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 Object Services (CORBAservices)

Chris Mayers

Abstract

Organizations that are considering using CORBA technology will wish to see examples of objects that are beginning to populate the CORBA Object Management Architecture.

The CORBA Common Object Services are the first of these, but, being somewhat abstract, the specifications can be hard to understand.

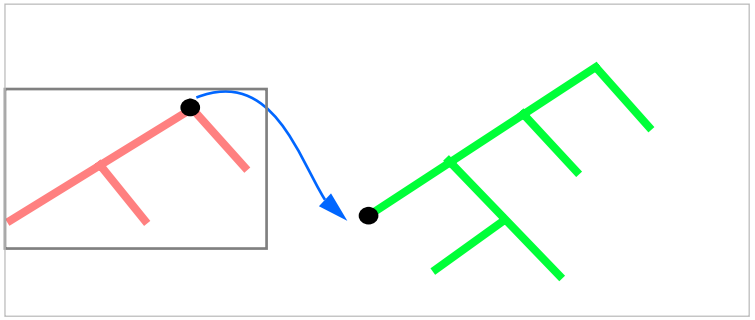
This module of the ANSAwise training programme explains the function of the first Object Services to be specified. It gives a brief summary of Object Services principles, then a summary of each Object service.

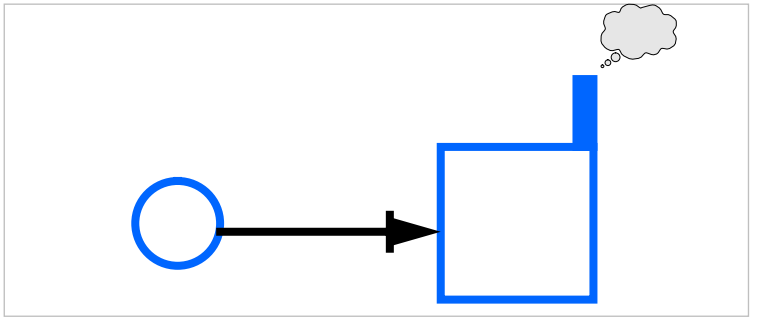
[This module replaces APM.1349; the detailed descriptions of the other Object Services have been transferred to other modules.]

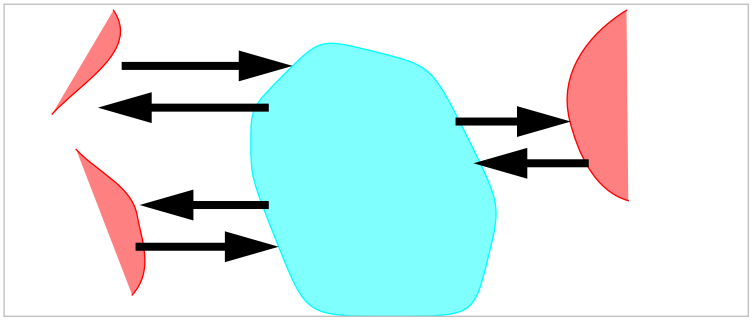
APM.1767.01 **Approved** 13th May 1996
Briefing Note

Distribution:
Supersedes:
Superseded by:

CORBA Object Services (CORBAservices)



