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

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.

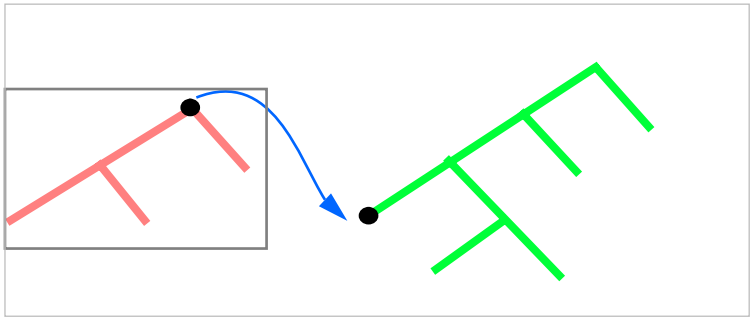
The CORBA Common Object Services are the first of these, but, being somewhat abstract, the specifications can be hard to understand.

This module of the ANSAwise training programme explains the function of the Object Services specified in the CORBA Object Services Specification Volume 1, relates them to the ANSA/ODP Computational Model, and outlines the likely future of Object Services.

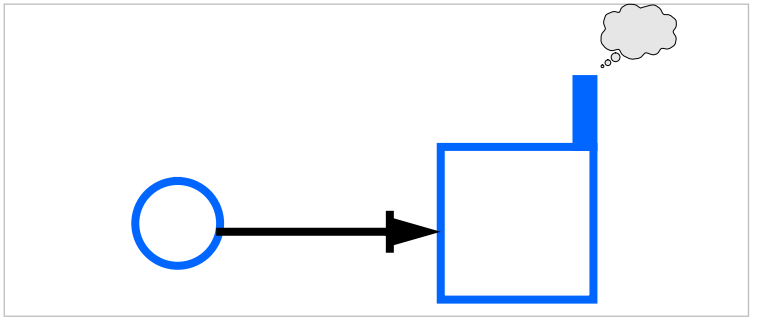
APM.1349.02 **Approved** 20th February 1995

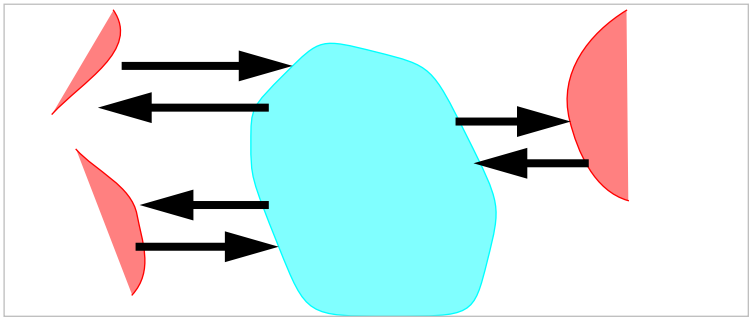
Briefing Note

Distribution:
Supersedes:
Superseded by:



