Payoff

The need for enterprise-level client/server applications to communicate with one another has engendered both the challenge of interoperability and its potential solution—middleware. This article discusses the business and technical issues associated with enterprise computing and how middleware is being used to resolve them.

Problems Addressed

The increasingly distributed nature of business locations and operations has led to a concomitant expansion of client/server computing from the department level to the enterprise level. Yet the successful implementation of client/server, or distributed, business applications depends on interoperability—the ability of applications to work together across a network to perform business functions. Systems integrators need to know exactly how a client application will talk with a server application before either application can be designed or written. If they do not, unrealistic assumptions about applications-level connectivity can be project killers.

Because enterprises typically have many and diverse systems to meet their business needs, interoperability problems are almost always encountered as soon as applications on desktops, mainframes, midrange systems, and servers need to interact with each other. No products have emerged as clear-cut, widely supported standards, de facto or otherwise, for distributed enterprise applications. Systems integrators are tested to the utmost by the fact that these client/server applications must be developed with wide-ranging assortments of platforms, networks, data bases, and tools.

The need for applications to be able to directly exchange information in real-time in a distributed heterogeneous environment has led to the development of middleware—software that bridges the gap between business applications and systems-level services such as data bases, network protocols, and operating systems. This article discusses the business issues in enterprise computing and the myriad interoperability problems associated with achieving distributed business applications. It then reviews how middleware is being used to solve these problems.

Distributed Applications Defined

A distributed application is an automated business activity broken down into multiple processing functions running on different computers and performed in a coordinated fashion by sending information across a network. Depending on the relationship of the components, such applications are also referred to as client/server or peer-to-peer applications. Because the application components must work together across the network, the applications are more generally referred to as cooperative processing applications.

The so-called two-tier client/server model divides the processing into a client portion, which interfaces with the user, and a server portion, which interfaces with the data base. Execution of business rules is divided between the client or server components.

The three-tier model divides the work into presentation on the client platform, business rules on one or more application platforms, and data base access on one or more data base platforms. This model attempts (at least conceptually) to isolate the business rules to the
middle tier so that client applications and data base server applications are less affected by the frequent changes that occur in business rules. It is believed that this approach can lead to applications that are easier to maintain and that scale better as the volume of business transactions grows.

**Business Issues In Enterprise Computing**

**IT Infrastructure, Legacy Systems, and Changing Technology**

Organizations invest in the IT infrastructure necessary for conducting business. This IT infrastructure comprises computers, software, and networks. An organization that has been in business for any period of time is likely to have legacy systems (i.e., hardware, software, and networks) that may not be easily replaced as newer capabilities become available. Such an organization is also likely to replace its computers or networks to increase speed and capacity, for example.

Because technology, especially information technology, changes continuously, organizations face at least two major challenges to their ability to manage their investments in IT infrastructure:

- Ensuring that business-critical applications can easily adapt and remain in operation when computers, operating systems, and networks are changed or replaced for reasons of capacity, price/performance, or functional fit. This is very much an issue of managing assets and operating costs.

- Choosing infrastructure components that allow for the quick use of new technologies. This relates particularly to applications software, because the cost efficiencies of operating the business are often directly related to the applications. New technologies often have steep learning curves and existing applications may not be easily adaptable. Although this is an issue of managing assets and costs, it is also one of business adaptability and responsiveness. It is especially important for a rapidly growing business.

**Integration in a Distributed Business World**

The challenges of managing an IT infrastructure are complicated further by the increasingly distributed nature of business organization and operations. Employees and business functions that were centralized in a single headquarters ten years ago are now likely to be scattered in dozens or hundreds of locations around the globe. Departments that previously consisted of employees performing the same or similar functions are now likely to be organized as distributed teams with team members in many different locations. This increasing physical distribution of people, functions, and supporting systems engenders at least three major challenges:

**Integrating the business (internally) in a distributed environment.**

This issue actually breaks down into a series of related issues, such as:

- Ensuring that employees located in remote or branch offices have the information they need to do their work.
- Ensuring that employees across different locations can communicate effectively and work together as teams.

- Ensuring that employees across all locations understand critical objectives, are working together toward common goals, and receive the information feedback they need to evaluate and fine-tune their work. This is a huge problem and the reason why the concept of enterprisewide information systems is becoming more important.

