Patricia Seybold Group

Editor-in-Chief
Michael A. Goulde

INSIDE

EDITORIAL

Page 2

***Removing Barriers to Reengineering** must begin with the technological barriers which are imposed by closed, rigid, proprietary information systems. In order to deliver information on an immediate basis, the ultimate goal of most reengineering efforts, systems must have all the characteristics we attributed to open information systems.*

ANALYSIS

Page 23

***The Open Software Foundation** celebrated its fifth birthday with Challenge '93, a demonstration of **OSF** technologies featuring DCE. Although Motif has achieved industry standard status, DCE and DME are where the action is for open distributed computing, and they will take the **OSF** through the next five years.*

OPEN INFORMATION SYSTEMS

Guide to Unix and Other Open Systems

Vol. 8, No. 6 • ISSN: 1058-4161 • June 1993

Open Electronic Mail

Interoperability through Standards

By David S. Marshak

IN BRIEF: The goal of ubiquitous mail connectivity is driving vendors and users to a standards-based mail approach. There are three types of standards being proposed and implemented: desktop, de facto, and de jure. The desktop standards wars are being fought over who controls the "standard" APIs for mail-enabling desktop applications. The de facto approach focuses on the Internet and the Unix Simple Mail Trasfer Protocol. And the de jure approach taps X.400 and X.500 as the key elements of global mail interoperability.

In this report, we find that the choices of standards are not mutually exclusive. Rather, there is an evolutionary path which should eventually bring about a high level of standardization and interoperability while maintaining user choice and flexibility.

Report begins on page 3.

Removing Barriers to Reengineering

Rapid Deployment Is Critical for Success

SOME SKEPTICS MAY claim that business process reengineering is just another consultants' big-money scam or that business processes that weren't engineered in the first place can't be *re*engineered. However, we have reached a point now where a significant number of case studies has been reported in the press citing companies which have realized important gains from their business process restructuring efforts. Many successful reengineering projects haven't been driven by technology, and, in many cases, new technology hasn't been required.

However, in the vast majority of instances of successful business process redesign, information technology has played a central role. The reason is simple: Information plays a central role in virtually all business processes, and, in order to reengineer those processes, one has to reengineer the associated information systems and technology.

The challenge of redesigning business processes is in throwing out all previous assumptions and practices and focusing on the basic objectives that need to be accomplished. Whether the objective is delivering shipments of shoes to a large number of small customers as quickly as to a few large customers, cutting the time required to underwrite an insurance policy from 15 days to 5, or approving a lease in six hours instead of six weeks, the task for the business process engineer (is that who does reengineering?) is distilling the essential steps and ignoring "the way we've always done it." Without fail, information becomes the critical variable as well as the critical design element.

Once the revised process has been optimally designed and the required information flow has been determined, the next step is for the technologists to step in and design systems to implement the optimum information flows. Here is where things can start to get ugly. The scene at a typical planning meeting often goes something like, "We want to give the telemarketing representatives access to customer account information, but their PCs are running 3270 emulation to connect to the order entry system, and they aren't connected to the account information system. To make matters worse, the interface builder we used for the graphical user interface to the order processing system can't be used to develop a compatible interface to the customer account system. We could strip everything out and start over, but we cannot afford that option, and we don't have the time to rebuild the back-end systems. The interfaces and gateways we would have to build would take 30 months to develop."

This kind of scenario is guaranteed to severely undermine an organization's freedom to reengineer business processes without being constrained by existing technology or applications. As processes are restructured, the information required to support those processes is identified, and the flows required throughout the organization for that information take shape. The necessary information may be in existing files or in existing databases. If not, it will need to be captured from some source. The newly-engineered information flows may require that individuals have access to information who have not required it in the past. The information may require new analysis, presentation, or formatting that has not been used in the past. Information may need to be combined in new ways.

Rapid application development is obviously required. What is less obvious is the need to ensure that barriers to information reengineering have been eliminated *before* reengineering so that incompatible systems and the lack of interoperability do not stand in the way of reengineering efforts. Open systems and reengineering go hand in hand. ©

OPEN INFORMATION SYSTEMS

Editor-in-Chief
Michael A. Goulde

MCI:
MGoulde
Internet:
mgoulde@mcimail.com

Publisher
PATRICIA B. SEYBOLD

Analysts and Editors
JUDITH R. DAVIS
STANLEY H. DOLBERG
MITCHELL I. KRAMER
DAVID S. MARSHAK
RONNI T. MARSHAK
JOHN R. RYMER
ANDREW D. WOLFE, JR.

Copy Editors
ALBERT C. D'AMATO
MIRIAM F. D'AMATO

Art Director
LAURINDA P. O'CONNOR

Sales Director
PHYLLIS GUILIANO

Circulation Manager
DEBORAH A. HAY

Customer Relations Manager
DONALD K. BAILLARGEON

Patricia Seybold Group
148 State Street, 7th Floor,
Boston, Massachusetts 02109

Telephone: (617) 742-5200 or
(800) 826-2424
Fax: (617) 742-1028
MCI: PSOCG
Internet: psocg@mcimail.com
TELEX: 6503122583

Open Information Systems (ISSN 1058-4161) is published monthly for $495 (US), $507 (Canada), and $519 (Foreign) per year by Patricia Seybold Group, 148 State Street, 7th Floor, Boston, MA 02109. Second-class postage permit at Boston, MA and additional mailing offices.

POSTMASTER: Send address changes to *Open Information Systems*, 148 State Street, 7th Floor, Boston, MA 02109.