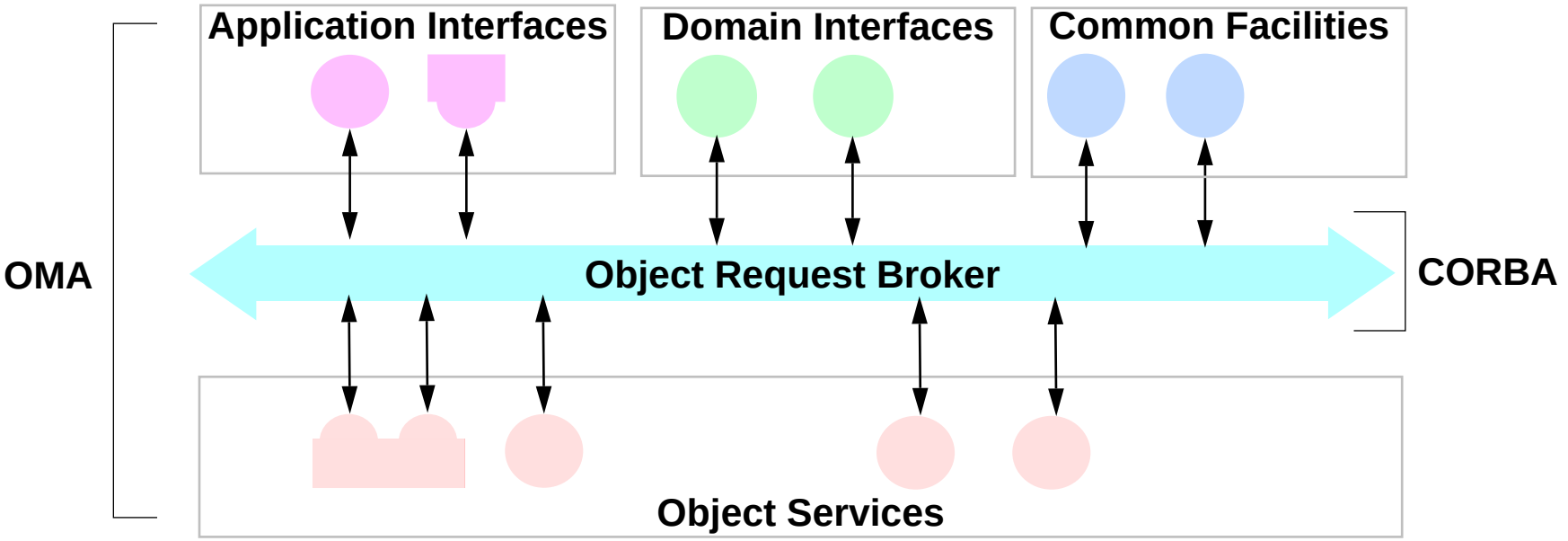




In this session

- **Describe briefly the first Common Object Services to be specified**
- **Describe the general design principles for Object Services**
- **Outline the future Object Services that will be provided**

The Object Management Architecture



- Consists of the Object Request Broker (ORB), plus objects
  - Objects are Object Services, Common Facilities, Domain Interfaces, or Application Interfaces

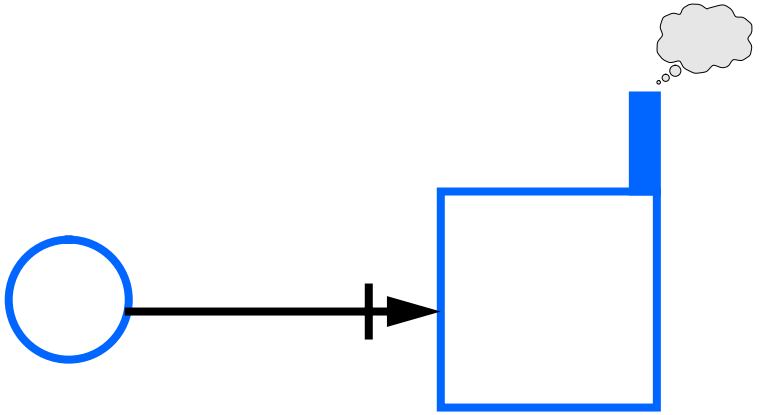
Current and Planned Object Services -1

- Relationships - associations between objects
- Naming - ‘white pages’ service lookup
- Trading - ‘yellow pages’ service lookup
- Persistence - object storage when an object is inactive
- Externalisation - object storage on (removable) media
- Concurrency - control of concurrent operations
- Transactions - serializable operations

Current and Planned Object Services - 2

- Events - asynchronous communication
- Licensing - support for controlling and charging for service usage
- Security - authentication and authorization
- Time - synchronized clocks
- Properties - associating named data with an object
- Query - query language used to select objects from collections
- Collections - collective operations on objects
- Startup - activating objects when the ORB is activated

The LifeCycle service

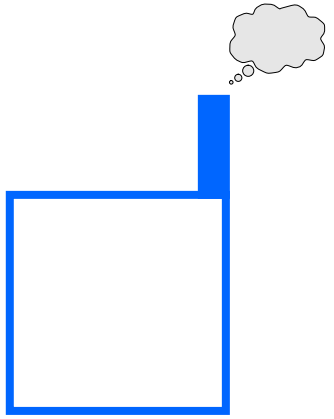


- **Life cycle services allow applications to control**
  - **creation of objects**
  - **removal of objects**
  - **copying of objects**
  - **moving of objects**

Where should objects be created?

