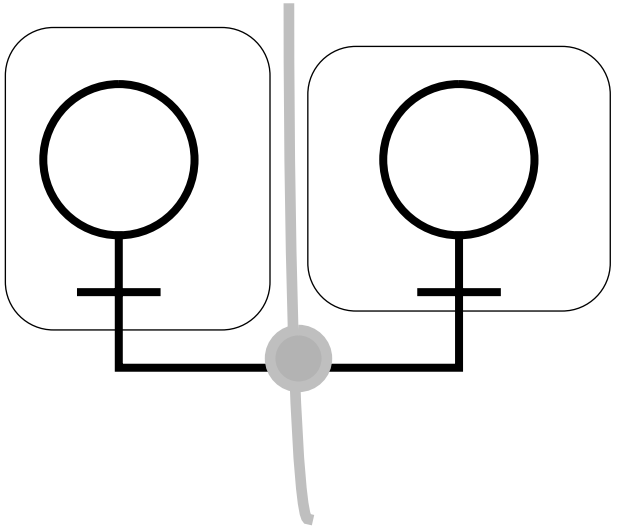


Migration Transparency

- ***Object migration needed:***
  - when a node fails, and its capsules have to be moved to another node
  - for load-balancing between capsules
- ***Like a stronger form of location transparency***
  - relies on location transparency mechanism

Selective Transparency Engineering - Federation

- ***Federation Transparency***
  - **application need not know where administration boundaries are**



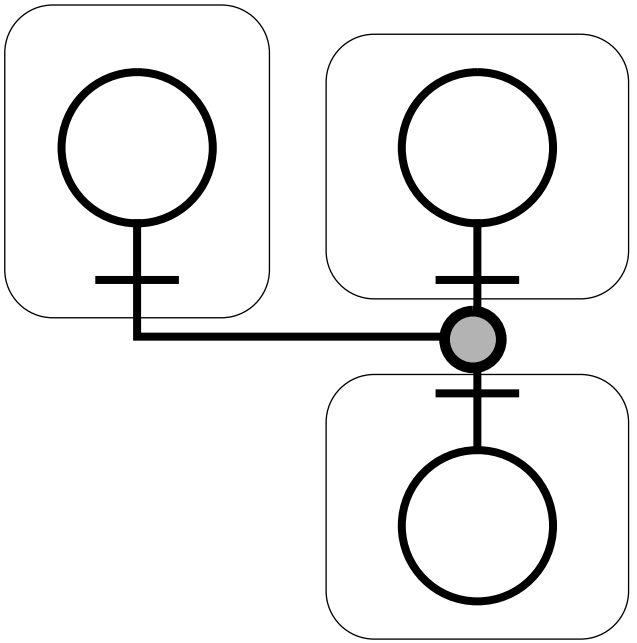
  - **interception may happen at the boundary, but this is not visible to the application**

Federation Transparency

- ***Federation is an Enterprise issue***
  - **there are many different kinds of federation boundaries: administration, organizational, contractual, and so on**
  - **constructing the transparency requires Enterprise knowledge**
- ***Federation is an ANSA research area***
  - **how it relates to trading**
  - **part of ANSA Phase III**

Selective Transparency Engineering - Replication

- ***Replication Transparency***
  - **application need not know how many copies**