FEATURED REPORT: BY DAVID S. MARSHAK

Open Electronic Mail

Interoperability through Standards

The Problem

The Clash of Expectations and Reality

Imagine that we live in a fully connected world. We can send messages and documents to people in all corners of the globe. We can hold discussions with virtual workgroups consisting of suppliers, customers, and collaborators. We have automated work processes, such as order fulfillment between organizations, to the point that no human intervention is required. And we have the infrastructure to build mail-enabled applications that can treat our disconnected and occasionally connected world as if we were all on a single machine.

This is the vision, and much of it is true. But sometimes, and often more frequently than sometimes, it does not work. Have you ever sent a message to a valid address and received a virtually indecipherable non-delivery message? Or have you ever asked someone to send you information and received an unreadable/unlaunchable/unprintable file attachment? Or have you ever tried to implement something as simple as a group calendar/scheduler and been told that it does not support your mail system? Have you ever felt the frustration that accompanies any of these situations? I have. And I'll guess that most of you have, too.

The result of our mail systems' inability to consistently provide us with a predictable level of service is more than mere frustration. We have reached a point where the gap between the expectations and reality is threatening to paralyze many companies as they struggle through endless meetings on the most politically correct E-mail architectures, relative merits of different backbone approaches, and the effects of the latest machinations in the application programming interface (API) wars.

The war cry to attack these problems has become "Standards!" Your standard, my standard, her standard (but not his standard). Just as long as it's a standard. And, while users are crying standards, the vendors are responding "Open!" It's mine, so it's open; it's yours, so it's not open; it's hers, and it's open (but his is closed). And, by the way, it's X.400-like.

And the standards/open din has led to more confusion and paralysis, with vendors and users alike trying to find a way out of the morass so that the promised systems and services can really be delivered.

In this report, we will examine the current status of electronic mail, focusing specifically on the various efforts at standardizing interoperability between mail systems. We will look at the various approaches that can be taken today. And we will try to answer the question of whether we have enough assurance to rely on E-mail and to build mail-based systems and businesses.

Setting Proper Expectations

Levels of Service

The first problem in understanding electronic mail is the confusion over what mail systems do. The three frustrations experienced above demonstrate the three levels of service that an electronic messaging system should be able to provide:

- Simple messaging

- Rich messaging
- Mail-enabling

SIMPLE MESSAGING. A simple messaging service allows a message to get from one location to another. Its basic job is to assure that the message gets to the correct address. Most messaging systems provide all of the common mail functionality that users have come to expect: creation, receipt, reply, forward, and cc of messages, and the ability to file received mail for future use. The simplest mail system may only provide a subset of these functions. However, all mail systems have three specific components:

- Transport
- Message Store
- Directory

Typically, the mail system moves (Transport) messages between mail boxes (Message Store). The mail boxes are determined by the Address Book (Directory). (For more on E-mail architecture, see sidebar, page 7.)

Within a single mail system, there are usually few problems in sending and receiving messages. The problems begin when you try to connect different mail systems that use different transports, message store file formats, and naming systems.

RICH MESSAGING. While simple mail systems provide for the delivery of the message to the right person, they do nothing to ensure that the receiver can read the content of the message. Simple messaging should be looked upon as delivering the envelope. In order to read the contents, we need additional functionality. Most mail systems support the delivery of text-only messages. The problems occur when richer data types are introduced. These data types can be enhanced text, such as formatted text or international characters. They can also include formatted documents, graphics, images, sound, and video. As our desktop world becomes richer and less confined to simple text, the ability to exchange rich mail becomes increasingly crucial.

Unlike simple messages, rich messages may be unreadable even if delivered within the same mail system—the receiver may need to have the application that created the content to view or edit it. Across systems, this becomes an even greater problem, since frequently much of the rich information is lost when crossing gateways. (See "The Trouble with Gateways," page 18.)

MAIL-ENABLING. The third level of mail service, and one that is receiving increasing attention, is the use of the messaging system to provide services to other applications. There are three categories of applications that require access to mail services: simple mail enabling, mail-based applications, and full mail front ends.

Simple Mail-Enabling. The large majority of what developers want to do with mail falls under the category of simple mail. The simplest form of mail-enabling is allowing mail to be sent from within another application, such as Microsoft Word or Lotus 1-2-3. The application is generally termed "mail-aware." The mail-aware application may present the user with a mail interface or dialog box, or it may limit the user to sending the current document to a specific address. The latter case would be used for mail-enabling an order-entry system, for example. Simple mail-enabling is most commonly used to add a "send mail" icon or menu choice to an existing application, such as a word processor or spreadsheet.

The key requirement for simple mail-enabling is that it be kept simple. Mail should be able to be sent from within another application via a single call or very few calls.

Mail-Based Applications. Mail-based applications are those whose primary purpose is not messaging, but which, nonetheless, need underlying E-mail services. These include workgroup applications such as calendar/scheduling and workflow management. (This is not to say that all of these workgroup applications must be based on mail. They can also be effectively built on shared databases or file systems.)

In a mail-based application, the movement of information via the mail system is an essential part of the application. The application needs access to the mail system services, including directory service look-up and manipulation and direct access to the message store. It may also require the mail service's security and authentication facilities where available.

Mail-based applications must have robust and fully-functional access to all of the mail services. Transport-only access is not sufficient.

Full Mail Front Ends. ISVs writing full mail front ends independent of a specific back end (examples here are DaVinci, Reach, and Beyond) require complete access to the mail services. They need to be able to treat the back end as if it were their own. They also require complete control over the user interface. They cannot accept any compromise of functionality in the name of "openness."

