



**Poseidon House
Castle Park
Cambridge CB3 0RD
United Kingdom**

TELEPHONE:
INTERNATIONAL:
FAX:
E-MAIL:

**Cambridge (01223) 515010
+44 1223 515010
+44 1223 359779
apm@ansa.co.uk**

Training

ANSAwise - The Engineering Model

Chris Mayers

Abstract

Organizations wishing to understand how to design and build distributed systems need to appreciate engineering concepts to enable them to make trade-offs in these systems.

This module of the ANSAwise training programme describes the ANSA/ODP Engineering Model, covering each construct and concept briefly, from the top down. It compares and contrasts it with the Computational Model.

It does not describe applications of these features; these are covered in other modules of the training programme. Similar, quality-of-service issues (for example, performance) are not covered here.

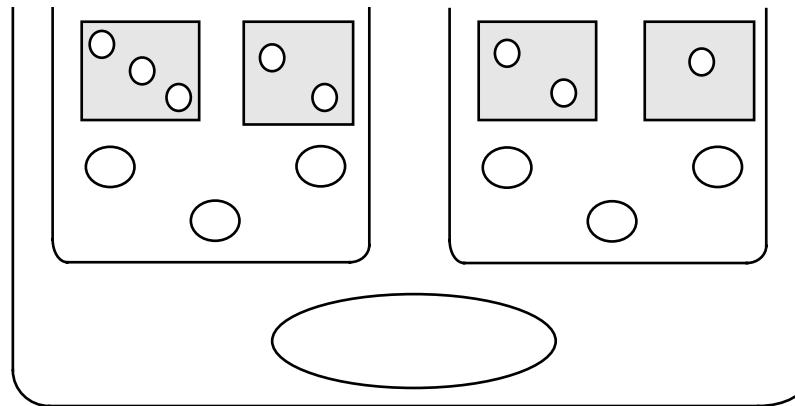
APM.1351.01

Approved
Briefing Note

25th November 1994

Distribution:
Supersedes:
Superseded by:

The Engineering Model





In this session

- *Explain the concepts in the Engineering Model*
- *Explain the various types of transparency mechanisms*
- *Compare the Engineering and Computational Models*



What is the Engineering Model for?

- *To allow trade-offs*
 - flexibility versus performance
 - time versus space
 - ... and many others
- *But without affecting the Computational Model*