**Integrating externally with other entities.**

Companies that previously carried out business transactions with suppliers or customers primarily by phone or mail are now interacting through such electronic communications methods as Electronic Data Interchange, electronic mail (E-mail), and the World Wide Web. The question here is how to establish effective communication yet ensure that information is not shared inappropriately.

**Providing a consistent—if not common or at least unified—supporting infrastructure.**

Such an infrastructure comprises voice communications, fax, networked computers, and information access and exchange across all locations.

**Interoperability: the Technical Challenge**

Because organizations depend on the enabling tools of information technology, their business objectives for enterprise computing are accompanied by a host of technical issues. Interoperability, however, is most often the stumbling block to mission-critical client/server systems.

Large-scale client/server applications involve complex networks, usually with many local area networks (LANs) interconnected through a Wide Area Network. More often, such applications involve multiple wide area network (WAN) and multiple network protocols, such as IBM’s System Network Architecture (SNA), NetBIOS, Transmission Control Protocol/Internet Protocol (TCP/IP), and frame relay. They typically involve several different computing platforms, or different types of computers running different operating systems, such as PCs running Microsoft Corp.‘s Windows; servers running Hewlett-Packard’s HP-UX; and mainframes running IBM Corp.’s MVS and Canadian Independent Computing Services Association. They often involve multiple data bases, perhaps based on different data base management system (DBMS) platforms, such as Oracle Corp.’s ORACLE and IBM’s DB/2 and Internet Multicasting Service. And, they will certainly involve business applications on distributed platforms tied together in a number of different ways, such as by transaction monitors, message-oriented middleware, data access middleware, and Remote Procedure Call (RPCs), or sometimes by clumsier mechanisms like file transfers or sequential batch processing jobs.

Systems integration at the enterprise level entails getting many different information systems components to work together across the enterprise network. Because these myriad components must interoperate effectively, interoperability is the first key to success. But, interoperability is not simply a problem of network protocol compatibility—it exists at many different levels, such as:

- Network interoperability.
Network Interoperability

Today, many companies running very large networks use multiple network protocols. If they are or were large IBM shops, they typically have 3270 terminal protocol, plus one or more SNA protocols on their wide area network (WAN), NetBIOS on their LANs, TCP/IP on their UNIX-based engineering networks, and perhaps even some Novell Internetwork Packet eXchange. Multiple Network Operating System may be a management issue, but at the application-to-application level, differing protocols and spanning across networks of varying types are usually the biggest problems. For example, on an System Network Architecture Logical Unit 6.2-only network, a client application can be written to invoke the APPC application programming interface (API) to establish a session and exchange information with a server application that also uses the APPC API. However, when one application is on an System Network Architecture network and the partner application is on a TCP/IP network, a major interoperability problem arises.

Platform Interoperability

Organizations striving to implement mission-critical distributed applications face the difficult challenge of interoperability among platforms of completely different types, such as IMS on IBM mainframes and UNIX platforms. Much of what has been done to date under the client/server classification involves decision support applications. Most mission-critical functions are performed primarily with the assistance of mainframe applications; yet, getting IMS or CICS to talk to non-IBM platforms, and especially nonmainframe platforms, is proving to be difficult.

Data Base Interoperability

This category of interoperability has to do with applications accessing information in data bases located on multiple systems, in data bases on different platform types, or—the most difficult of all—in data bases of completely different types (such as ORACLE and IMS). The interoperability problem is somewhat lessened if all data bases are relational data bases using Structured Query Language, although not all structured query language (SQL) are the same. It is definitely easier if all data bases use the same DBMS product, but even then there may be difficulties between certain platforms or with certain network protocols. In any of these cases, data base interoperability is a major consideration, especially when legacy systems are involved and are expected to work with newer systems.
Object/Software Component Interoperability

The advent of object-oriented systems in which data is encapsulated in objects allows information to be exchanged between applications as objects. The exchange is handled by an Object-Request-Broker, originally defined by the Object Management Group. Object request broker (ORB) are now available from multiple software companies.