CORBA Object Services

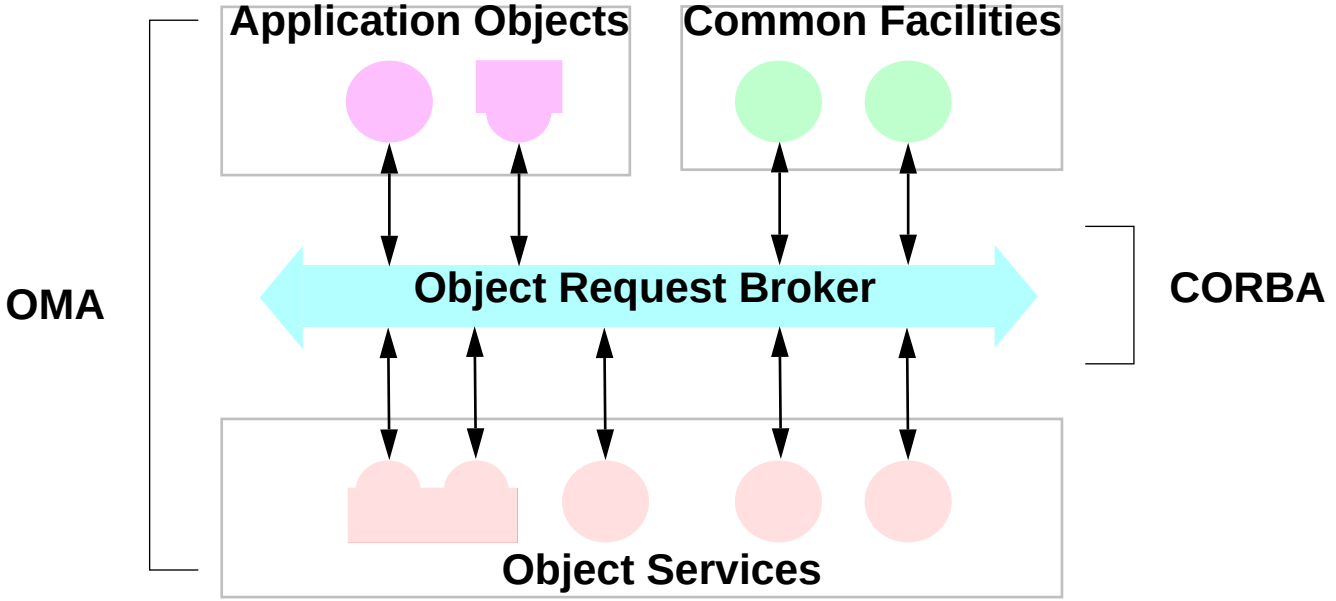




In this session

- ***Describe the general design principles for Object Services***
- ***Explain the first Common Object Services to be specified***
  - **Naming, Events, LifeCycle**
- ***Show some parts of these specifications in detail***
- ***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, or Application Objects**

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***

General Object Service design techniques

- ***Partitioning of interfaces to a service***
- ***Inheritance of specifications***
- ***Callback interfaces***
- ***Avoidance of global identifiers***

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 so far**

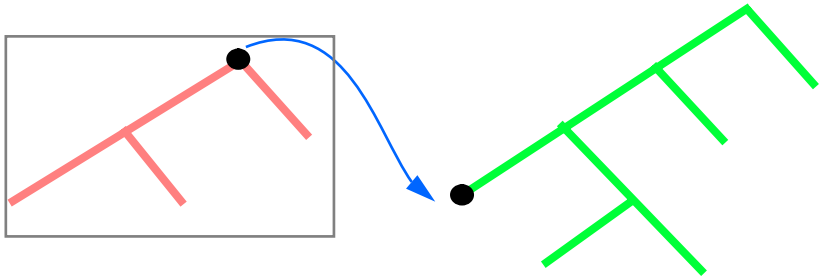
Scalable Service Implementations

- ***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**

Common Object Services Specification, Volume 1

- ***This covers***
  - **Naming**
  - **Events**
  - **Life Cycle**
  - **Persistence**
- ***The first edition of this volume does not contain Persistence***
  - **it has been published as an addendum**

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**

Names

- ***Names can be simple or compound***
  - simple names have one component (like **filenames**)
  - compound names have multiple components (like **pathnames**)
- ***Each component has***
  - an identifier (a **string**)
  - a kind (a **string**)
- ***Consider a typical filename...***

  `myfile.c`
  - the identifier is **myfile.c**
  - the kind is **c_source**
- ***The naming service does not interpret in any way the identifier or kind***
  - it does not understand what they signify; it just matches them

No name interpretation

- ***Naming conventions are not part of the naming service***
- ***The naming service does not have a syntax for names***
  - **it does not parse compound names**

    `mydir/subdir/myfile.c`
  - **there are no wildcards, or case-matching rules**
- ***Applications are responsible for structuring compound names***
  - **for example, this is a three-part structure**

    `ansa.co.uk`
- ***There are no predefined special names, not like***

  `/.:/hosts, /.:/fs, /.:/sec`

IDL for Names

```
module CosNaming
{
   typedef string Istring;// For future internationalization

   struct NameComponent {
      Istring id;
      Istring kind;
   };


   typedef sequence <NameComponent> Name;

   ...

};
```