- Objects must be created in some location
  - and when they are created, they use resources...
  - ... memory, disk space, CPU time,...
- So, the choice of location must be controlled
  - possibly under client control
  - possibly subject to some administrative policy
- The same issue arises when moving or copying objects
  - for their new locations

Factory objects



- A factory object is an object that can create other objects
- Factories do not have special interfaces
  - they are specified in IDL
- Factories are responsible for allocating resources to objects that they create
  - according to client control/administrative policy

Factory interfaces

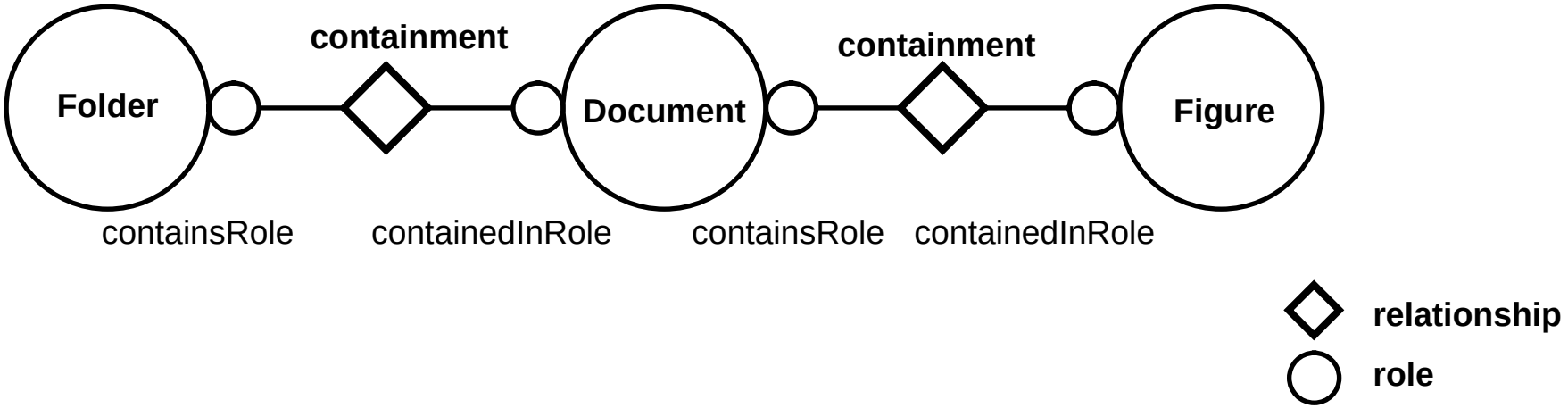
- **Some services contain their own factories**
  - **for example, the Persistence, Externalization, and Property service do**
- **To create an object, an application simply needs the object reference of an appropriate factory**

The Relationship service

- **The Relationship service is a separate service**
  - **allowing the roles of related objects to be represented...**
  - **...for example a document *containing* figures...**
  - **... a figure being *contained in* a document**
- **Appendix A of the LifeCycle specification describes how the LifeCycle service handles relationships**
  - **it handles *containment* and *reference* relationships specially**
- **This allows a client to use the LifeCycle service to copy (or move) a document and all the figures contained in it, for example**

Relationships

- An example of containment
  - folders containing documents, documents containing figures

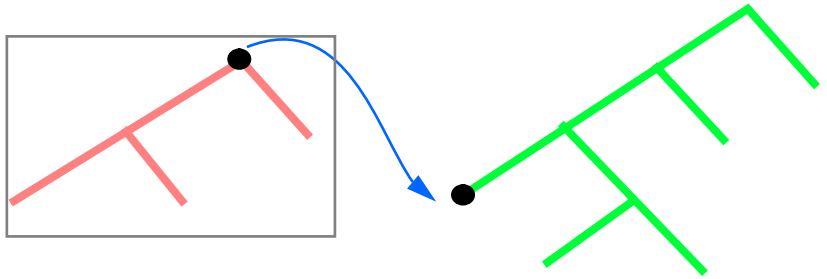


- 2 relationships, 4 roles, 9 objects involved

When relationships are necessary

- **Object references**
  - can be stored persistently, and retrieved later, for invocation
- **Relationships**
  - can be followed in both directions
  - can relate two or more objects
  - can have their own attributes and operations
  - can be manipulated by third parties
- **Relationships are a natural fit to most analysis and design methods**