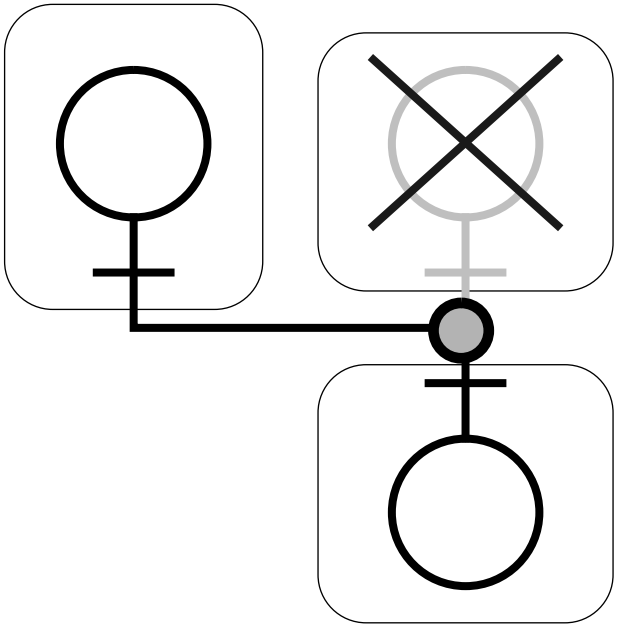
  - **application only sees a single interface**

Replication Transparency

- *Server objects are members of a group*
- *Replication transparency uses special mechanisms to make sure the group members are consistent*
  - for instance, it may use multi-point channels and special protocols
- *Implementing replication transparency efficiently is difficult*
  - it may need information from the application
  - it is under active research in the distributed systems community

Selective Transparency Engineering - Failure

- ***Failure Transparency***
  - **application need not know when an object fails**



  - **may use replication transparency to achieve this**

Other transparencies

- ***Security***
  - application need not be aware of security policy
- ***Concurrency***
  - application need not be aware of other concurrent operations
- ***Transaction***
  - applications need not be aware of inconsistent states

Transparencies in CORBA

- ***CORBA implementations support access and location transparencies***
  - **but none of the others**
- ***CORBA does support Object Services for Concurrency and Transactions***
  - **but these are not transparent...**
  - **... applications must interact with these services explicitly**
- ***Other transparencies will appear in ORB implementations eventually***

Some vendor-specific mechanisms

- ***These are supported by some CORBA products, but are not part of the CORBA specifications***
  - **therefore, using these mechanisms will affect applications portability...**
  - **...and remember that these mechanisms usually impose an overhead, even on applications that do not need them**
- ***Examples include***
  - **filters**
  - **smart proxies**
  - **loaders**
  - **locators**

Filters in Orbix

- *Filters allow application code to intervene at specific invocation points*
  - before and after marshalling, on requests and replies, at client and server
- *Filters are implemented as C++ objects*
- *Filters can also be attached to individual CORBA objects*
- *Filters can be used for*
  - monitoring
  - debugging
  - audit trails
  - ...

Smart proxies in Orbix

- ***Smart proxies allow the client side of an interface to do some of the work of the server (object implementation)***
  - **effectively an extension of a stub**
- ***Smart proxies can be used for***
  - **caching**
  - **access to local physical resources**
  - **load balancing among replicas**

Loaders in Orbix

- *If an invocation arrives at the destination, but the object cannot be found, an object fault occurs*
- *Loaders allow applications to intercept the object fault and activate the object*
  - reading its state from persistent storage
  - creating the corresponding data structures
- *Loaders can be per class, per process, or per database*
- *Loaders can be an alternative to the CORBA Persistence service*

Locators in Orbix

- ***Locators are used at bind time when the destination of an invocation is not yet known***
- ***Note that a inefficient locator can waste time***
  - **it can never cause the wrong object to be invoked**
- ***Locators can be used when control is needed over selection of the destination***
  - **in some systems, static lookups may be best...**
  - **... in others, random lookups may be best**

Summary

- ***Engineering is concerned with trade-offs and the efficient use of resources***
- ***For more information:***
  - **on stubs and binding, see *The ANSA Binding Model* (APM.1392)**
  - **on transparency mechanisms, see *The Challenge of ODP* (TR.033.02)**
  - **on replication transparency and groups, see *A Model for Interface Groups* (AR.002.01)**
  - **on federation, see *The ANSA Model for Trading and Federation* (AR.005.00)**
  - **on streams, see *Integrating Multimedia into the ANSA Architecture* (TR.028.00)**