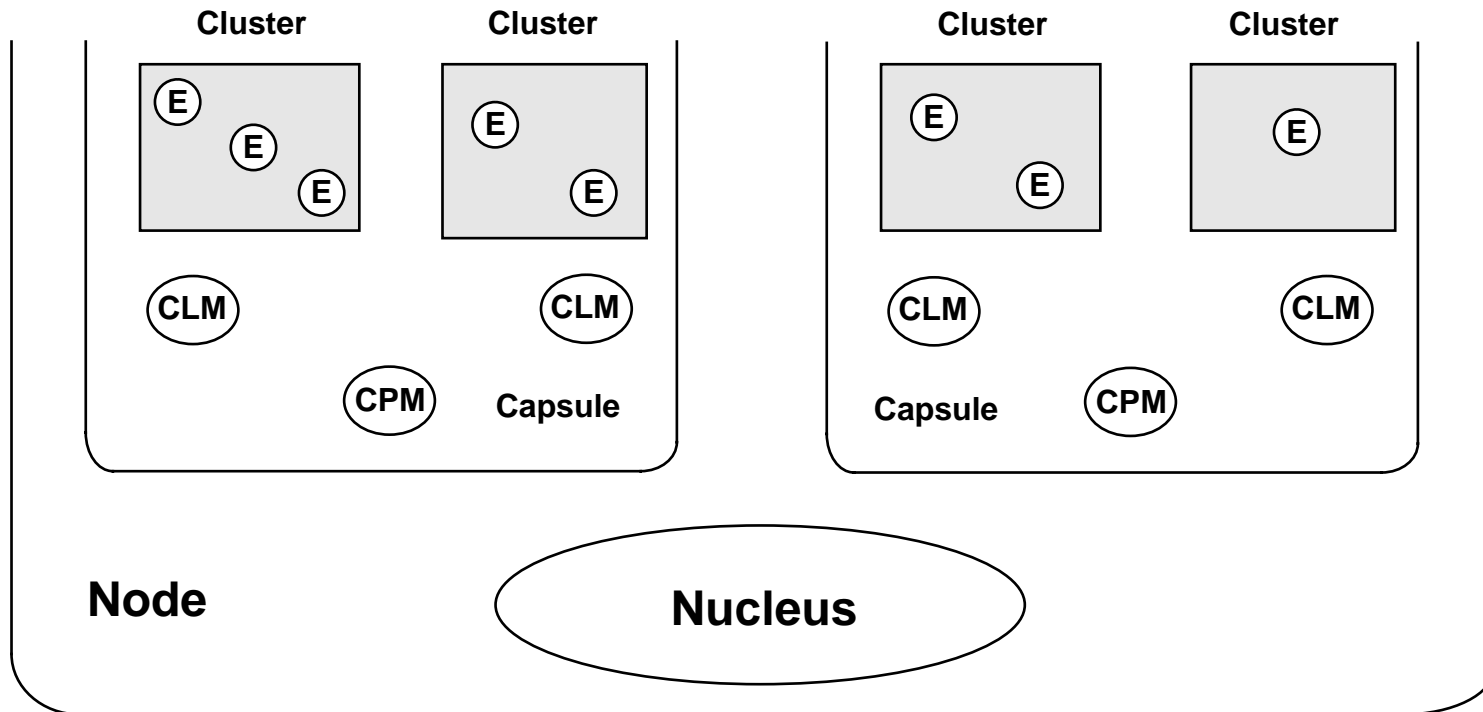


Engineering is infrastructure

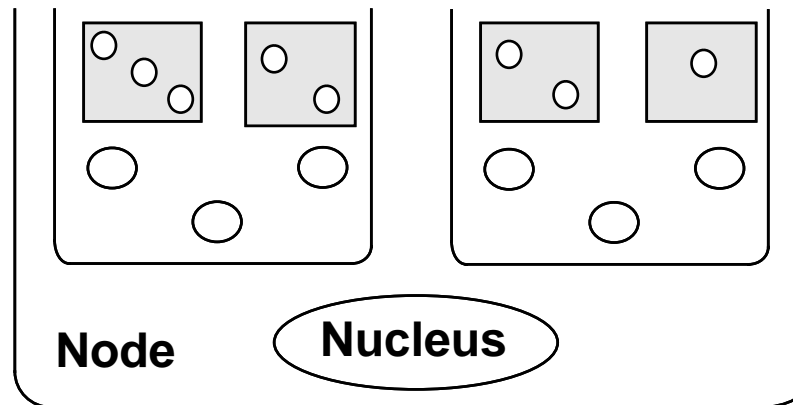
- ***Provides infrastructure for encapsulation***
 - ***capsules*** for resource allocation and protection
 - ***clusters*** for collective activation, deactivation, and migration
 - ***nodes*** for network addressing
- ***Provides channels for communication***
 - **point-to-point** (simple client-server)
 - **point-to-multipoint**
 - **streams**
- ***Provides concurrency mechanisms***
 - ***threads*** for concurrent execution
- ***Provides transparency mechanisms***



Engineering Encapsulation Infrastructure

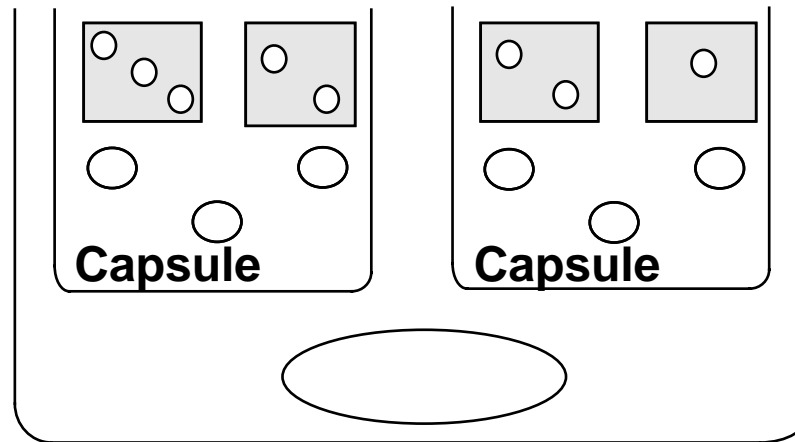


Node and Nucleus



- *A node is the unit of network addressing*
- *The nucleus handles communications between nodes and capsules*
 - *it may be part of the operating system (if there is one), or a layer on top of it*