CROSS-PRODUCT REQUIREMENTS. For all three of these service levels, certain formats, protocols, or APIs are required to create cross-platform and cross-product functionality. Simple message transfer across systems requires the translation of file formats and the reading of address information; this can most often be achieved via a file-based API that can be used to move messages in and out of the message store. Cross-product messaging is generally handled in a binary way—with a gateway running between two specific systems mapping the formats of each system to the other. A backbone approach translates each format into an intermediate format before translating it back into the destination format. (See below for more on gateways and backbones.)

Rich messages can be handled with viewers and launchers provided by the mail system, which allow the user to see an attachment. These require the support of specific file formats or platform-specific interapplication communication protocols such as Object Linking and Embedding (OLE) in Microsoft Windows. Mail systems may also support specific rich formats such as the international standard Open Document Architecture (ODA), Microsoft's Rich Text Format (RTF), or the WordPerfect document format.

Mail-enabled applications require programmatic access to the mail system. This enables these applications to use the specific functionality of the mail system, rather than to merely have access to the messages in the message store. Access may be required to the mail system as a whole or to just one component—transport, message store, or directory. It is this area that is currently receiving the most attention as companies seek common APIs for desktop applications to reach mail services regardless of the vendor of the mail system.

The Standards Landscape

Levels of Standardization

There is no shortage of efforts at coming to standards in the world of electronic mail. International standards bodies, industry consortia, and market leaders are all proposing and promoting their standards. Some, such as the ISO and CCITT X.400 protocol, are being widely accepted, while others are still being hotly debated. (See Illustration 1.)

The standards efforts fall into three distinct areas:

- The desktop standards movement, which is oriented toward providing common mail APIs that promote mail-enabled applications and mix-and-match mail systems

- The de facto standards approach, which seeks to leverage the wide acceptance of the Internet and Simple Mail Transfer Protocol (SMTP)
- The de jure standards movement, which uses legislated protocols—X.400 and X.500—to assure interoperability between systems

Layers of Mail Standards

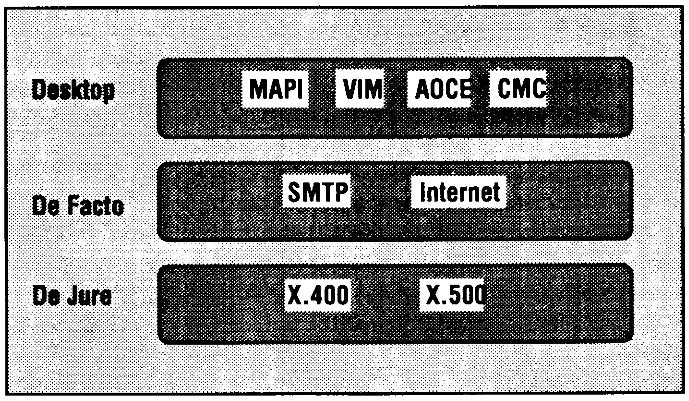

Illustration 1. Attempts to promote standards are occurring at three levels: desktop, de facto, de jure.

Desktop Standards Movement

Different Goals Create Confusion

There are two specific goals for providing standard desktop APIs to mail systems. The first, both chronologically and in complexity, is to allow desktop applications to use the underlying messaging system without having to be rewritten for each vendor's mail product. This goal is set mainly for mail-enabling current applications and for allowing new applications, chiefly workgroup applications such as calendar/scheduling and workflow, to be built on top of the mail system.

The second goal is to allow users to mix and match mail front ends and back-end services. Thus, users could choose a mail client from one vendor and a mail server from another—or even choose each mail service (transport, message store, directory) from a different source. With the client and server written to a published API rather than to each other, users would have a wide range of choices as well as a new level of investment protection.

Much of the confusion in the desktop API arena is due to a lack of recognition that there are two distinct goals operating here and that a given API set that promotes one goal should not necessarily be compared with another set that promotes the other. Specifically, much of the ado between VIM and MAPI is due to a lack of understanding of each set's fundamentally different goals. In order to sort out the confusion, we must first look at the acronyms and the players.

E-Mail Architectures and the Role of APIs

E-Mail Architecture

Front End and Back End

Electronic mail systems normally consist of two parts—a front end that handles the user interface (UI) and a back end that provides the mail services. Most PC LAN mail systems are not built on a client/server model. Thus, both the UI and the mail services actually run in the client machine, though both the application code and the actual mail messages may be stored on a file server. Exceptions to this are products such as Lotus Development Corporation (Cambridge, Massachusetts) Notes and Digital Equipment Corporation's (Maynard, Massachusetts) MailWorks, where part of the mail system runs on the client and part runs on the server. Communication between client and server is via remote procedure calls (RPCs).

Historically, mail systems have come with the front end directly tied to the back end. The UI application is directly bound to the specific transport, directory, and message store services provided by the single vendor that sells both. And, in the case of a client/server mail system, the RPCs between the front and back end have been proprietary.

What Are Mail APIs?

Providing Access to Mail Services

Mail application programming interfaces (APIs) are simply interfaces that programmers use to access services common to any mail system—transport, message store, and directory. They allow applications, be they full mail clients or your common word processors, to do one or more of the following: submit a message to the mail service, receive a message, file or delete that message. Other mail capabilities, such as directory look-up and manipulation or direct access to the message store to move messages around, can also be exposed by APIs.