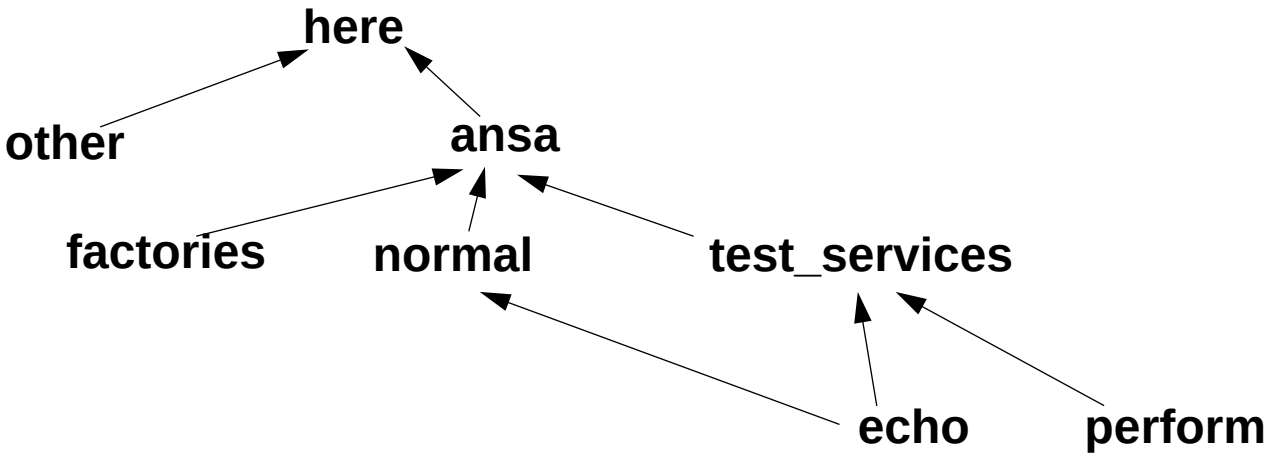
The Naming service



- The naming service is the simplest possible directory service
  - given a name, it will return you an object reference...
  - ...any kind of CORBA object can be named
- The naming service is a ‘white pages’ service...
  - it is not a ‘yellow pages’ service for finding objects from a description
  - ... that would be the function of a separate trading service