Capsules



- ***Capsules contain objects***
 - **objects must be encapsulated**
 - **every object is contained in a capsule**
 - **different capsules separate the objects**



Capsules can contain more than one object

- *Objects of the same type, or of different types*
- *Put more than one object in a capsule to*
 - *share resources*
 - *share state information*



Capsules and Shared Memory

- *In a distributed system, two objects cannot usually share memory*
 - the objects could be distributed anywhere in the world!
 - but if particular objects do, their implementation may be more efficient
 - ... for example, you might keep objects of the same type together in the same capsule
- *Objects in the same capsule can share memory for efficiency - but then there is no protection boundary between them*
 - resources: they compete for the same resources (physical and virtual memory)
 - robustness: object failure within the capsule can cause the whole capsule to fail
 - security: there is no security between objects in the same capsule
- *Sharing memory between different capsules is not permitted*



Capsules and Objects

- *So, the Computational and Engineering Models of an object are different:*
 - in the Computational Model, every object is separately encapsulated; objects cannot share state
 - in the Engineering Model, objects can share a capsule, and share state



Capsules and Interfaces

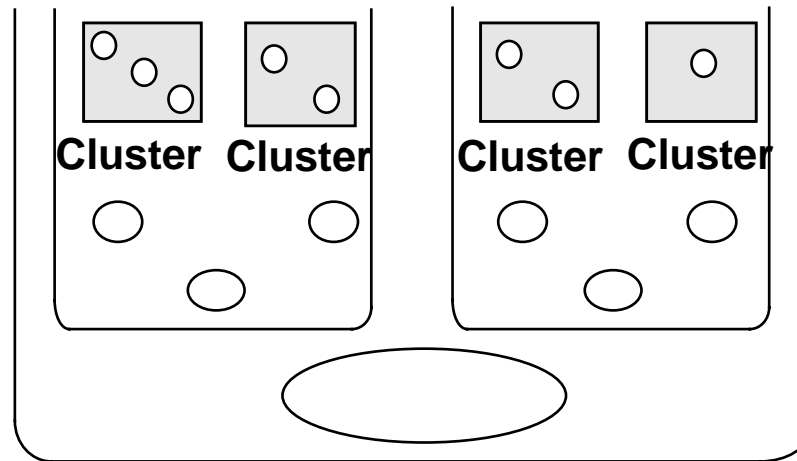
- *Objects in the same capsule can still invoke each other's interfaces*
 - you are not compelled to use shared memory within a capsule
- *Object interfaces are invoked in the same way...*
 - within a capsule
 - between two capsules on the same node
 - between two nodes
- *...the infrastructure will optimize communications between objects on the same node*



Nodes and Capsules

- *Most systems can support more than one capsule per node*
 - using the multi-tasking and memory protection features of the operating system
 - Unix can...
 - ...DOS can't (only one capsule per node)

Clusters



- ***A cluster is a collection of objects that must be 'kept together'***
 - **if an object moves (*migrates*) between capsules, objects that share memory must move together with that object**
 - **if an object fails, you might want to shut down other objects automatically**
 - **when starting up one object, you might want to start up other objects automatically**



