



Taming the beast with CORBA

Maintaining legacy applications presents some interesting challenges. An application that has evolved over many years may still be a critical part of an essential business process. The source may have become large and unwieldy and have outgrown its original design. Sometimes the only thing you can rely on is that it is well tested and works. Such applications are difficult to maintain and even the smallest of changes require extreme care and rigorous testing. With each new release comes the risk of introducing unwanted bugs into an otherwise stable product. Alternatively the legacy application may just be hampered by outdated interface protocols that are no longer actively supported. Either way, a major update might entail a costly development exercise at best - replacing several layers of code or at worse - a complete clean-sheet approach.

But before firing up the CASE tool, what about all those man years of effort? Can you re-package the software without resorting to major changes to the underlying code? Maybe the solution is closer than you think ...

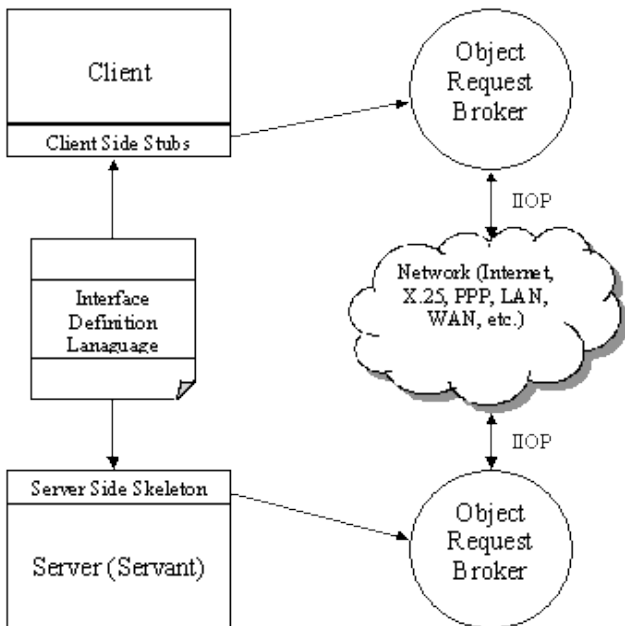
Before discussing how MPC Data can tame your beast with CORBA, let us first take a look at what it is.

Common Object Request Broker Architecture is a mechanism for developing distributed applications across heterogeneous networks. Put simply: it allows a client to transparently access remote objects and perform operations on them. It is the system designer's duty to define what the objects actually represent. It could be as simple as a light switch (the object) that can be switched on & off (the operations) or as complex as a complete Point of Sale terminal that handles transactions. A whole system would typically consist of multiple objects that the client would interact with to perform a given task. An object's interface - its name and operations - is defined in Interface Definition Language (IDL).

At the heart of CORBA is the Object Request Broker or ORB. As its name implies it simply forwards operations on objects to the desired object and returns results to the client. The forwarding operation may be from the client through an ORB directly to an object or from an ORB to a separate ORB and then on to the object. These inter-ORB communications permit a client to communicate with an object hosted on a remote system. CORBA uses its own standard protocol IIOP (Internet Inter-ORB protocol) to communicate between ORBs.

ORBs are available from many commercial vendors such as Orbix & Borland's Visi-broker. There are also many freely available ORBs such as Omni-ORB (<http://omniorb.sourceforge.net/>), ORBit (<http://orbit-resource.sourceforge.net/>) and ACE's TAOS (<http://www.cs.wustl.edu/~schmidt/TAO.html>). The choice of ORB is largely a matter of cost, support and the provision of extra value added services to the base product. MPC Data has used several of the freely available ORBs mainly because they are distributed in source form allowing our engineers to see (and debug!) the inner workings of the ORB.

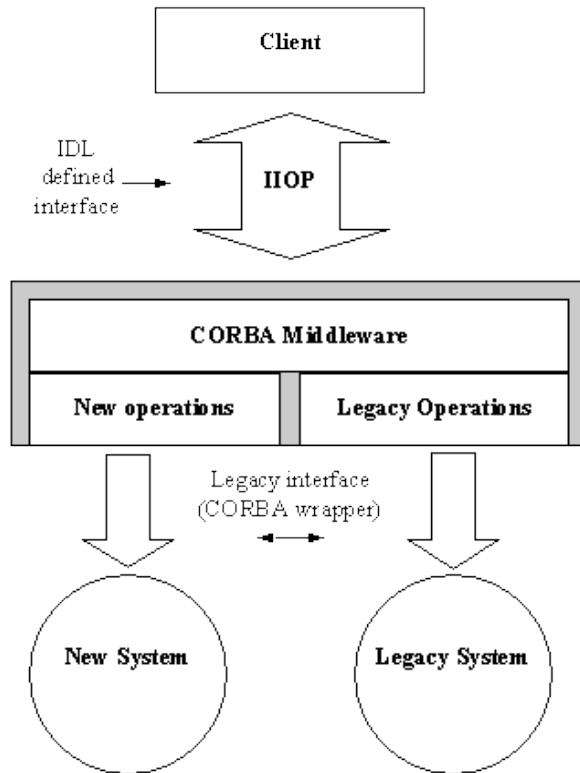
The IDL is compiled to generate code for both the client and the server. The generated code for clients is in the form of object stubs. From the client's perspective these stubs are function calls directly into the object. In actuality the stubs forward the request to the remote object via the ORB. The compiler also generates skeleton code for the server (typically referred to as a servant in CORBA), which needs to be fleshed-out with the implementation of the requested operations. IDL compilers are available to target a vast range of programming languages including C/C++, Java and Python and indeed the client and server may be implemented in different languages.



