**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 - Designing Applications with CORBA [CNET]

**Chris Mayers**

**Abstract**

Organizations adopting CORBA technology need to develop an insight into how applications are designed.

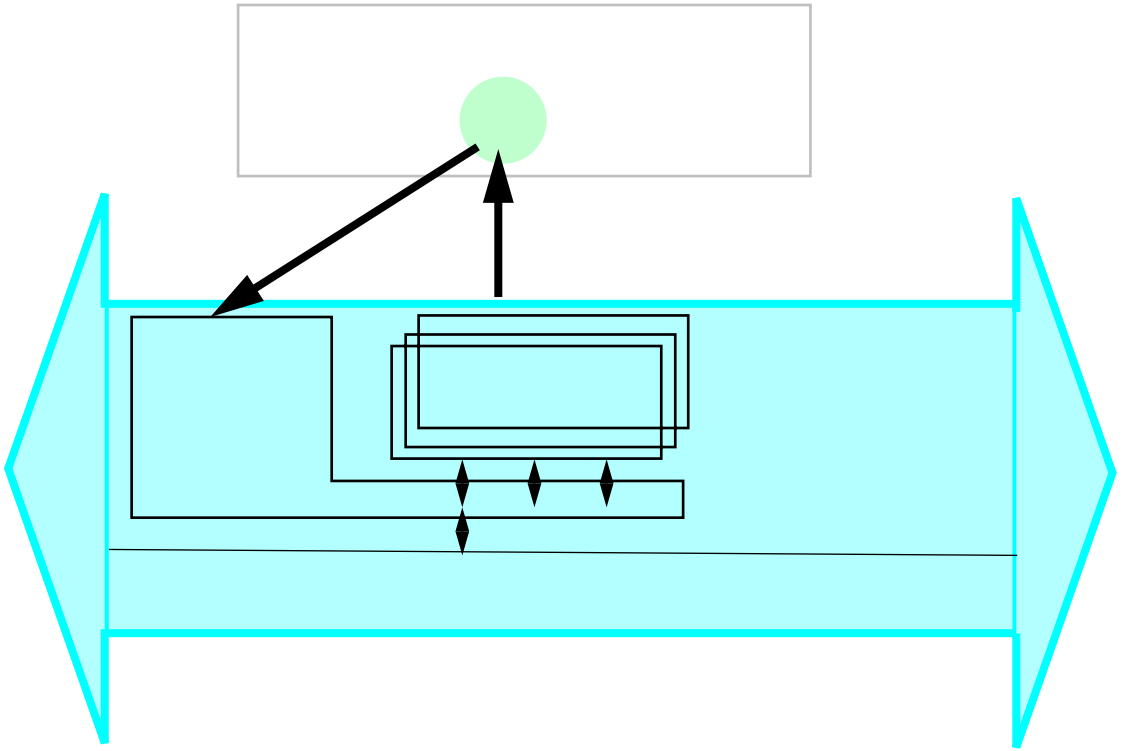
This module of the ANSAwise training programming describes the ORB interfaces (concentrating on the Interface Repository), and also discusses activation policies for servers.

[This is a variant of APM.1352, with the paper exercise removed. It was developed for CNET.]

APM.1630.01 **Approved** 20th November 1995
Briefing Note

**Distribution:**
**Supersedes**:
**Superseded by**:

# Designing Applications with CORBA




# In this session

- *Explain some of the standard CORBA interfaces*
  - including the Interface Repository and Implementation Repository
- *Explain how to design CORBA object implementations*


# More CORBA standard interfaces

- *Interface Repository Interface*
- *Implementation Repository Interface*
- *Dynamic Invocation Interface (DII)*


# CORBA Interface Repository

- ***Holds information about the specifications of objects***
  - **their interface definitions**
- ***Effectively, a compiled form of the IDL***
- ***The Interface Repository Interface allows the interface repository to keep in step with the object implementations that are available***