Client APIs Access the Mail Services

In order to build a mail-enabled application based on published and common APIs, the application has to make calls that will eventually be executed by some back-end service. This can be accomplished in two ways. The calls can go to a mail subsystem (part of the operating system that handles mail independently of specific applications) that transfers them to a set of back-end services. Or the calls can go to a specific application running on the client that knows how to connect to the mail services. This application can be a small "applet" or a full mail front end. In either case, a specific set of code that receives the API calls and transfers them to the back-end services must run on the client. (See Illustration 2.)

Services Must Support These APIs

In order for this to work, the mail services must recognize the API calls and be able to act upon them. Mail services may support one or more sets of API calls and thus support front-end applications that use different APIs.

Types of Mail APIs

FILE-BASED APIs. There are two types of APIs in a mail system. The first, called a file-based API, allows an application to submit files to the mail transport to be sent to an address. File-based APIs, which are published for most mail systems, provide only a limited degree of mail-enabling. Generally, these APIs only permit applications to submit files to the mail transport for sending.

PROGRAMMATIC APIs. More robust and functional are programmatic APIs, which allow other applications to use the messaging services; for example, the other apps could use directory look-up and manipulation of the message store by allowing messages to be moved from one folder to another. Access to programmatic APIs has generally been reserved only for the E-mail vendor itself and for specific partners working with that vendor. The current debate over APIs centers on these powerful programmatic APIs.

E-Mail Architecture

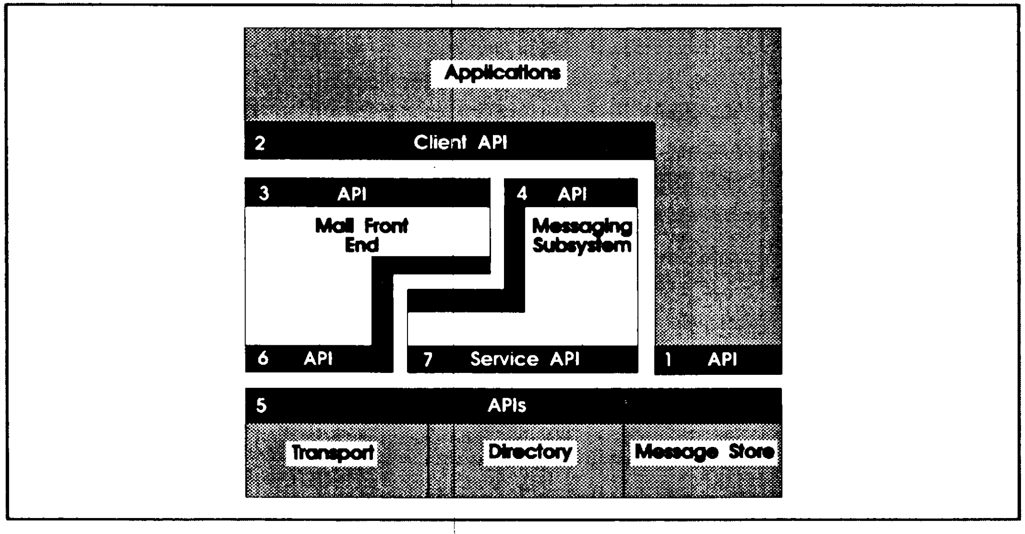

Illustration 2. Electronic mail APIs provide access to messaging services from client applications. These applications may have to be written directly to a back end (1) or can make use of another application (2 and 3) or an operating system-specific mail subsystem (2 and 4) to reach the mail services. The back-end services must support one or more sets of API calls (5), which come from the mail front-end applications (6) or messaging subsystems (7). Front-end applications may call services directly or through a messaging subsystem. APIs may be proprietary or "open." Examples of open APIs are Microsoft's MAPI (2,4,5,7), Apple's AOCE (2,4,5,7), CMC (2,3,4), and VIM (2,3).

The Meaning of "Open"

Much of the confusion over APIs exists because virtually all the players who have an API claim that their API is open. Openness is in the eye of the beholder. It can mean an API agreed upon by some independent standards body, it can mean a *common* API, or it can mean a *published* API. Most of the time, it means that someone is trying to sell you something.

E-mail APIs are best categorized as follows:

- Standard APIs. Standard APIs, such as the X.400 API Association's X.400 API or the X.500 Directory Service API, are those recognized and promoted by national or international standards groups. If a large enough group within the industry supports a given API, it is termed a *de facto* standard API. The most notable de facto mail standard is the Simple Mail Transfer Protocol (SMTP). (See page 11.)

- Common APIs. Common APIs can be used to access services from more than one mail product. This enables a single front end to be written to access multiple back-end services.

- Published APIs. Published APIs allow anyone (not just the mail system provider) to write applications that use the exposed services. This generally enables multiple front-end applications to be written to a single back-end service.

The Standards Landscape

The Acronyms and the Players

MAPI, SPI, VIM, AOCE, MHS, XAPIA, XAPI, CMC. Acronyms have become the code words of the mail API debates. The first thing we must do is clarify which is which and to whom each belongs.

MAPI. MAPI is Microsoft Corporation's (Redmond, Washington) *Messaging Application Programming Interface.* It will first appear as a Windows 3.x and Windows NT subsystem and eventually be ported to other platforms, including Mac, DOS, and OS/2.

There are two flavors of MAPI—Simple and Full. Simple MAPI is now available as a Windows 3.1 extension. It ships as part of Microsoft's Windows for Workgroups. Full MAPI, which implements the SPI (see below), is due later this year.