Wrapping your application in CORBA middleware

CORBA is a robust international standard supported by a sizeable development community.

The key advantage is that CORBA provides a level of abstraction between legacy application and client that permits implementation of new functionality without altering (and thus potentially breaking) a stable product. The CORBA middleware acts as a mediator passing existing operations onto your application, and forwarding new operations onto separate clean-sheet applications. You can potentially freeze your legacy code at the last stable release. Your man-years of effort are now safe!



Defining the IDL

It is vitally important that the IDL is defined in such a way as to provide a complete abstraction of the legacy system. There are many caveats to producing a good IDL that encapsulates the functionality of the legacy application without restricting its future growth. As the application is enhanced new IDL files can be issued adding new interfaces and operations. Allowing more than one IDL to define the interface to the application can control migration between versions. This is critical for customers requiring side-by-side operation of old and new systems during handover. MPC Data is highly proficient at IDL design and has experience in maintaining backward compatibility, supporting multiple releases and ensuring migration pains are minimised.

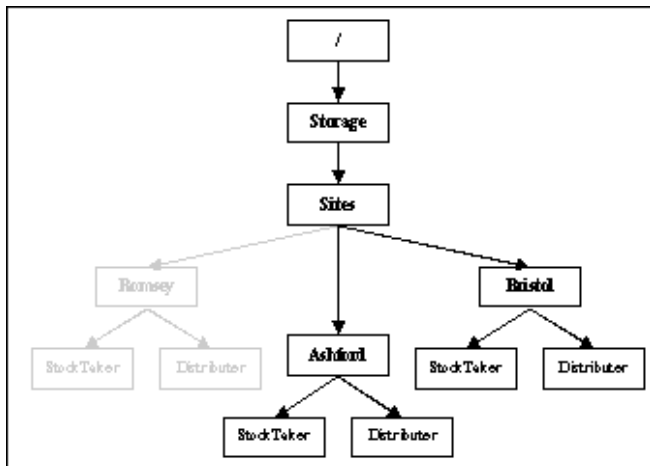
Simple example of IDL file:

```
interface light_switch
{
void flick_switch(in boolean flick_on);
boolean is_on(void);
};
interface room
{
typedef sequence<light_switch> light_switch_list;
light_switch_list get_lights(void);
void add_light_switch(in light_switch s);
void remove_light_switch(in light_switch s);
};
```

Naming Service

The Naming Service is a core feature of any CORBA distribution that provides a mechanism to publish active objects such that they can be located and used by clients. Access to the name services itself is defined in IDL, and provides a hierarchical tree of objects not unlike a directory tree in a file system. Location of services within the legacy application can be simplified by publishing the objects representing them in the naming service. This also increases the level of integration of the systems within the product. Groups of legacy systems at a given site could all be stored within branches that represent the site. An object may also publish itself in more than one place allowing other views of the system to be represented. A key use of the naming service is to provide redundancy. More than one legacy system may be able to service a given request if they have the same IDL definition. By publishing both systems into the name service when one system is unavailable then another object representing another system may be able to service the request instead. MPC Data has successfully developed a simple

solution that permits redundancy capabilities in a simple yet effective way.



Security

The IIOB protocol that is transmitted between ORBs is not encrypted. CORBA expects security to be dealt with at a higher level. The commercially available ORBs provide additional mechanisms to deal with this. Simple solutions to this problem include sending the IIOB protocol through SSH, or ensuring that the IIOB protocol is only sent over known safe paths.

MPC Data Experience

MPC has in-depth experience in supporting legacy applications and has been producing middleware solutions to solve customer problems since its inception. We have just completed a major CORBA implementation for a large telecoms application in Solaris 8 using Omni-ORB 3.0.5 which includes its own naming service. The initial project delivered 3 IDLs for 2 objects with around 60 non-trivial operations. The work consisted of consulting and design, providing a framework for implementing a common method of operation for all the CORBA objects within the system, implementing the 60 operations and providing a session control mechanism. The framework provides the customer with the ability to easily add in extra objects and provide backward compatibility with older releases.

If you would like to discuss how your own application might be tamed with CORBA contact rjones@mpc-data.co.uk

-- Richard Jones