Clusters and Capsules

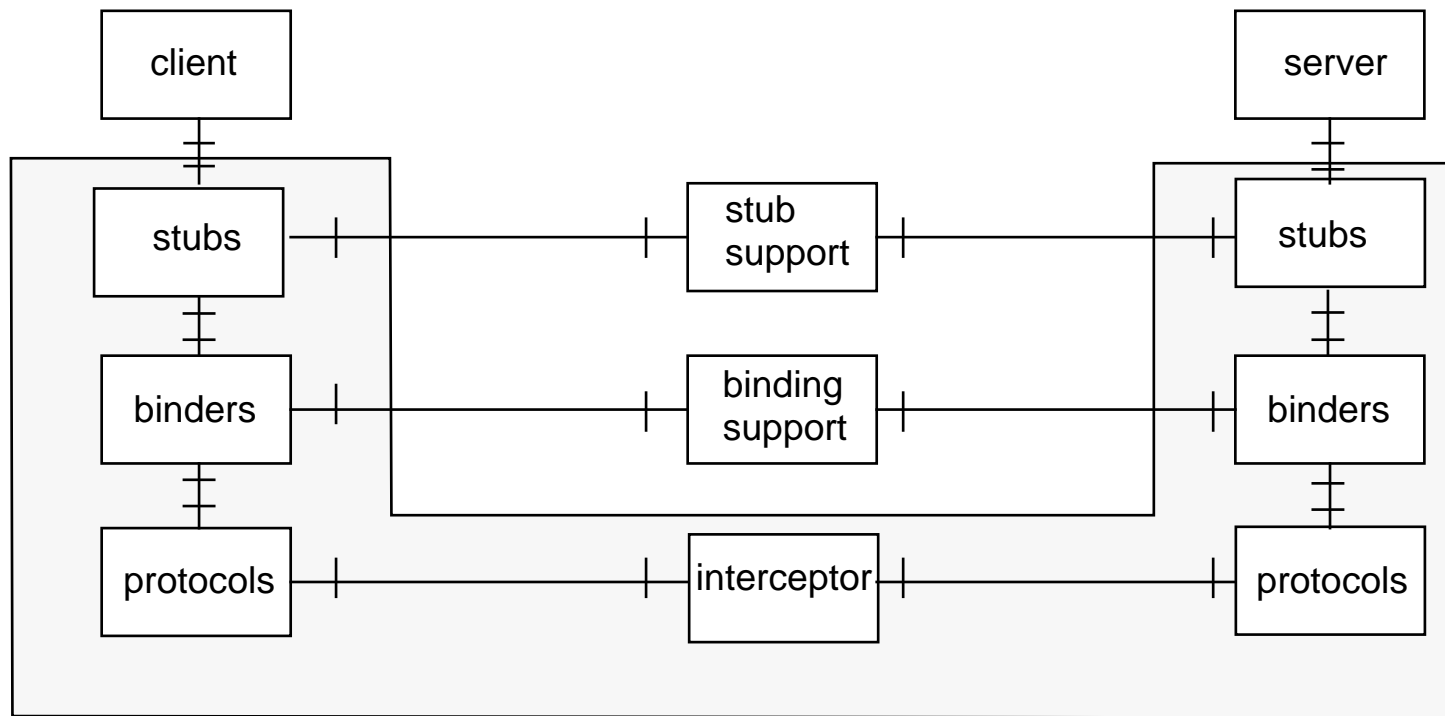
- *A capsule can support multiple clusters*
- *A Cluster Manager handles this for each cluster*
 - *it coordinates checkpoints for the cluster (snapshots of its state)*
- *If there are no clusters, no Cluster Manager is needed*



Channels

- ***Channels are communication paths between objects***
- ***Channels may be:***
 - 1 to 1 (point-to-point)
 - 1 to many (point-to-multipoint)
- ***Channels may be:***
 - operational
 - stream
- ***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*
- *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*



Concurrency

- *Capsules on the same node run concurrently*
- *Within a capsule, you can use threads to achieve concurrency*
 - *these may be supported by the operating system, or by the nucleus*
 - *... threads are always available, even under DOS!*
- *On a multi-processor system, different threads in the same capsule can run simultaneously on different CPUs*