Naming contexts

- ***A naming context is like a directory***
  - ***names*** **are bound to** ***objects*** **in a** ***naming context***
- ***Naming contexts can be arbitrarily structured***
  - **No ‘universal root’**
  - **No ‘absolute pathnames’**
- ***A naming context is itself an object***

Separation of creation and naming

- ***Creating an object, and binding a name for it...***
  - **...these are two quite separate operations**
- ***So, this is not like 'creating a file' in most file systems...***
  - **...normally, when you create a file, you must give it a name and a directory to put it in**
- ***The naming service is not involved with creating objects***
  - **except for naming context objects themselves**

IDL for Naming Contexts - 1

```
module CosNaming {
   ...
   interface NamingContext {

      ...
      // ... Exceptions omitted,see full specification

      void bind (in Name n, in Object obj)...;
      void rebind (in Name n, in Object obj)...;
      void bind_context (in Name n, in NamingContext nc)...;
      void rebind_context (in Name n,
                           in NamingContext nc)...;

      Object resolve (in Name n)...;
      void unbind (in Name n)...;
```

IDL for Naming Contexts - 2

```
        NamingContext new_context();
        NamingContext bind_new_context(in Name n)...;
        void destroy ()...;

        void list (in unsigned long how_many,
                   out bindingList bl,
                   out BindingIterator bi);

    };


    ...


};
```

Some approximate Unix equivalents

- ***bind ~= link***
- ***unbind ~= unlink***
- ***bind_new_context ~= mkdir***
- ***destroy ~= rmdir***

Uniqueness of names

- ***Objects can have more than one name***
  - **or no name at all (in which case the naming service cannot find them)**
  - **... the naming service is just one way of finding objects**
  - **... it is not the only way**
- ***Conversely, in a naming context, a name can denote only one object***
  - **simple names cannot be duplicates**

Other features of the naming service

- ***The naming service does not know about the types of objects***
- ***Names are independent of the location of name services***
- ***Existing (non-CORBA) name services can be encapsulated***

Exclusions from the Naming service

- ***No 'rename' operation***
  - **it is excluded, because it would have to rely on a transaction service to guarantee correctness**
- ***No administration of names***
  - **this is the responsibility of higher-level software**

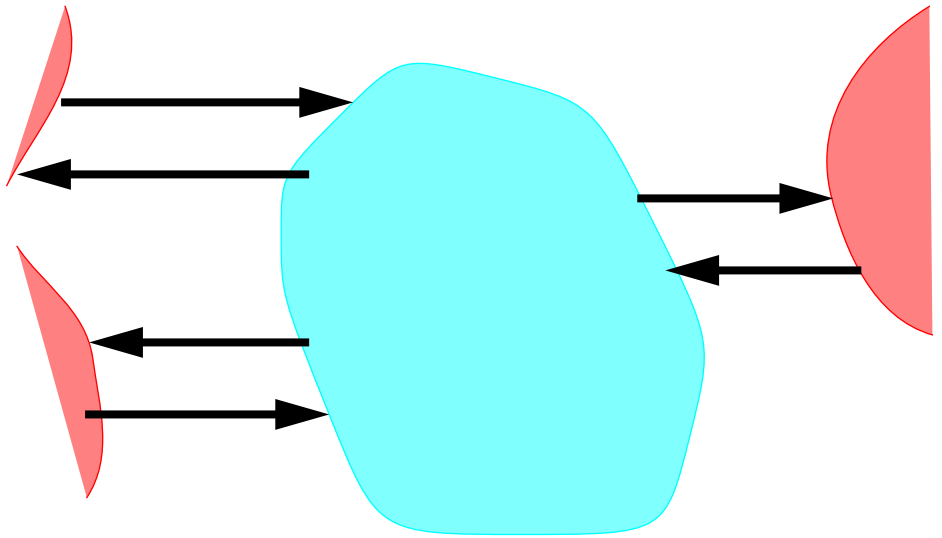
The Naming service specification

- ***One module (CosNaming), containing***
  - **interface BindingIterator**
  - **interface NamingContext**
- ***A names library, containing***
  - **interface LNameComponent**
  - **interface LName**
- ***The names library is specified in pseudo-IDL***
  - **allowing names to be handled as pseudo-objects**
  - **... pseudo-objects cannot be passed across IDL interfaces**
- ***You can use CosNaming without the names library***