Issues are surfacing, however, with object request broker (ORB) dependence on remote procedure calls when operating across enterprise networks, and with object request broker (ORB)-to-ORB interoperability—that is, getting different object request broker (ORB) products from different vendors, usually also involving different platforms, to work together. Applications built using other types of component-based software are also becoming more commonplace—with Microsoft's VBX (Visual Basic Custom Controls) being the most frequently cited type. The major issues are how such reusable components exchange information with other components and how they can work consistently and compatibly on different platforms.

GUI/MUI Interoperability

Another issue concerns how applications using a Graphical User Interface or multimedia user interfaces can be written to work on different platforms. This is, in part, a portability problem rather than an interoperability problem.

The real interoperability problem with multimedia user interfaces applications, which are expected to proliferate in the future, is twofold. It concerns interoperation of graphical user interface (GUI) functions as part of client/server exchanges when different types of graphical user interface (GUI) are involved, such as Windows, Presentation Manager, and Motif, and how to make Full-Motion Video or interactive compound media information exchanges work across heterogeneous platforms and heterogeneous networks.

Workgroup/Workflow/E-Mail Interoperability

As groupware connectivity becomes more common, one workgroup using one groupware product will increasingly need to interoperate with other workgroups using different groupware products. This is especially true with intercompany connectivity. Workflow interoperability, therefore, is a problem of:

- Integrating different groupware, workflow, and E-mail products.
- Supporting these types of applications across heterogeneous platforms and networks.
- Integrating groupware, workflow, and E-mail applications with other types of applications.
- Resolving differences in document formats such that, wherever possible, format conversion takes place automatically under the covers.

Applications Interoperability

Distributed computing usually refers to distributing the processing among applications located on different systems. Enterprise computing extends distributed computing to a larger scale—across an enterprise network of LANs, wide area network (WAN), and
multiple kinds of platforms—but it may also go much farther by integrating applications in
different business disciplines, such as the employee skills database and corporate directory
services. In both cases, at the level where things must happen based on business events,
one application somewhere on the network must exchange data with another application
somewhere else on the network. Interoperability in terms of client/server computing
always comes down to application-to-application interoperability regardless of how many
other kinds of interoperability issues are actually involved.

Middleware Solutions

Nearly all cases of successful large-scale distributed computing applications involve the use
of middleware to solve interoperability problems. Middleware, as the name is meant to
imply, is software that sits between business applications and the systems-level services, or
so-called platforms, that are the source of compatibility problems. Software layering, from
which the middleware idea derives, is illustrated in Exhibit 1.

Layered Software Architecture

<table>
<thead>
<tr>
<th>Business Applications</th>
</tr>
</thead>
<tbody>
<tr>
<td>Middleware (Common Application Services)</td>
</tr>
<tr>
<td>System Services (i.e., Data Base, Network, Operating System)</td>
</tr>
</tbody>
</table>

Because middleware is based on layering, with a new layer of software being inserted
as a higher-level platform on which business applications will reside, it provides a degree
of encapsulation or abstraction of the lower-level services. In fact, middleware typically
introduces new APIs that are used to invoke the higher-level services. That is why it is
common for applications designers and programmers to talk in terms of the new APIs—
for example, Microsoft's Open Database Connectivity or MAPI, or IBM's DRDA—when
describing how applications will be interconnected and how one or more of the
interoperability problems will be solved.

Because of the layering effect, middleware helps insulate business applications from
changes in platforms, networks, or other systems services. IT executives can therefore
change the underlying technologies, using more effective and efficient ones, without
changing the applications. The abstraction of services through the higher-level APIs also
simplifies applications programming, enabling programmers to create or modify
applications more quickly in response to business changes. By providing the means for
linking applications together across a network, middleware provides a mechanism for
applications interoperability and information access.

There are several types of middleware, including:

- X.400, MAPI, Simple Mail Transfer Protocol.
- X.500, StreetTalk.
- ODBC, Distributed Relational Data Base Architecture, distributed DBMS.
- DCE, ONC.
- CORBA/ORB, OLE2/COM, OpenDoc.
- Gateways (such as Structured Query Language Server and OmniConnect).
- RPCs.
- Message passing and queueing.
- Transaction monitors.

Most of these types of middleware are ultimately aimed at the application-to-application connectivity problem. Some are specific to E-mail interoperability (X.400, MAPI, SMTP); some are specific to data base interoperability (ODBC, DRDA, distributed DBMS, data base gateways); some are specific to object-oriented interoperability (ORB, OLE2, OpenDoc); and some are more generalized (DCE, Remote Procedure Call, message passing and queueing).