# CORBA Implementation Repository

- ***Holds information about implementations of objects***

- ***The information is ORB-dependent***

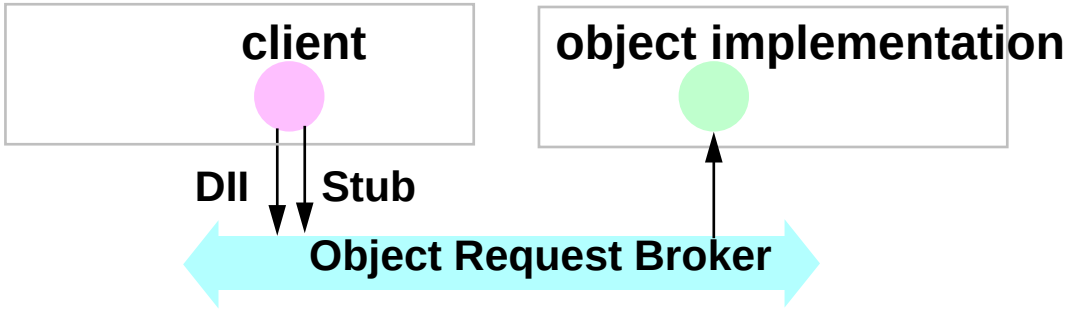
  - **applications using the Implementation Repository will not be portable between ORBs**




# Two forms of request

- *Clients can make requests via IDL Stubs, or via the Dynamic Invocation Interface (DII)*




- *There is one IDL Stub per interface*
  - but only one DII, which supports all interfaces
- *Very roughly speaking*
  - requests via IDL stubs are ‘compiled’
  - requests via the DII are ‘interpreted’