Microsoft Mail is the first back end that accepts MAPI calls. Other companies, including AT&T, Digital, Hewlett-Packard Company (Palo Alto, California), Novell Corporation (Provo, Utah), Banyan Systems (Westboro, Massachusetts), CompuServe (Columbus, Ohio), SkyTel (Washington, D.C.), and Soft*Switch (Wayne, Pennsylvania), have committed to supporting the MAPI interface on their mail systems.

SPI. SPI is Microsoft's *Service Provider Interface,* which is the part of MAPI that allows developers to integrate back-end mail services, such as transports or directories, into the Windows subsystem.

VIM. VIM is the *Vendor-Independent Messaging* interface that is jointly backed by Apple, Borland International Corporation (Scotts Valley, California), Lotus, and Novell. VIM supersedes the earlier OMI effort, which was based on Lotus's cc:Mail and Notes APIs. VIM is being built from the beginning as a cross-platform interface, and it will be supported on Windows (by Lotus), DOS (probably by Lotus), OS/2 (by Lotus and IBM), Macintosh System 7 (by Apple), and Unix (probably by Lotus). Novell has committed to support VIM on MHS. The VIM Interface Specification V. 1.0 is now available for developers. The first VIM-compliant product is cc:Mail. The recent Release 3.0 of Lotus Notes is also VIM compliant.

AOCE. AOCE is the *Apple Open Cooperative Environment* for the Macintosh. AOCE is available as an extension to System 7. Like MAPI for Windows, AOCE provides a messaging subsystem accessible to Macintosh applications and messaging service providers. Microsoft, for one, has talked about making its mail engine AOCE compatible. Apple is talking about providing AOCE capabilities on other platforms, though no firm commitments have yet been made.

MHS. MHS here refers to Novell's *Message Handling Service,* though sometimes the acronym is also used for the generic message handling function within a mail system. MHS is actually the transport service (Standard Message Format [SMF] is the API into MHS). MHS was first developed by Action Technologies Incorporated (Alameda, California) for its Coordinator product. It has since been taken over by Novell and incorporated into NetWare. MHS is a back end that supports multiple front-end E-mail packages such DaVinci Mail (Raleigh, North Carolina), BeyondMail (Cambridge, Massachusetts), and Reach Software Corporation's (Sunnyvale, California) MailMan. MHS is currently the closest product we have to an open back end, and it is a clear choice for those who are trying to build mail-enabled applications without tying themselves to a single mail product. Eventually, we see MHS as simply one of the services accessible via one of the other APIs.

XAPIA and XAPI. The X.400 APIA, the X.400 API Association, is a consortium of vendors that needed to define the way to reach X.400 services. X.400, as defined in the Open Systems Interconnect (OSI) model, did not provide an API to reach its services. Out of this consortium came XAPI, the *X.400 API Gateway Protocol.* This protocol is the most common one used for other systems to communicate with X.400 services. (For more on X.400, see "De Jure Standards," page 12.)

CMC. As pressure from users for a cross-platform API increased, a number of vendors looked to the XAPIA to define a set of Common Mail Calls. This process, which is supported by Microsoft, Hewlett-Packard, and the VIM Consortium, is expected to deliver its final specification this month.

Positioning the Desktop APIs

As we noted above, there are two goals for desktop mail APIs. CMC and VIM are oriented toward the first goal. They are specifically built to provide access to full mail systems. Their greatest values are in being cross-platform and relatively simple to implement. It is likely that VIM will eventually be incorporated into CMC.

MAPI and AOCE also provide applications with access to the underlying mail system. However, in addition, they seek to provide a mix-and-match capability by separating the mail front-end application from the back-end services. Both MAPI and AOCE are essentially mail subsystems that are part of an operating system platform—Windows 3.x/Windows NT and Macintosh System 7, respectively. Both provide a mail client and local mail services—transport, message store, directory. Each also provides a method for other vendors to supply more robust versions of these services.

Right now, most attention is being paid to CMC as a simple cross-platform API, with virtually all mail vendors voicing public support. MAPI and AOCE are emerging as the platform-specific APIs of choice. VIM remains somewhat viable, since it is supported in both cc:Mail and Lotus Notes. Lotus has committed to providing a Windows driver that will translate VIM calls to MAPI calls. It is also likely that Lotus will write a future version of cc:Mail to the MAPI SPI specifications.

As the dust settles, the results seem to be very good for the user. At some point in the not too distant future, an application written to any of the predominant desktop APIs—CMC, MAPI, AOCE, VIM—will be able to use any major mail system on any major platform.

Limits of Desktop APIs

The availability of common desktop mail APIs will go a long way to increase the availability and rate of adoption of mail-enabled applications. This will benefit both developers (internal corporate developers as well as ISVs) and users, with users both seeing a greater selection of applications and experiencing greater confidence that their investments in messaging will be highly leverageable.

However, desktop APIs only directly address the third level of messaging service—support for mail-enabled applications. They cannot fully assure that mail can be effectively exchanged between two systems; until everyone agrees on a single API set or all mail products support all APIs, gateways between systems will still predominate. And they do nothing to address the fidelity of message content across mail systems and across platforms.

Thus, the good news on desktop APIs must be taken with a bit of caution. Other, hopefully complementary, solutions are required to meet the promise of electronic mail.

De Facto Standards

The Internet—Ubiquitous E-Mail Today

While many debate what it will take to reach ubiquitous messaging, a significant group argues that it exists today. People and organizations that use the Internet to communicate question what all the fuss is about. On a daily basis, they are able to send and receive mail from virtually anywhere in the world.