Anticipated changes to the Naming service

- ***Other forthcoming Object Services will have an impact on the Naming service***
  - **Security (in COSS 3)**
  - **Transactions (in COSS 2)**
  - **Change Management/Versioning**
  - **Internationalization**
- ***The Naming interfaces will need to evolve accordingly***

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**

Conspiracies - a ‘new’ object concept

- ***Recall that in CORBA, an object can only have one interface***

- ***But a service can have multiple interfaces***

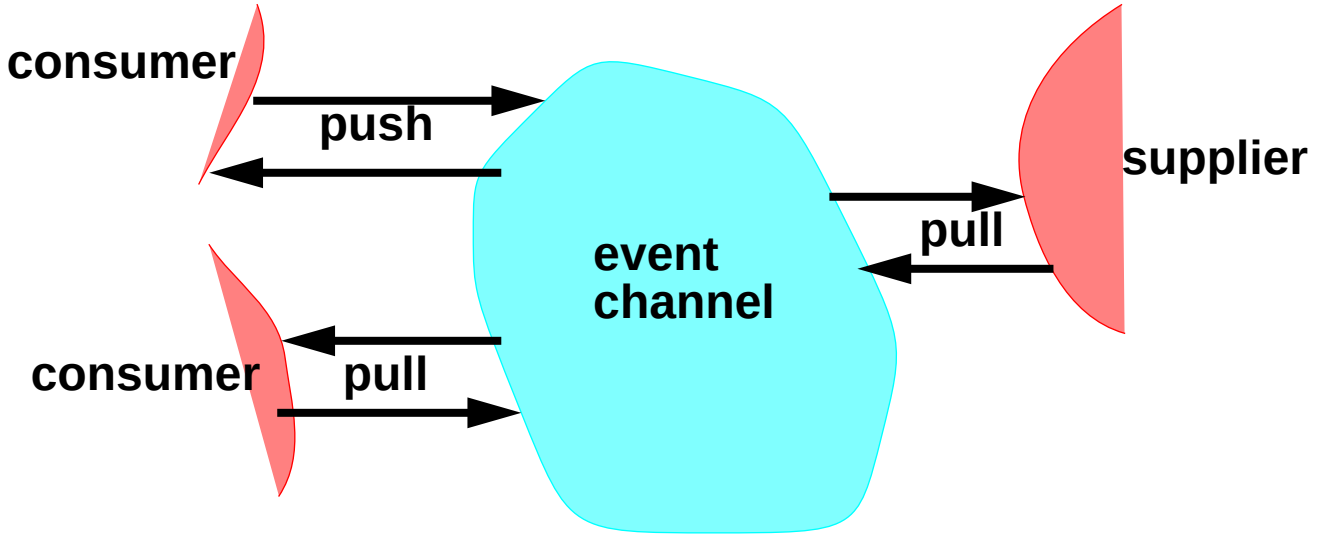
  - **provided by a set of ‘conspiring’ objects**

Push and pull

- ***Two models of communicating event data***
  - **push model: suppliers push data to consumers**
  - **pull model: consumers pull data from suppliers**