Information held by the Naming service

- **The Naming service holds a *naming graph***

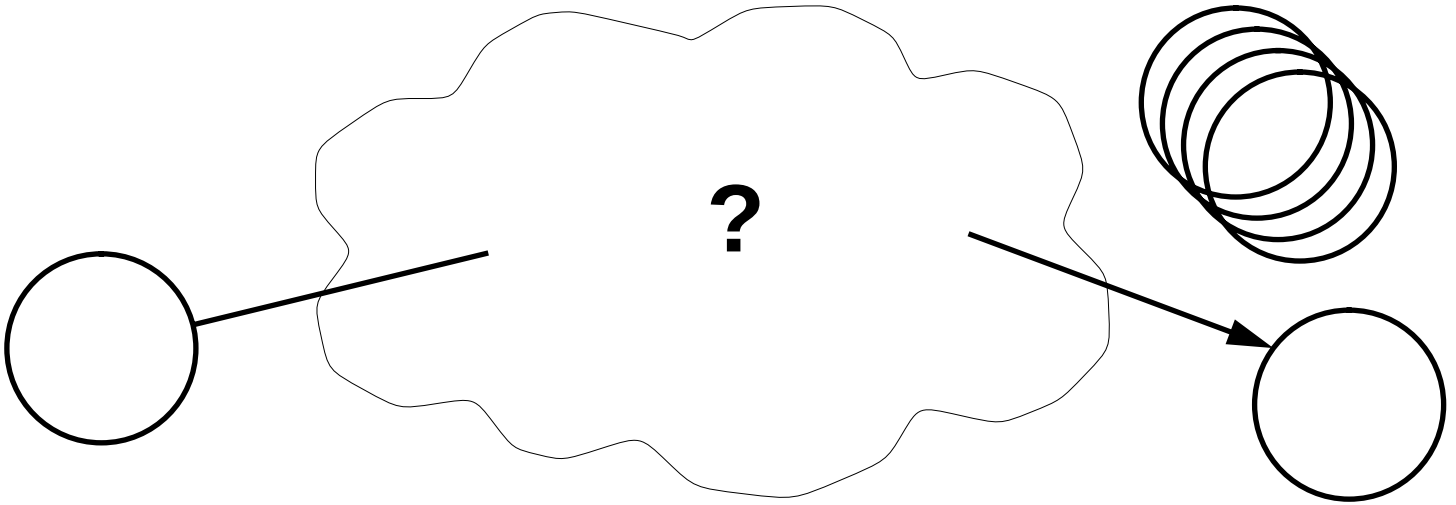


- **... here is a possible example**

A unifying service

- Traditional systems have held separate directories for separate kinds of objects
  - files, processes, disks, tapes,...
- The CORBA Naming service allows you to hold all this information in the one Naming service
- The Naming service allows you to organize the names in any way you like
  - the naming graph can be any shape you like (including loops)
  - the names can be any length and any form you like

The need for Trading



- How can clients find servers that provide the services that they need?
  - in the future, there will be millions of interconnected servers around the world
  - clients will come and go dynamically
  - servers will come and go dynamically

Naming is not enough

- **Trading is necessary**
  - **we cannot rely on clients being able to name servers...**
  - **....the server may not even exist when the client was created**
- **Trading works by matching descriptions provided by clients and servers**
  - **it is a ‘yellow pages’ service**

Matching descriptions - type conformance

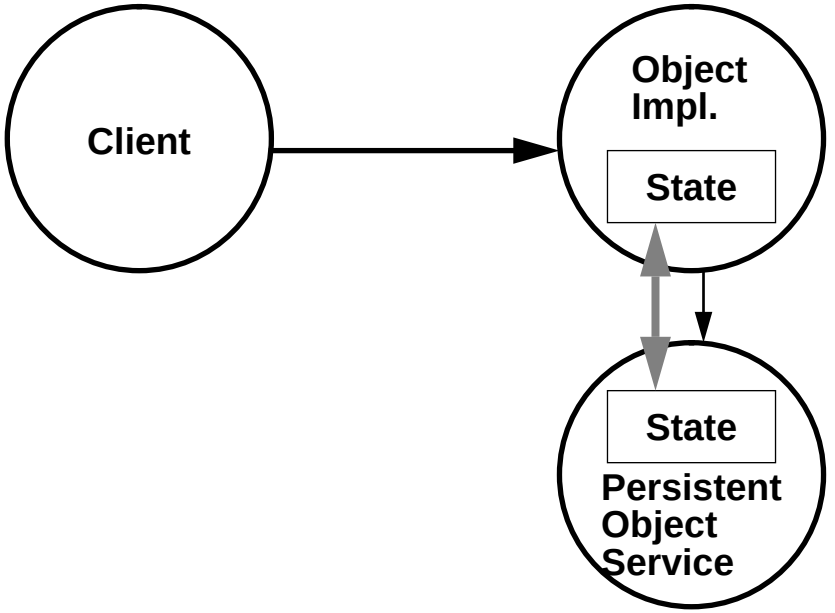
- **How does Trading decide whether a client request matches a server offer?**
  - **it uses the interface *type conformance* concept**
- **If the client request and server offer interface types do not conform, they are incompatible, and cannot match**

Other matching criteria

- Type conformance is not sufficient
  - who owns the service?
  - what will it cost to use?
  - where is the service, and can it be reached?
- These criteria are known as properties
  - (name, value) pairs
- Preference criteria sort matching offers into order
- Scope criteria control where to look for offers

The Persistent Object service

- **Persistent Object Service stores persistent state**



Persistent state

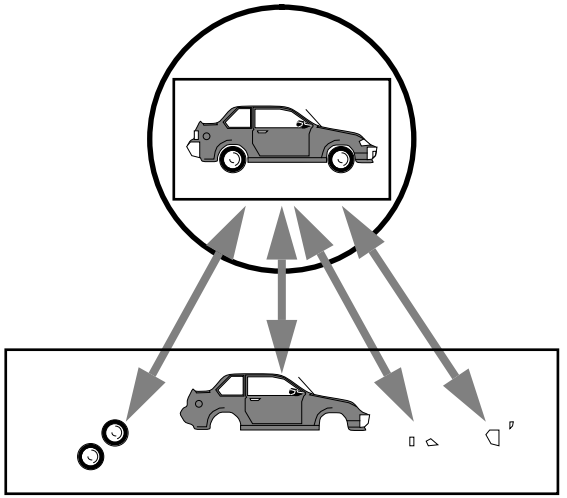
- The state of an object can be treated as two parts
- Dynamic state
  - typically in memory
  - lost if the object crashes
- Persistent state
  - typically on disk
  - preserved over crashes, and can be used to reconstruct the dynamic state