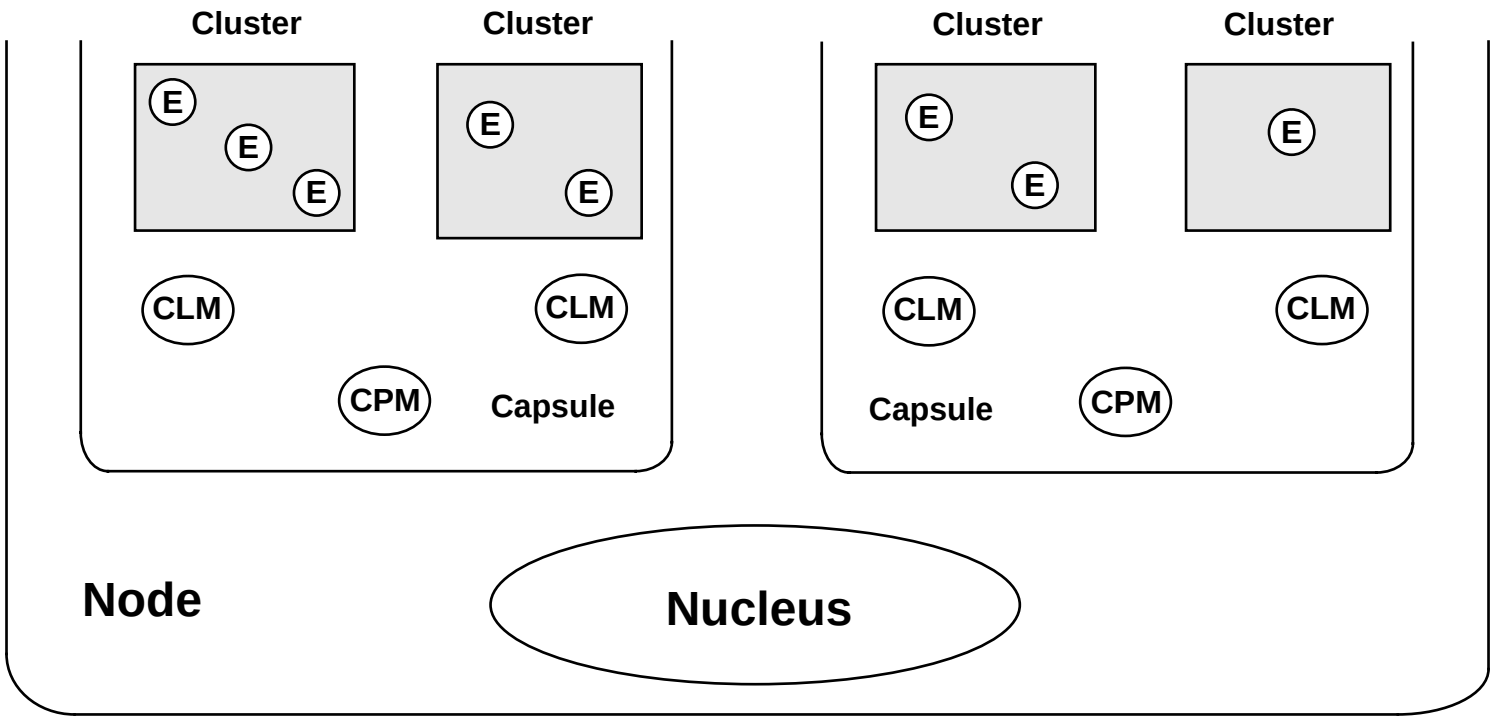

# IDL Stubs or DII?

- *The interfaces to IDL Stubs and the DII are the same for all ORBS*
  - clients are portable, whether they use IDL Stubs or the DII
- *But the interfaces to IDL Stubs and to the DII are very different in form*
  - you need to write different source code for the two cases
- *Clients can make requests via IDL Stubs for some interfaces, and the DII for others*
- *Servers support both IDL Stubs and the DII*
  - they are not aware of the form of the client request


# When you should use the DII

- *Application programmers must construct DII parameter lists by hand*
- *The DII API is large and complex*
- *There are no checks until run-time*
- *... You should therefore only use the DII for writing ORB tools*
  - for example, an object browser