De Facto Standards

In the past few years, the Internet has grown from an academically oriented, Unix-bigoted, propeller-head-dominated playground to a significant presence in business-to-business communications. These companies may use the Internet as simply an intermediate transport between two systems. However, more and more, presence on the Internet—FTP-ing significant information or participating in a news group—is becoming a necessity in many areas of the commercial world.

Internet Mail Protocols

Internet protocols are overseen by the Internet Activities Board (IAB). The IAB requires that any person or group desiring to implement (or design, document, propose, test, etc.) a protocol for the Internet must document that protocol with a Request for Comment (RFC). Each RFC is assigned a number and published. The RFC then goes through three stages: *proposed* standard, *draft* standard, and *Internet standard.*

There are four significant mail-oriented protocols, shown in Illustration 3, for the Internet:

- Domain Name System (DNS) is defined in RFC 1034 and 1035 and provides mapping between host names and IP addresses.
- Simple Mail Transfer Protocol (SMTP) is based on RFC 821 and provides services for store-and-forward of text messages. RFC 822 defines the format for these messages.
- Post Office Protocol (POP) is defined in RFC 1225 and provides a simple mailbox retrieval service. Two other retrieval service protocols are in more limited use: Interactive Mail Access Protocol (IMAP)—RFC 1203—and Distributed Mail System Protocol (DMSP).
- Network News Transfer Protocol (NNTP) provides for store-and-forward of news articles. NNTP is defined in RFC 977.

The Internet Mail Architecture

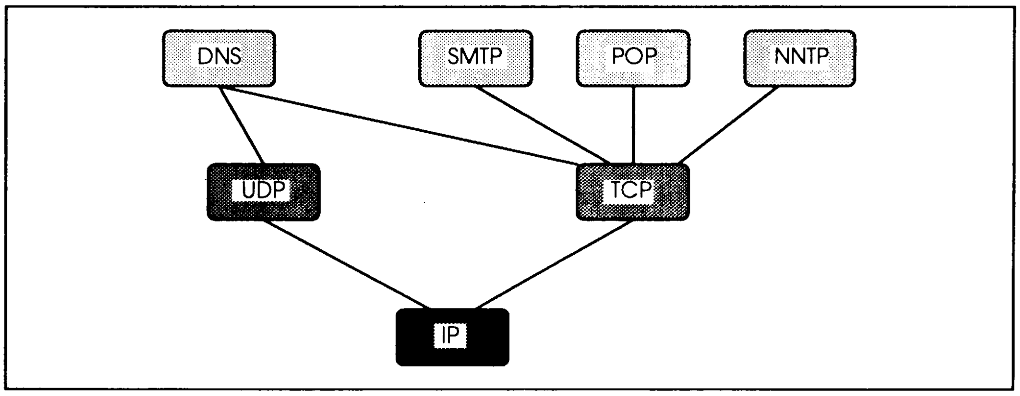

Illustration 3. Internet mail is built on four key protocols: Domain Name System (DNS), Simple Mail Transfer Protocol (SMTP), Post Office Protocol (POP), Network News Transfer Protocol (NNTP). These protocols run over User Datagram Protocol (UDP—the connectionless-mode transport service for the Internet Protocol) and Transmission Control Protocol (TCP—the connection-oriented transport service for the Internet Protocol).

MULTIMEDIA VIA MIME. There is now a strong effort to take Internet mail from text-only to support multimedia. This is really the first attempt to define message content, since SMTP can only define the message envelope. The technology that is being introduced is called MIME (Multi-purpose Internet Mail Extensions). MIME adds the specification of body type to the content of Internet messages. MIME defines seven body types, each of which can have an unlimited number of subtypes:

De Facto Standards

- Multipart
- Message
- Text
- Image
- Audio
- Video
- Application

The purpose of MIME is to ensure that any body part can be displayed (or played) by any user agent. MIME is looked upon by some as leapfrogging X.400, which has traditionally been seen as having a multimedia advantage over Internet mail. X.400 does not define such a specific set of body types. Rather, it relies on Body Part 15, which allows the content to be self describing to a target application. (See page 14.)

SMTP Backbones?

In addition to accessing the Internet for cross-organizational communications, many companies are using SMTP as the common protocol to connect their different mail systems. All Unix and most current LAN-based mail systems have SMTP gateways, which makes SMTP a logical choice for an enterprise-wide backbone.

The Internet and X.400

Although the Internet and X.400 are frequently viewed as mutually exclusive, there are efforts to bring these constituencies together. First, there are a number of RFCs, most notably RFC 987 and RFC 1148, that map the calls between RFC 822 and X.400. Second, there are some efforts, including RFC 1006, that allow X.400 to run over TCP/IP in addition to its native OSI stack. Other efforts in this area include work being done by Hewlett-Packard, OSIware, and the ISODE Consortium. Each of these companies has developed software—Hewlett-Packard and OSIware for their own X.400 backbones, the ISODE consortium for its member companies—that allow X.400 and X.500 to run in non-OSI environments.

The Internet vs. X.400

There is a running debate between advocates of both camps and both can deliver very convincing arguments to back up their cases. Ultimately, this is not so much a technical argument as a discussion of reality. Today, many more companies transfer mail to each other through the Internet than through an X.400 connection. However, as more and more companies commit to X.400—and we firmly believe that this is the dominant trend—the Internet may lose favor as an intermediate transfer point. However, we are well aware that these predictions have been made for OSI protocols before (OSI vs. TCP/IP or CMIP vs. SNMP for network management) and have yet to be realized.