Preserving state

- It is an object's responsibility to preserve its dynamic state as persistent state
  - and to recover it after a crash
- The object may use the Persistent Object service for this purpose
  - or any private mechanism it chooses instead...
  - ... flat files
  - ... direct access to a relational database
  - ... direct access to some other kind of database

Advantages of using the Persistent Object Service

- Typical data storage mechanisms do not have object characteristics
  - uniform interfaces, self-description, and abstraction



- This is sometimes known as an ‘impedance mismatch’

Persistent Object Service Implementations

- As with all CORBA Object Services, a wide range of implementations is envisaged
  - on top of relational databases
  - on top of object-oriented databases
  - on top of specialized compound document storage mechanisms
  - on top of lightweight storage mechanisms
  - ... or any combination of these

The Externalization service

- Externalization is a separate Object Service
- Externalization records an object's state in a stream of data
  - in memory, on disk, across the network
- The stream owns the externalized form
- The externalized form can be stored indefinitely and transported outside an ORB
- The stream can be later internalized
  - into a new object...
  - ...in a different ORB, not necessarily connected to the previous ORB

The Transaction service

- **Supports distributed transactions**
  - **having the usual ACID (Atomicity, Consistency, Isolation, Durability) characteristics**
  - **with the usual commit/rollback outcomes**
- **Allows transactions to be nested**
- **Allows objects to choose whether or not to participate in transactions**
- **Is modelled on the X/Open distributed transaction processing service**
  - **many implementations will be on top of existing TP monitors...**
  - **... integrating with existing transactional protocols**

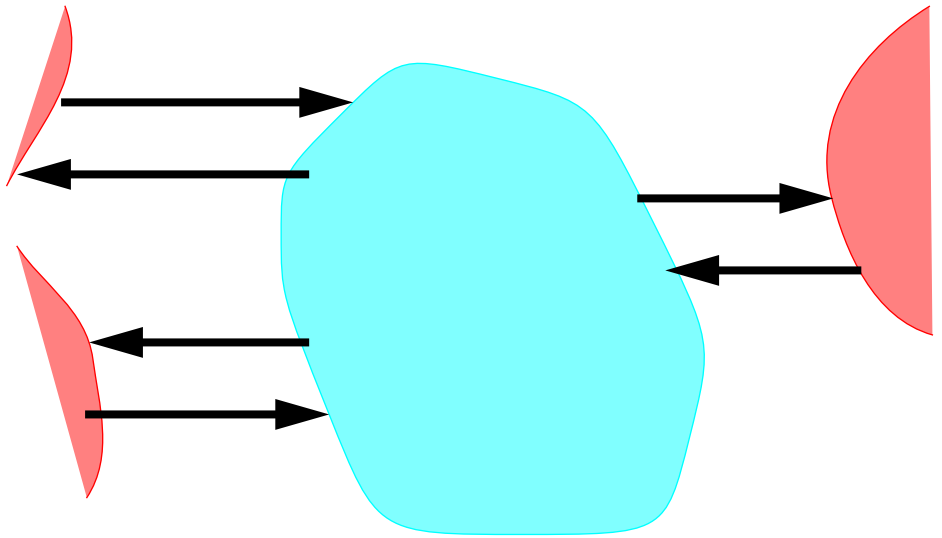
The Concurrency Control service

- Enables coordination of access to shared resources
- The Concurrency Control Object Service does not define what a resource is
  - this is determined by the application
- Supports conventional locks
  - all the usual modes, including intention locks
- Supports transactional and non-transactional modes of operation
  - transactional mode: the CORBA Transaction mode drives the release of locks when transactions commit or abort
  - non-transactional mode: applications are responsible for releasing locks