All these types of middleware let one application exchange information with another application. The exceptions are X.500 and Streettalk, which are directory services (i.e., middleware that addresses the problem of how applications are identified and actually found in large enterprise networks).

There are other interoperability solutions as well, such as protocol converters, bridges, gateways, data format translators, and other special-purpose hardware and software, but often these work at a system or network level and do not facilitate the application-to-application dialogues that are fundamental to client/server and other forms of distributed computing. The need for the direct exchange of information between applications in a heterogeneous environment has caused middleware to come into existence and to now play a dominant role in the IT architectures of progressive enterprises.

**Message-Oriented Middleware**

One particular type of middleware—message-oriented middleware—allows an application to send messages (i.e., data) to other applications and to receive messages in return. It encompasses message passing, message queueing, and transaction monitors. Messages in this context are any type of transaction or other exchange that might occur between distributed applications. The meaning and the format of the messages are defined by the structure and contents of the data to meet the requirements of each particular distributed application.

One example of commercial message-oriented middleware, and probably the earliest to be used in a mission-critical production environment, is the Communications Integrator (CI) of Covia Technologies (EnglewoodCentral Office). The Communications Integrator, first used in the computerized airline reservations system industry, was initially developed by United Airlines for its Apollo reservations network. CI originated in the mid-1980s to allow applications to become independent of network protocols, which in turn would facilitate moving applications to new servers/hosts within the network, allow new hardware and software platforms to be added to the network more readily, and simplify the complexities of programming for application-to-application communication.
The Apollo network was already a very large network with data base server applications running on mainframes, new services being added regularly, and transaction volumes growing rapidly. Because plans were being made for PCs and LANs at customer sites, LAN servers, and PC-based LAN-to-WAN gateways to be added to the reservations network, a much more dynamic and adaptable approach was needed for dealing with distributed applications in a changing network environment. It is also interesting to note that between 1985 and the early 1990s when commercial message-oriented middleware was not yet available, many other companies with large networks in industries other than airline reservations were going through similar transitions and developing their own in-house message middleware.

The approach used in the CI, which was sold for the first time in industries other than the airlines industry beginning in late 1991, was to architect an API having consistent functions, verb set, and options (i.e., parameters) across all platforms regardless of operating system, language used for the product implementation, or network protocols supported. The Communications Integrator API allows applications to register with the message service and then call a send routine to send messages or a receive routine to receive messages. Applications do not have to deal with the problems of network sessions because the Communications Integrator, running on each node, takes care of all session management under the covers.

When sending a message, applications take advantage of the CI's directory services to simply specify the name of the application to receive the message; select a message type (i.e., asynchronous or one-way, or synchronous or query with correlated reply); select other options such as priority, assurance level, and whether notification is requested; and then issue the send. When receiving a message, applications select the mode (i.e., blocking or nonblocking), select whether looking for a reply to a specific query or simply the next one available, and then issue the receive.

An Example of Middleware Use.

Healthcare Data Exchange (HDX), headquartered in Malvern PA near Philadelphia, provides one example of how middleware is used in a large distributed application. Using the Communications Integrator, Half-DupleX channel has implemented a patient insurance eligibility and claims system for its multistate network of healthcare providers. Client applications resident in PCs at the admission desks of providers initiate requests for patient records, eligibility, and admissions based on information supplied by the patient. These requests are sent to appropriate server applications running on local servers or on mainframes at HDX data centers. Switching applications at intermediate servers may trigger multiple requests to systems both inside the HDX network (e.g., HDX claims processing on an IBM mainframe or HDX accounting systems on Digital Equipment Corporation mainframes) and outside (e.g., eligibility with Medicare or commercial insurance companies). Responses containing screen displays, printed patient records, admissions paperwork, or billing statements are sent back to the admission clerk's PC or to a print server application located nearby. Response times must, of course, be kept as short as possible.