De Jure Standards

What Are X.400 and X.500?

X.400 and X.500 are international standards developed by the Consultive Committee on Telephone and Telegraph (CCITT) and ISO to enable the interchange of electronic messages and name/address information. X.400 defines a Message Handling Service (MHS) specification which, if written to, guarantees interoperability between mail systems. X.500 defines specifications for global directory services. Both are OSI application layer (layer 7) functions that are designed to run on an OSI stack. However, because of their increasing popularity, users are attempting to standardize on their services without being bound to a specific transport.

X.400 and X.500 are important for two reasons. First, they serve as reference examples for vendors to build their products. For over a decade, X.400 has provided service definitions for electronic mail that have been the basis of virtually all E-mail products, whether they support the standard or not. Likewise, X.500 has strongly influenced the new generation of hierarchical naming schemes for directory services. In both of these cases, X.400-like or X.500-like mail service has served to bring divergent architectures closer together.

The second impact of the X.400 and X.500 movements is that companies are beginning to standardize on them—in fact or in principle—as open, cross-platform standards for mail and directory. While in no way yet fully implemented (or even fully implementable), these standards are driving corporate (and government) architectures and purchases. It is not only unfashionable to dismiss X.400 (and, to a lesser extent, X.500); for many—both vendor and customer—it is the equivalent of technological suicide.

X.400 Architecture

X.400 is an alphabet soup of services and protocols. Some of the key ones to be aware of are:

- ADMD and PRMD define the two types of X.400 domains: Administrative Management Domains (ADMD), which are public services such as AT&T, MCI, Sprint, etc., in the United States and PTTs in Europe; and Private Management Domains (PRMD), which are private networks. This distinction is often compared to public telephone systems and customer premises equipment such as PBXs. (Remember that the CCITT is dominated by "phone companies.")

- UA + MTA = MHS. The User Agent (UA) and the Message Transfer Agent (MTA) are the two key components of the X.400 Message Handling System (MHS). (See Illustration 4.)

- P1 protocol defines MTA-to-MTA message delivery and routing (defined in X.411).

- P2 protocol defines syntax for UA-to-UA messages, i.e., messages sent between people or interpersonal messages (defined in X.420).

- P3 protocol provides UA-to-MTA connection (defined by X.411). P3 is not widely implemented in 1984 X.400 because, in a gateway environment, the gateway acts as an MTA and uses P1 to talk to other MTAs. P3 is redefined for 1988 X.400. (See "1984 to 1988" below for 1988 X.400 protocols.)

- XAPI gateway protocol allows non-X.400 mail systems to communicate with X.400 MTAs. It is the result of a consortium of vendors known as the XAPI Association. XAPI gateway protocol is also supported by X/Open.

X.400 messages have two parts: Envelope and Content. Most of the work prior to the 1988 specifications has gone into defining the X.400 envelope. The key parts of the envelope are the O/R addresses, which are the originator/recipient names. The basic X.400 address structure is:

Country Name
ADMD Name
PRMD Name (optional)
Personal Name (optional)
OrgName (optional)
OrgUnitNames (optional)
Domain-defined attributes (optional)

1984 TO 1988. X.400 comes as a set of recommendations from the CCITT. The first implementable set of recommendations was proposed in 1984. Because it takes about four years to reach agreement on the proposals, virtually all X.400 implementations between 1988 and 1992 were built on the 1984 specifications. Last year, the 1988 specifications were finalized, and new products began to appear. (See Illustration 5.) Right now, most X.400 vendors have or are about to have 1988 implementations.

The 1988 X.400 specifications enhanced X.400 to the point that it is ready for prime time implementation (the 1992 specifications being worked on are deemed by most to be only minor enhancements). The major additions in the 1988 specifications include:

- *Body Part 15.* Body Part 15 is an external body part which is self-describing to the user agent. Thus, rich text, multimedia objects, or executable files can be included in mail messages.
- *New P1 Envelope Protocol.* P1 has been extended to support multiple contents and attachments, use of X.500 directories, expansion of distribution lists, and new security features.
- *Integration.* X.500 directory information can now be accessed by X.400 user agents. X.400 MTAs can use directory information to decide the format in which to deliver information to a specific user. The directory can also store security information such as public keys.
- *Message Store.* A separate message store is now available to user agents. This allows client/server access from UA directly to the message store and provides support for off-line and remote users. The message store is accessed by the UAs via the new P7 protocol.

X.400 Architecture

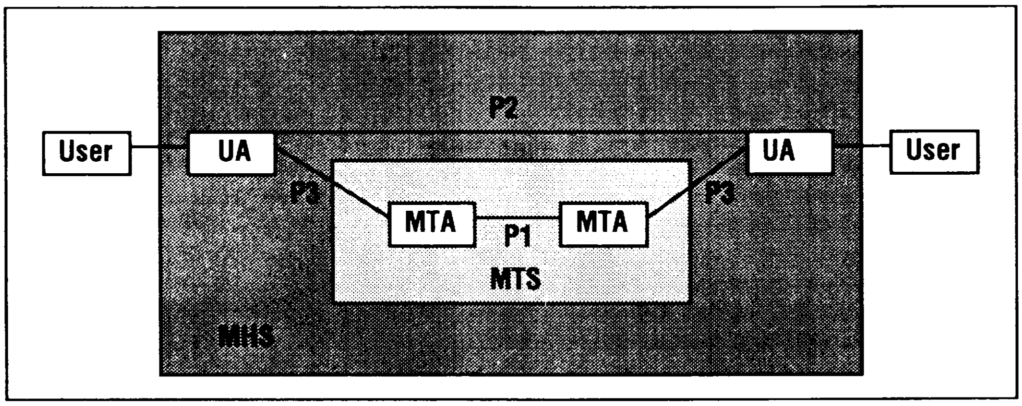

Illustration 4. X.400 architecture defines Message Transfer Agents (MTAs) and User Agents (UAs), which make up a Message Transfer System (MTS). The elements of the MHS communicate via protocols known as P1, P2, and P3.