Push and pull with event channels

- ***Suppliers and consumers can use the same or opposite models...***



- ***... the models are decoupled by the event channel***

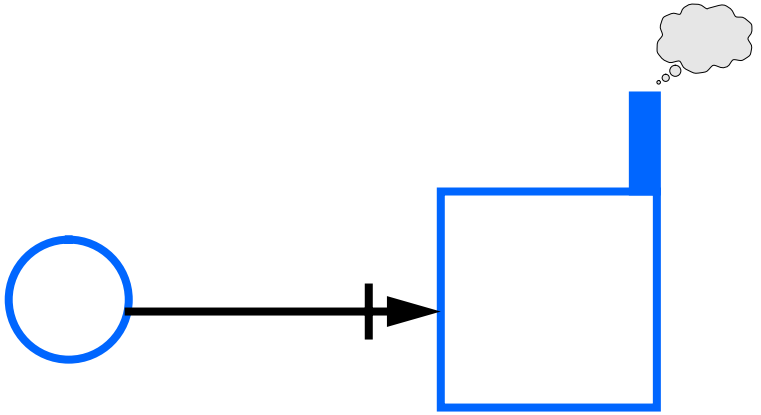
Generic and typed channels

- ***Generic event channels allow communication of arbitrary (untyped) data***
- ***Typed event channels allow communicate via an IDL interface***
  - **any IDL interface by mutual agreement between consumer and supplier**
  - **... operations in that interface are transformed into 'pull'/'try' operations, with the same parameters**
- ***Typed event channels also support generic events***
- ***Push and pull models are supported for both generic and typed event channels***

The Event specification

- ***Module CosEventComm***
- ***Module CosEventChannelAdmin***
- ***Module CosTypedEventChannel***
- ***Module CosTypedEventChannelAdmin***

The LifeCycle service

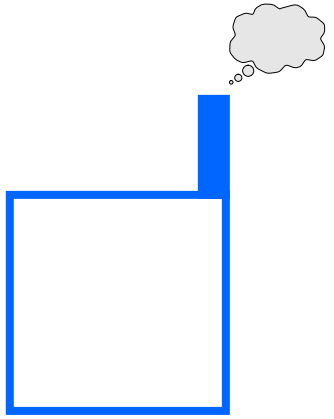


- ***Life cycle services allow applications to control***
  - **creation of objects**
  - **removal of objects**
  - **copying of objects**
  - **moving of objects**
- ***The LifeCycle service depends on the Naming service***

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



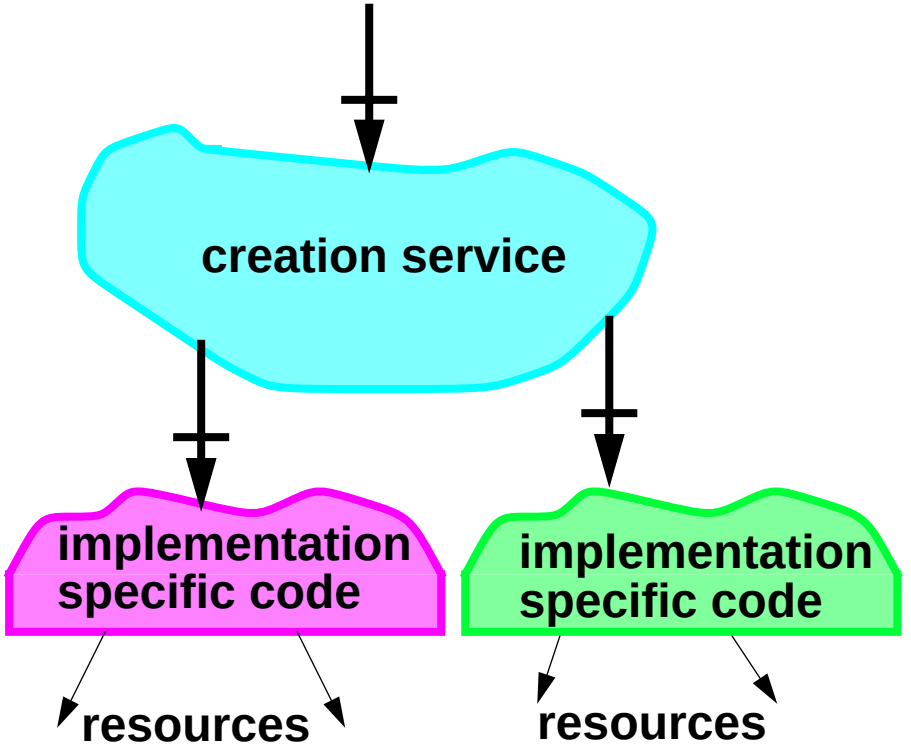
- ***A factory object is an object that can create other objects***
- ***Factories are not special***
  - **they are specified in IDL**
- ***Factories are responsible for allocating resources to objects that they create***
  - **according to client control/administrative policy**

