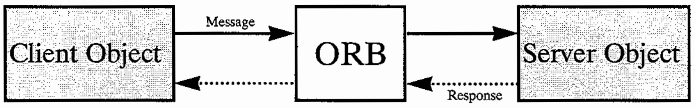
What CORBA will Mean to the M Community

by Erik Zoltán

What is CORBA?

CORBA is a standard object-oriented architecture that permits communication between many different kinds of applications in a distributed computing environment. CORBA is an emerging data interchange standard that has been embraced by the Department of Defense (DoD) and has gained ever-widening acceptance within the computer industry.

The primary benefit of CORBA is a high degree of interoperability. ***Interoperability*** refers to the ability of different applications to work together. There's not much difference between the words "interoperate" and "cooperate." CORBA is a complex standard comprised of many technical details. But ***without getting technical***, the following diagram illustrates the essence of CORBA.



An Object Request Broker (ORB) distributes messages between application objects in a highly interoperable manner. The Common Object Request Broker Architecture (CORBA) specifies a standard ORB for communication between any two objects in a distributed computing environment. The basic idea is simply that one object sends a message to another and receives a response back via the ORB. For the purposes of that particular message, the sending object is called the "client," and the receiving object is called the "server." But most objects function as both servers and clients of the ORB.

CORBA was designed to be independent of any specific programming language, operating system, hardware platform, or network configuration. An object's interface is defined with an object-oriented Interface Definition Language (IDL) that can be used in conjunction with almost any programming language. As a result, CORBA is *language-independent*, permitting seamless interactions between objects created with different programming languages.

CORBA is very flexible. There are two basic ways for objects to communicate with each other: *statically* (in which the objects know about each other's capabilities beforehand, communicating in pre-arranged ways) and *dynamically* (in which the objects determine each other's capabilities at runtime, using this information to determine how they can best communicate). The static method is like talking to an old friend, while the dynamic method is like meeting someone new on the street, discovering they are from Paris and pulling out a French phrase book. Both methods may be necessary in a distributed environment where a diverse blend of objects can be encountered.

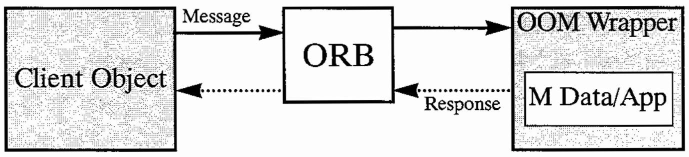
Will CORBA Work with M?

No existing M system currently offers CORBA support. ESI Technology Corp. provides an object-oriented M (OOM) system called EsiObjects™ which already supports Windows Object Linking and Embedding (OLE), a competing OO standard. The DoD has awarded a six-month contract to add CORBA interoperability to EsiObjects, opening up existing M databases to the next generation of data interchange standards. This project is already under way, so CORBA will be available very soon.

But merely supporting CORBA is not enough. Existing M applications and databases are not object-oriented, so it is inherently difficult to get them to interoperate smoothly in an object-oriented environment. (The same issue comes up when interfacing M with Windows OLE.) It's easy for objects to communicate with each other, because they share a common set of design principles. M applications have very different design principles, making interoperation trickier.

Technical observation: an object-oriented application may use thousands of different objects to represent the information normally contained in a single M global database file—each entry in the database is an object that might contain its own component-objects, and the database object itself requires a number of different component-objects to store and index these entries. An OO application working with a database naturally expects to communicate with many different objects, each having a clearly-defined role with no danger of overlap. But ***M systems just aren't designed that way.*** In M, a single global can contain one or more entire databases, and *any* routine can modify *any* global node.

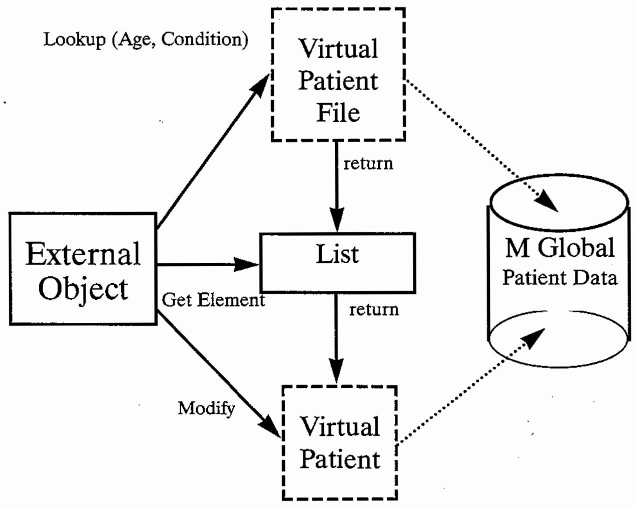
Simply put, the problem is that M brings everything together into an integrated whole, while OO systems divide everything up into discrete, modular components. (This is not a criticism of either approach—the important point is merely that the two approaches are irreconcilably *different.*)



In an OOM system the M database/application can easily be given an object representation (known as a **wrapper**). In this way, it can act like an object to the outside world, making any unit of M data indirectly capable of responding to (and initiating) communications via the ORB.

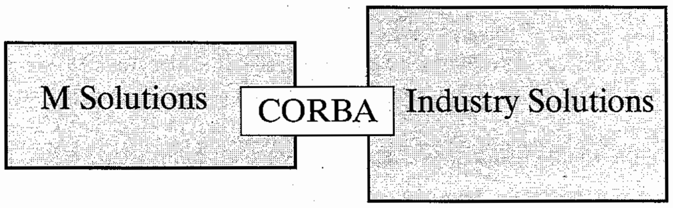
OOM programmers use **virtual objects** to create these database wrappers. Each virtual object is associated with a specific global location (a single file, a single record, a single field within a record, etc.) and provides an object-oriented representation of the information and functionality associated with that item. Since they are extremely lightweight, virtual objects can be generated "on the fly," as they are needed, with very little overhead.

To illustrate this, let's very briefly see how an external program might communicate with a typical M Patient File global using CORBA and virtual objects. In the following example, all the communications would occur via the ORB, which has been omitted in order to simplify the diagram.



Let's suppose that medical application needs to update the status of the patient records for those patients over 65 who have Alzheimer's. An external object might communicate with a virtual **PatientFile** (representing a patient global), requesting those patients over 65 with Alzheimer's. The virtual **PatientFile** object would then send back a List object. The external object would interact with the **List** to get at the virtual **Patient** objects inside. Modifying a virtual **Patient** object would result in changes to the global location(s) it represents.

The bottom line is this: any system that wishes to provide a general solution to the important task of interfacing M with object-oriented standards such as OLE and CORBA must address two critical issues: the first is supporting the standard, and the second is making M routines and globals work like objects in the external system.



CORBA will provide a conduit between the M world and a much wider arena of emerging industry solutions. This will allow M systems to benefit from many new resources, while also making M solutions available to a broader audience. The power and flexibility of M will make it very competitive in this arena, if it can be "tamed" and made to conform with the standard object-oriented expectations and requirements. And—as shown above—OOM appears to be the most natural way to accomplish this. M

Erik Zoltán has been programming, writing, and teaching in the M community for the last 7 years. He is now also teaching EsiObjects™ programming classes and is involved with the DoD contract for CORBA connectivity to EsiObjects™ for ESI Technology Corporation in Natick, MA.
