

flexibility using distributed object computing

Paul Augustin is NOLA's vice president of operations. He is currently assigned as the contractor assistant general manager for systems architecture and engineering at the SPAWAR Information Technology Center in New Orleans. In this role, he is responsible for the direction of requirements analysis, architecture, design, and testing for Navy personnel systems as well as technical support that the SPAWAR ITC is providing to other federal agencies.

are you running systems on a mixture of mainframes, servers, and midrange platforms? Do your systems operate on various versions of UNIX, Novell NetWare and Windows NT, with workstations running assorted flavors of Microsoft Windows, IBM OS/2, or Apple Macintosh? Perhaps your systems have communication switches or real-time process control devices. Maybe your applications use PowerBuilder, COBOL, or Java clients to talk to Sybase, Oracle, DB2, or IMS databases. Do you have trouble making all of these systems communicate? If so, you're not the only one.

Over time, as development and purchases accumulate, legacy systems become too crucial or costly to replace. If, for twenty years, you've been successfully running a critical customer information application on a mainframe, for instance, you're not likely to scrap the system in favor of the latest technology. Or, if you've invested a large sum of money on a system, you probably want to keep it running until the investment has paid off.

In an ideal environment, heterogeneity lets us build an enterprise using the best combination of hardware and software

components. But, achieving heterogeneity isn't always easy. Although programming packages and interfaces are available for development on a single homogeneous platform, few facilitate integration of separate systems into a distributed heterogeneous environment. Traditional integration involves low-level network and operating system programming. The resulting environment is hard to maintain and harder to extend.

Distributed Object Computing views applications and services as *objects*—think of “black boxes”—that send and receive messages. Client applications don't need to know where the objects are or how they implement, because interface coding provides a transparent bridge. With DOC, software maintenance is easier. And, because new objects can be built from existing objects, development cycles are shorter, and modifications are less expensive.

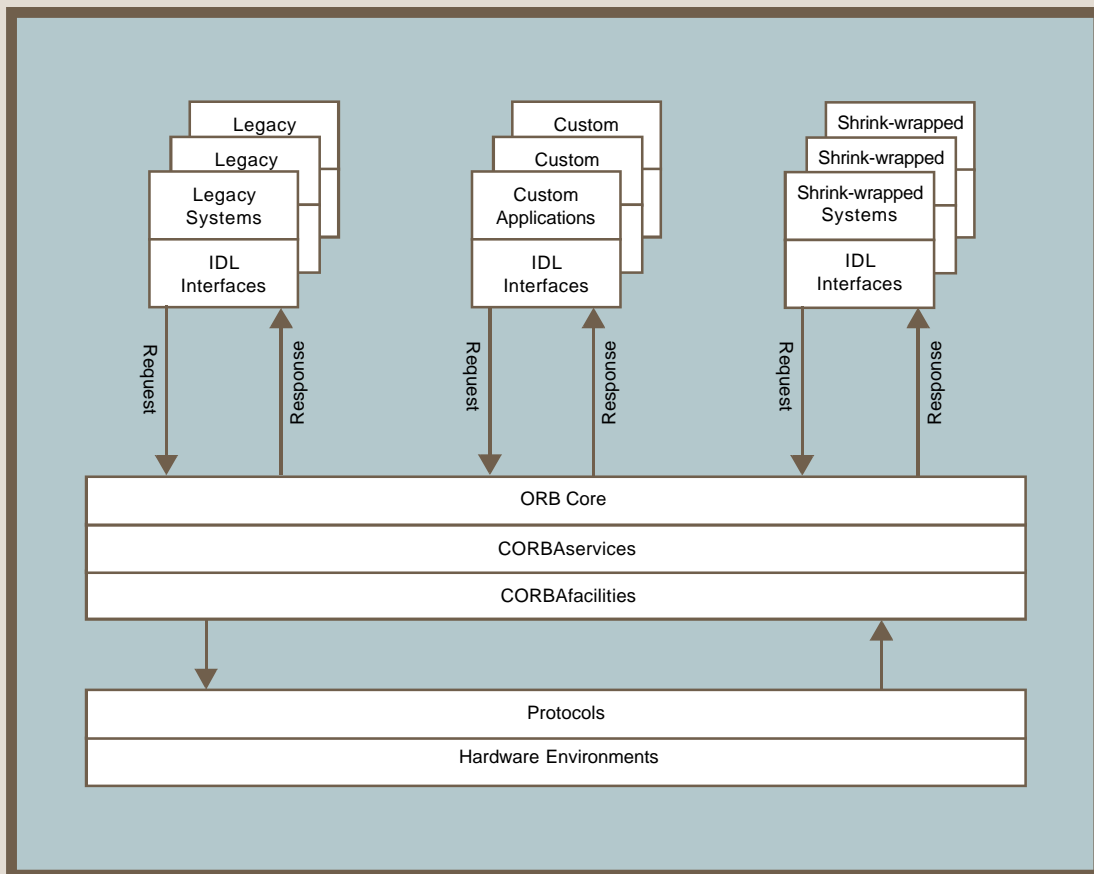
OMG Standards

The Object Management Group has been driving the DOC movement since 1989. With over 750 members, OMG is the largest software consortium in the world. It's first publication, “Object Management

Architecture,” guides development and deployment of interoperable distributed object systems within heterogeneous environments.

Common Object Request Broker Architecture, an OMA offshoot, specifies component characteristics for distributed object computing frameworks. CORBA is available for desktop (Windows, UNIX, Macintosh) and enterprise systems (OS/390, OS/400, UNIX, VAX/VMS, NT). CORBA's Object Request Brokers let diverse systems operate together as if they were one system. A key function of Brokers is letting applications use objects without knowing their network locations or protocols, whether the objects reside inside the application, inside a separate application on the same platform, or inside a separate application on a separate platform. Brokers also handle implementation details independently, enabling the development of modular—and reusable—software.

CORBA software components written in different languages are able to interact via CORBA's neutral Interface Definition Language. IDL can map to a number of different languages, and lets each developer use the language of choice. IDL also facilitates distribution for senders and



receivers on different network machines. Custom applications, legacy systems, and shrink-wrapped software from such vendors as SAP, Oracle, PeopleSoft fit easily into a CORBA environment.

CORBAservices addresses fundamental requirements common to all heterogeneous and distributed computing architectures, including Security and Transactions. CORBAfacilities are layered on top of CORBAservices, and include e-mail, printing, and compound documents. In addition, OMG teams are working on such specialized common facilities as system management, geo-spatial data

processing, and standards that enhance interoperability between software products of different vendors.

Component-based distributed systems have great flexibility. Client applications may be added to the system at any time. Object implementation can be changed at any time and, as long as the interface to the object doesn't change, client applications don't need to be rebuilt. So don't despair. No matter what you're running or what you're running it on, there's a good chance you can keep it running with DOC.