Implications of the CORBA concurrency approach

- All the lock sets are CORBA objects
  - so can be accessed remotely, and used anywhere a CORBA object can be used...
  - ...even between ORBs
- Specialized implementations of the Concurrency Control service are likely
  - for efficiency
  - for tight integration with the CORBA Transactions service
  - for integration with legacy concurrency control mechanisms

The Event service



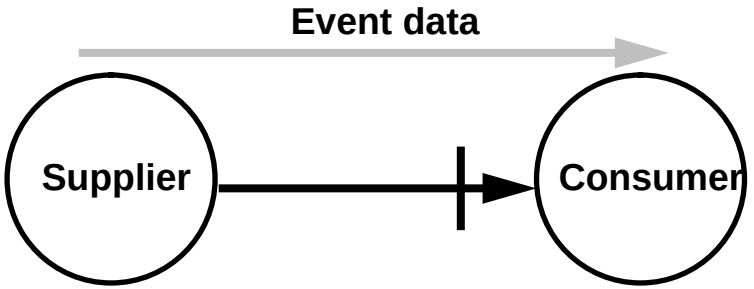
- **Events are concerned with asynchronous communication between objects**

Events

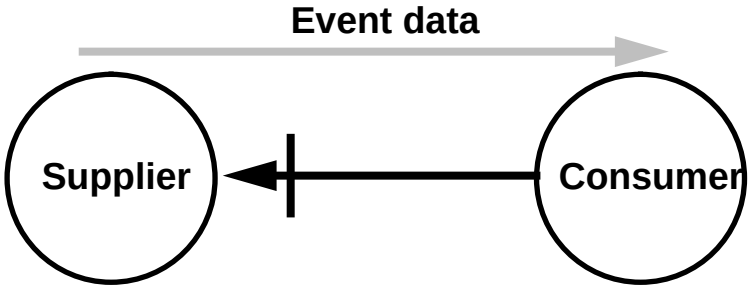
- Events support asynchronous notification
  - ...‘alerts’, ‘change notification’
  - for example, when disk space is getting low
- Suppliers and consumers of events are decoupled
  - via an event channel
- There can be multiple suppliers and multiple consumers
- Events could be used to build an RQM (Robust Queued Messaging) interface
  - or to interface to an existing one

Push and pull models for communicating event data

- **Push model: suppliers push data to consumers**



- **Pull model: consumers pull data from suppliers**



The Licensing service

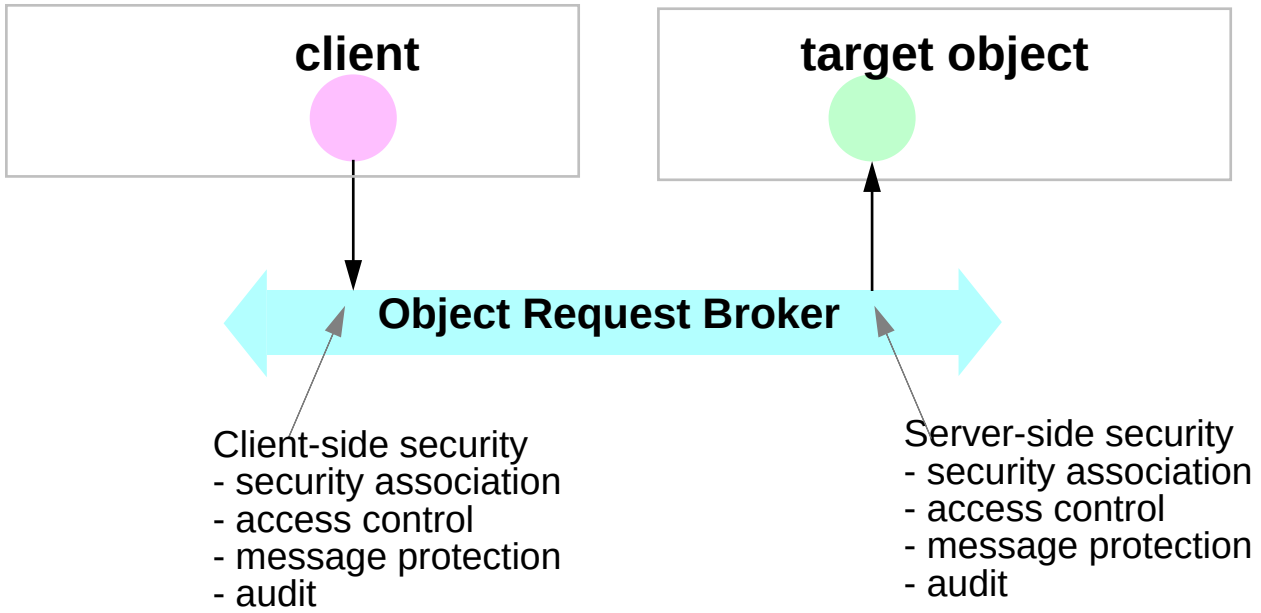
- Allows applications to control their producers' intellectual property
- Supports a wide range of fixed and variable licensing policies
  - concurrent use
  - reservation
  - consumption
  - expiration
- Allows policy enforcement by the applications, and license administration by the customer
- Integrates with the Event service to support periodic licence checks
- Integrates with the Security service to protect producer and customer