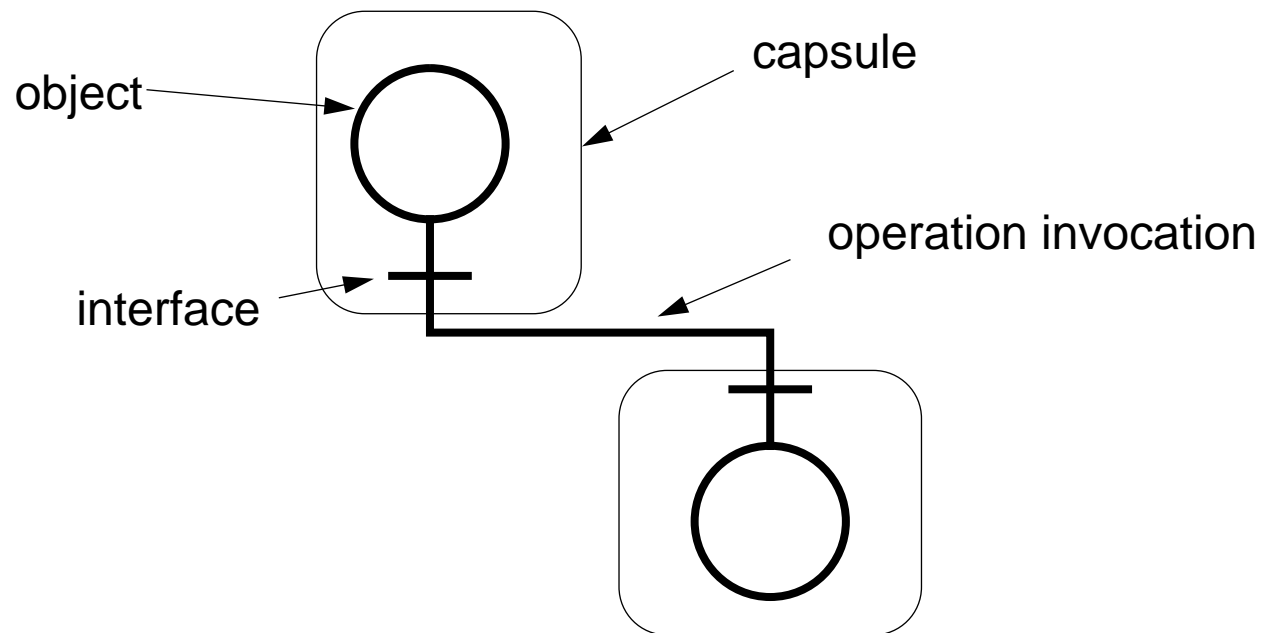


Transparencies - Simplifying distribution

- ***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***
 - Computational Model just assumes the transparencies are provided when needed
 - Engineering Model must show how transparencies are provided from engineering mechanisms
- ***Applications developers simply state which transparencies they require***
 - software tools construct them automatically from the engineering mechanisms

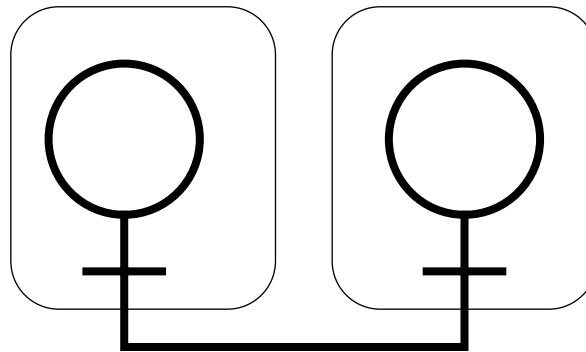
Transparency examples

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



Selective Transparency Engineering - Location

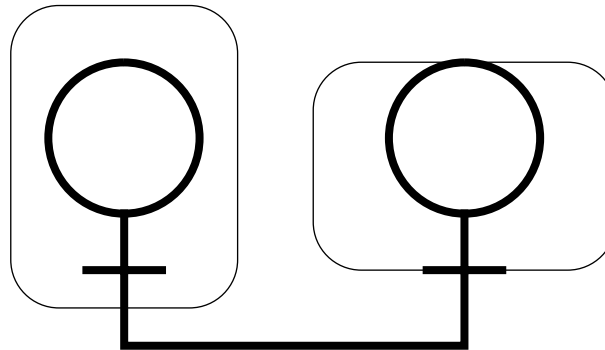
- Location Transparency
 - application need not know where object is to use it



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

Selective Transparency Engineering - Access

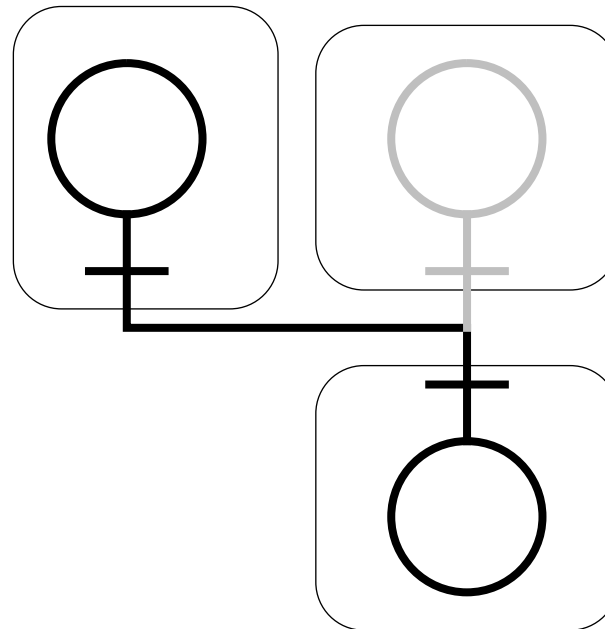
- **Access Transparency**
 - application need not know the type of machine where the object is executing