The networked healthcare information business places great demands on client/server applications. In the HDX case, middleware provided flexibility and adaptability to deal with several different platforms, the possibility of future network changes such as from System Network Architecture to TCP/IP, and rapid growth, while at the same time simplifying programming through a higher-level message API.
Although the Communications Integrator is no longer being sold, other middleware products are now available. Some examples include MQSeries from IBM, DECmessageQ from Digital Equipment Corp., and Pipes from PeerLogic. In addition, distributed transaction monitors, such as Tuxedo from Novell and TopEnd from AT&T, are now also positioned as message-oriented middleware products.

**Trends In Middleware Development**

Given the multiplatform, multiprotocol world in which most modern enterprises operate, middleware has come into existence in the last ten years as a necessary means of providing applications with a degree of insulation from the differences across platforms and protocols. As such, middleware allows the applications to be less affected by changes in those platforms and protocols, while simultaneously providing interoperability across a heterogeneous IT environment.

There is a great explosion in products within each niche or type of middleware, and new types of middleware products are being developed to meet new needs and to solve new interoperability problems. The rapid growth of the Internet, for example, has generated new products for Internet-based applications, and middleware that allows applications on corporate desktops and servers to interact in real-time with applications on Internet servers should be available soon. Middleware development is still in its growth stage, and the middleware marketplace has not yet seen much consolidation.

Ultimately, the market will determine which are the preferred middleware solutions. Such solutions will likely be strongly influenced by other IT trends, such as the development of object-oriented and multimedia technologies. In the end, the preferred middleware solutions must not only be embraced by end users, they must also be integrated by software vendors into the application and tool products that must interface with the end users' custom applications.

Critical issues to customers will be whether the middleware supports the customer's particular platforms and network protocols, is relatively easy to use, and is relatively easy to manage—that is, whether and how easily the middleware can be installed, configured, and tuned in a distributed manner. The market must also contend with issues relating to the degree of integration and compatibility with other middleware products and with common applications, especially those used by each customer to conduct day-to-day business.

Although applications developers would like it to be otherwise, evolution of middleware products, along with other client/server tools, will take time—maybe five to ten years. In the meantime, businesses must be able to solve their interoperability problems so that they can implement distributed computing solutions that meet business needs. In some cases, these systems might be characterized as enterprisewide information systems that are used throughout the enterprise and allow the enterprise to act in a more integrated way in serving customers. There may also be smaller enterprise client/server applications that improve some business process, such as customer support, by automating and facilitating customer interaction in a consistent way across many different functions of the enterprise.

In any case, distributed systems today, and for at least the next several years, will likely use point solutions—middleware tools selected according to the unique requirements of the particular system being implemented—rather than integrated solution sets that are suitable for use in all distributed applications of the enterprise.

Given time, however, client/server software and middleware tools will inevitably achieve greater maturity, and integrated solution sets will be offered by the major software
companies. Many software vendors, just like end users, are struggling to deal with diverse platforms and protocols and the related interoperability problems. Some vendors specialize only in selected software markets and systems, such as PCs or UNIX, but the most complete solutions will likely come from the software vendors who are now established players in enterprise networking, such as IBM or Computer Associates, or those who may be able to expand to that level, such as Microsoft.

**Recommended Course of Action**

Because most situations in which organizations are striving to implement client/server applications are unique, IT staffs should research middleware options themselves or hire specialist consultants to find the best solutions to meet their specific requirements. In some cases, for example, a distributed DBMS such as ORACLE may fit the particular situation; in others, message-oriented middleware may provide the right interoperability solution.

Assessing and managing the risks involved in proposed solutions cannot be taken lightly, however. Proof of concept should be considered a necessary phase of any first-time undertaking or sizable project to ensure that the software and the hidden complexities that are part of large-scale and mission-critical client/server applications are fully understood. System requirements must address the adaptability and probable life of the middleware as part of the adaptability and probable life of the overall client/server application. These strategies can be used to manage middleware decisions and distributed application projects.

Many successful mission-critical applications have been implemented in recent years, and middleware tools supporting such applications should continue to evolve to meet the needs of the market. As the market matures, middleware products will have added functions and features, improve in performance, and become more proven in real business conditions. These are the attributes that enterprise-level client/server computing demands.

**Author Biographies**

Raymond J. Posch

Raymond J. Posch is the manager of education services for Covia Technologies of Englewood CO, a software and professional services company known for middleware products and systems solutions targeted at the travel and transportation industries.