The Security service

- Is a security framework, not a single service
- Extends the structure of the ORB to allow for security mechanisms and protocols
- Does not define cryptographic protocols
  - an Encryption Protocol for secure interoperability is being specified separately

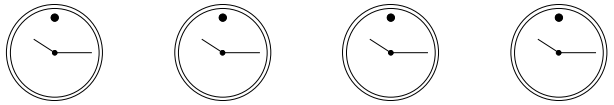
Secure Object Invocation

- **Objects still request the services of other objects via the ORB...**



- **... but with security protection**

The Time service



- The Time service supports
  - retrieval of (synchronized) universal time, including its inaccuracy
  - retrieval of time securely
  - calculation on time intervals (allowing for inaccuracy)
- The TimerEvent service integrates with the Event service
  - supporting events triggered periodically, or at relative or absolute times
- The Time service and TimerEvent service can be used separately
- Operations to convert units of time, and time zones, are not included
  - see the Common Facility

The Properties service

- **Allows typed, named values to be dynamically associated with objects**
  - **attributes defined in CORBA IDL are static...**
  - **... adding a new attribute means changing the IDL**
- **Client applications can get and set both properties and attributes**
  - **properties can be created and deleted...**
  - **... attributes cannot**
- **Properties will be exploited by other services**
  - **Licensing, Trading,...**

The Query service

- Is a general query service and framework, embracing
  - SQL, for relational databases
  - OQL, for object databases
- Allows queries to be prepared, and their status checked later
- Allows queries to be nested

General Object Service design principles

- Build on CORBA Concepts
- Specify Basic, Flexible, and Generic Services
- Allow Local and Remote Implementations
- Consider Reliability, Performance, Scalability, and Portability
- Consider Future Object Services
- Consider Conformance to Existing Standards

Consequences of these principles

- **Object service specifications are small**
  - much smaller than a typical API
  - typically only 100 lines of CORBA IDL, with 20 operations
- **Object service specifications contain several interfaces**
  - typically 2 or 3
- **There are no ‘convenience functions’ that gather together series of operations**
  - these do not belong in the interface, but at a higher level
  - there is only one basic operation for a given function
- **Object service specifications are rather abstract**
  - it can be hard to see how to apply them

Language Independence

- **Because the services are specified in CORBA IDL, you can call them from any programming language**
  - **provided there is a language mapping...**
  - **... C, C++, Smalltalk, Ada so far**

Scalable Service Implementations

- Because distributed systems are going to be very large...
- ... the US telephone network will contain
  - 100,000,000 nodes
  - 1000 administrative domains
  - 100 versions of any software component
- It must be possible to distribute not just access to object services
  - but also the implementation of the services

Obtaining Object Service implementations

- Initially, ask your ORB vendor
- In the future you should be able to ‘shop around’
  - and buy different Object Services implementations from wherever you wish
  - ... with the performance and quality-of-service that you require
- Or you could implement them yourself!
  - to integrate with your own existing event handling software, for instance

CORBAservices: Common Object Services Specification

- This covers many services
  - Naming, Event Management, Persistence, LifeCycle, Concurrency, Externalization, Relationships, Transactions
- Other services have been specified, but are not yet in this manual
  - Security, Time, Licensing, Properties, Query
- The OMG Object Services Task Force are working on new services
  - Collections, Startup, Trading,...

Summary

- For a description of the service design principles
  - see *CORBAservices* by the Object Management Group (Wiley)
- For the specifications of the services, also see CORBAservices
- For future Object Services
  - await future publications