- 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



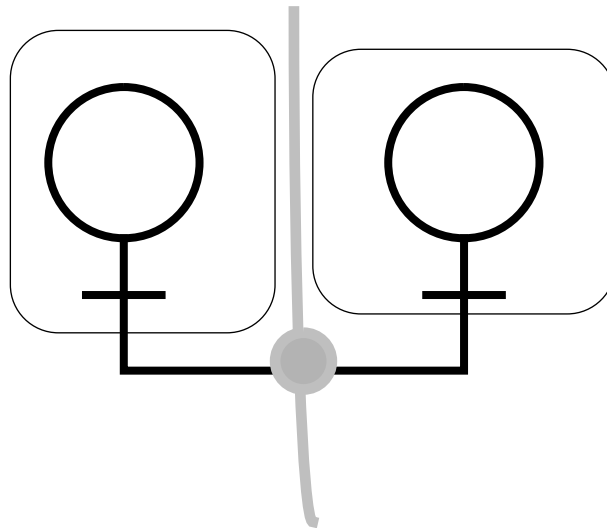


Migration Transparency

- ***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

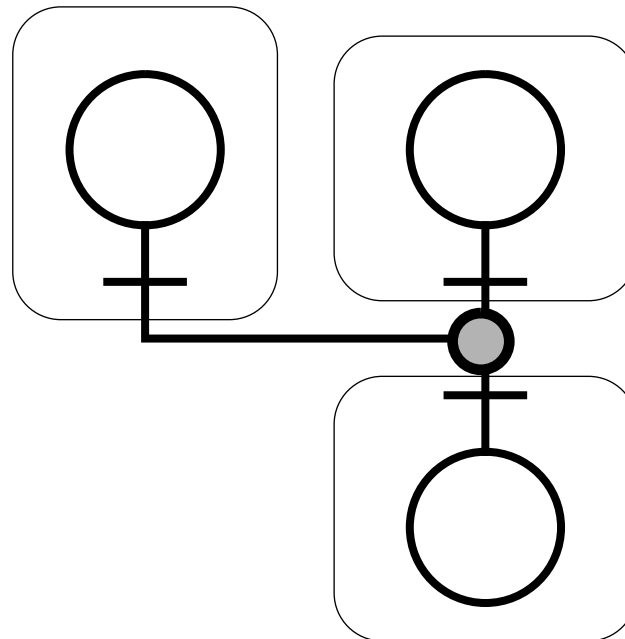


Federation Transparency

- ***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

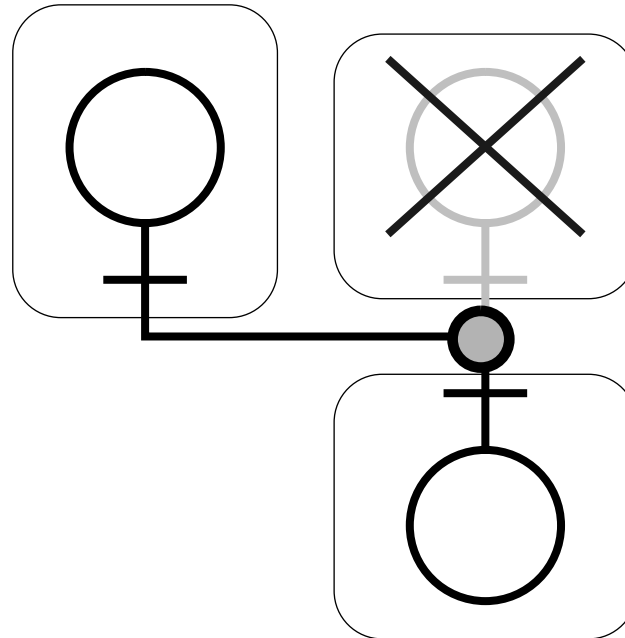


Replication Transparency

- ***Server objects are members of a group***
 - do not confuse with a cluster!
- ***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



Computational and Engineering Models - comparison

- ***Computational Model is for specifying interfaces for distribution***
 - interfaces consist of operations
 - operations have (multiple alternative) terminations
 - everything has a type
- ***Engineering Model is the infrastructure for Computational Model***
 - encapsulation using capsules, clusters, and nodes
 - communication using channels
 - concurrency using threads
 - transparency mechanisms
- ***The Computational Model is pure...***
- ***... the Engineering Model supports trade-offs***



Summary

- ***This has described the Engineering Model***
 - ANSAware implements some, but not all of the Engineering Model...
 - ... and not always as described in the Engineering Model

- ***For more information:***
 - 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)*