Generic factories

- ***Although factories are not special, a standard interface for factories is impractical***
  - **there are too many possible kinds of resources**
- ***Factories are certain to be implementation-specific***
- ***Instead, a generic factory interface is defined***
  - **which invokes the implementation-specific ones**
  - **... passing ‘criteria’ (name-value pairs) through as parameters**
- ***The criteria in the LifeCycle specification are “suggestions”***

Generic and implementation-specific factory interfaces

- ***Creations ultimately invoke implementation-specific factories***



The LifeCycle service specification

- ***One module (CosLifeCycle), containing***
  - **interface FactoryFinder**
  - **interface LifeCycleObject**
  - **interface GenericFactory**

Extensions to the LifeCycle service

- ***The document includes three Appendices covering***
  - **groups of related objects**
  - **filters to specify resource constraints when objects are created**
  - **administration**
- ***The document goes to some pains to stress that these Appendices are not part of LifeCycle specification***
  - **even though the level of detail is roughly the same**

Using the Naming service

- ***Almost every application will need to use the naming service***
  - **directly or indirectly**
- ***The naming service deliberately provides a bare minimum service***
  - **you may wish to build services with higher-level operations on top of it**
  - **...searching and sorting, for instance**
  - **...similar facilities to those in X.500 or the DCE Directory Services**
  - **...possibly as Common Facilities or Application Objects**

Using the Events service

- ***Use the Events service if you need***
  - **queued messaging**
  - **notifications**

Using the LifeCycle service

- ***In its current form, the LifeCycle service is not detailed enough to be directly usable***
  - **the Appendices are not part of the specification**
- ***However, you may find the specification helpful***
  - **when designing your own services**
  - **when designing your own factories**
- ***For the moment, you may need to use vendor-specific interfaces***
  - **for object creation and deletion**

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**

Summary

- ***For a description of the service design principles***
  - **see *Common Object Services Specification, Volume 1* by the Object Management Group (Wiley)**
- ***For the specifications of the following services, also see Volume 1***
  - **Naming**
  - **Events**
  - **Life Cycle**
  - **and (in a later edition), Persistence**
- ***For future Object Services***
  - **await future publications**

Understanding the OMG specification process

- ***Requests for Proposals (RFPs) are issued by an OMG Task Force***
  - **the Object Services Task Force tends to issue them in bundles**
- ***Potential submitters write Letters of Intent (LOIs)***
- ***Submitters supply initial submissions***
- ***Submitters supply revised submissions***
- ***The Task Force votes***
- ***The OMG Technical Committee votes***
- ***The OMG Board votes***

OMG RFPs and specifications

- ***So far, 4 RFPs have been issued by the Object Services Task Force***
  - **known as COSS 1-4**
- ***COSS 1 specifications now issued as Volume 1***
  - **with Persistence as an addendum**
- ***The timetable doesn't always match the order in which RFPs were issued***

Object Services RFP 2

- ***Externalisation - object storage on (removable) media***
- ***Relationships - associations between objects***
- ***Transactions - serializable operations***
- ***Concurrency - control of concurrent operations***

Object Services RFP 3

- ***Security - authentication and authorization***
- ***Time - synchronized clocks***
- ***...security implementation depends on (trusted) Time service***
  - **to thwart replay attacks**

Object Services RFP 4

- ***Licensing - support for controlling and charging for service usage***
- ***Properties - associating named data with an object***
- ***Query - query language used to select objects from collections***

CORBA Trading service?

- ***A trading service is being talked about***
  - **no RFP has yet been issued**
- ***Trading is mentioned in several places in Volume 1***
  - **particularly when federation is discussed**
  - **particularly when life cycle issues are discussed**