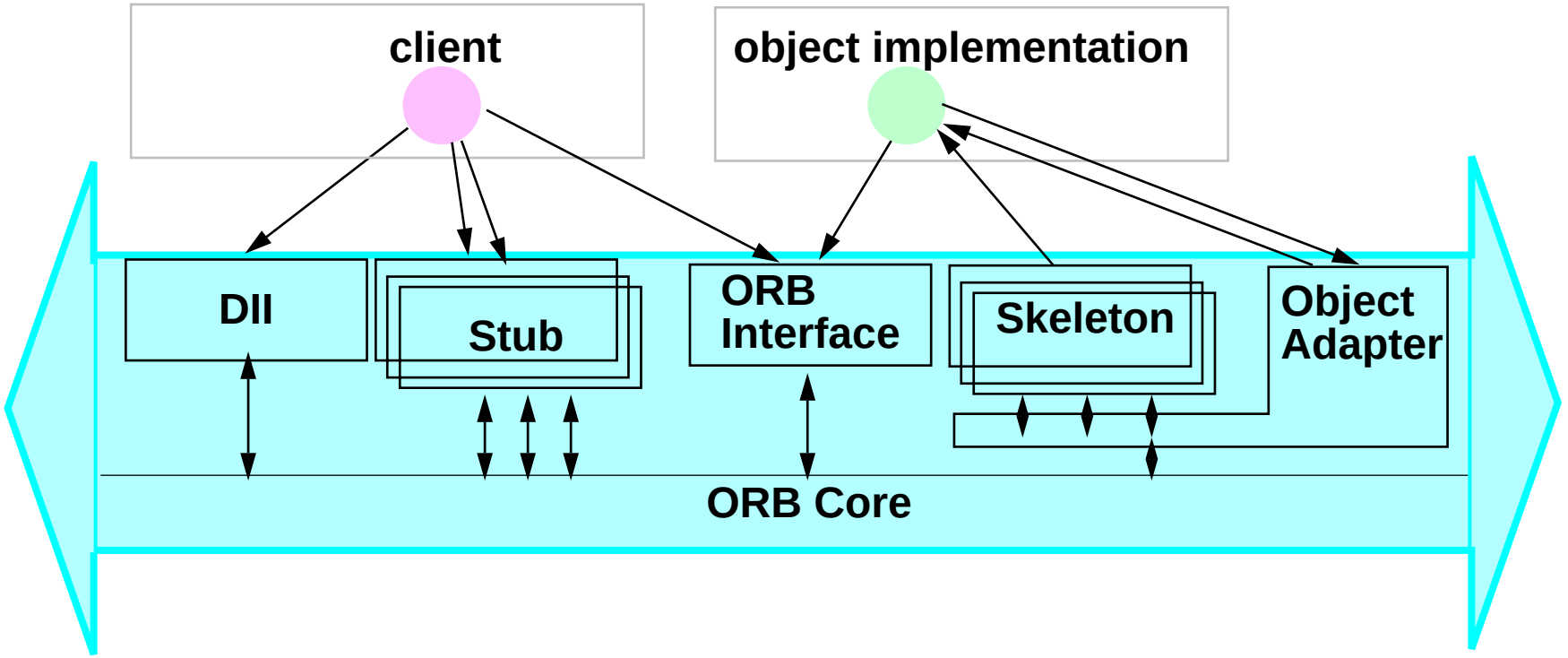

# ODP Engineering Encapsulation Infrastructure




# The ODP Engineering Model and CORBA

- *The ODP Engineering viewpoint defines*
  - *capsules* for resource allocation and protection
  - *clusters* for collective activation, deactivation, and migration
  - *nodes* for network addressing
- *CORBA does not have an equivalent engineering model*
  - the mapping of application programs onto capsules, clusters, and nodes depends on the ORB...
  - ... and on the activation policy for the applications

# Structure of the ORB




# Basic Object Adapter

- ***Calls flow in both directions between the Basic Object Adapter and the implementation***
  - **(down) calls from the object implementation to the BOA**
  - **upcalls from the BOA to the object implementation**
- ***Calls from object implementation use the BOA interface***
  - **an IDL definition, in the usual way**
- ***Upcalls from the BOA use the language mapping***
- ***Object implementations must be involved in both calls and upcalls***
  - **to implement the operations of their interfaces**
  - **during activation of implementation and objects**


# Activation policies

- *CORBA allows a variety of activation policies*
  - important when servers support multiple objects
- *The Basic Object Adapter supports four policies*
  - shared server: multiple active objects sharing the same implementation
  - unshared server: one object per implementation
  - server-per-method: each invocation activates a new server - the server terminates when the invocation's method completes
  - persistent server: the server is activated by some external mechanism - it must be available before invocations can be processed


# Limitations of the current Basic Object Adapter

- ***The current CORBA specification does not completely specify the Basic Object Adapter***
  - **how activating objects are mapped to processes**
  - **how implementations are registered**
  - **how implementations wait for incoming requests**
  - **how groups of objects (ODP clusters) are activated and deactivated**
  - **how multithreading can be achieved**
- ***The ORB Portability Enhancement RFP seeks to resolve these issues***
  - **possibly by specifying a range of Object Adapters, or an Object Adapter Framework**


# CORBA design and the Engineering model

- *It is straightforward to relate CORBA to the Computational model*
  - CORBA IDL shows this well
- *It is somewhat harder to relate CORBA to the Engineering model*
  - a CORBA server may or may not be an Engineering model capsule, for instance
- *This will become clearer as CORBA technology matures*


# Summary

- ***Clients and object implementations can use the ORB standard interfaces as necessary***
- ***Object implementations must use the BOA interface***
- ***For more on this topic***
  - **on the CORBA Interface Repository and Implementation Repository, see *CORBA* (OMG)**
  - **on the Basic Object Adapter, also see this document**
  - **on the ORB Portability Enhancements, see OMG document 95-6-2**