A key issue for current X.400 customers is how to migrate from the 1984 specifications to the 1988 specifications. The Electronic Mail Association (Arlington, Virginia) has published a guide to *1988 X.400 Migration* prepared by Daniel J. Blum, Rapport Communications, which details the migration and coexistence issues and options.

X.500 Directory Services

X.500 is the specification for adding directory services to X.400 (and other OSI application services). A glossary of terms is shown in the Table on page 16. X.500 directories provide the following services to the 1988 X.400 system:

- Unique naming and addressing
- Distribution management and expansion
- Authentication and security services
- Capability assessment

There are three parts to an X.500 directory service:

- Directory Information Base (DIB)
- Directory Service Agents (DSA)
- Directory User Agents (DUA)

The DIB contains information about a number of classes of entities, including users, resources, networks, etc. The classes are object-oriented, allowing a large number of attributes to be assigned to each. The basic job of the directory is to associate an address with a unique name for each entity.

X.500 supports three different methods of access:

- Whitepages, a one-to-one name/address look-up
- Yellowpages, look-up by attribute
- Browsing

1988 X.400 Architecture

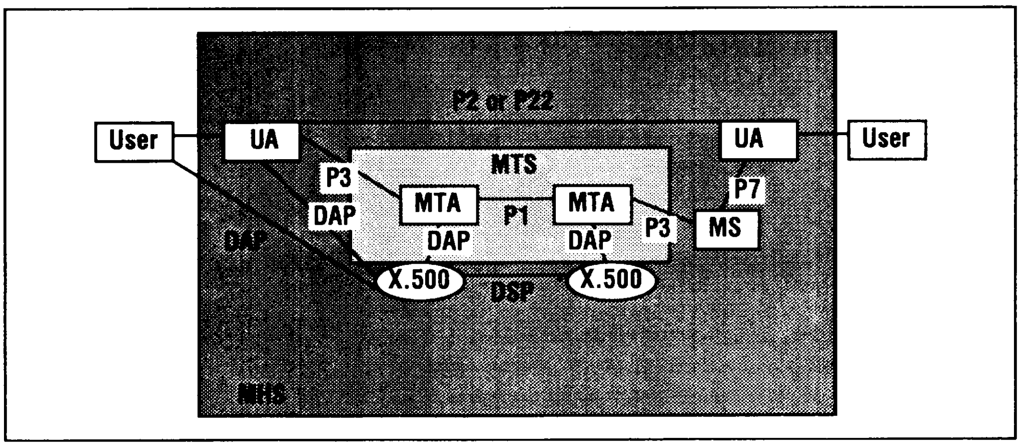

Illustration 5. The 1988 X.400 specifications include a separate Message Store (MS) and X.500 directory services. The Message Store can be reached by the new P7 remote user agent protocol. The X.500 Directory Information Base (DIB) can be accessed via the Directory Access Protocol (DAP). Directories can talk to each other via the Directory Service Protocol (DSP).

While the 1988 X.400 specifications are fully implementable, the 1988 X.500 specifications are still lacking in key areas, such as replication protocols for directory synchronization and standardized access control procedures. Thus, X.500 directory services are optional for 1988 X.400 implementations. There are a number of current X.500 implementations that use proprietary mechanisms to support the synchronization and access control functions. These areas are being addressed in the 1993 specifications.

The XAPIA and the Electronic Mail Association (EMA) are meeting this month to address the development of an electronic messaging directory synchronization standard that would allow current interoperability and easy migration to X.500.

The Evolution of Mail Interoperability: Past, Present, and Future

From Islands to Backbones

We are in the midst of a significant evolution in the way mail systems are interconnected. What were once separate islands can be now connected to each other with somewhat predictable success, with the various standards movements aiming at making success more robust and richer. And, as we speak, the methods of connecting systems are maturing from simple point-to-point gateways to more complex integrated messaging service backbones. (See Illustration 6.)

There are five steps in this evolution:

- Gateways
- Proprietary backbones
- Standards-based backbones
- Integrated client/server backbone
- Integrated standards-based system

X.400/X.500 Glossary

Directory Information Base (DIB)	Collection of information held by a directory service
Administrative Management Domain (ADMD)	A set of Message Handling Systems managed by a public entity
Directory Service Agent (DSA)	An application that provides directory functions
Directory System Protocol (DSP)	Protocol between Directory Service Agents
Directory User Agent (DUA)	Process between the user and the directory
Message Handling System (MHS)	Combination of the Message Transport System and User Agents supporting the exchange of electronic mail
Message Store (MS)	Location for long-term storage of messages for users
Message Transfer Agent (MTA)	Active component involved in the transfer of messages
Message Transfer System (MTS)	Collection of Message Transfer Agents and protocols required to transfer electronic mail
OR Address	Address of an originator or recipient of an X.400 message
OR Name	Name of an originator or recipient of an X.400 message
P1	Protocol to send messages between Message Transfer Agents
P2	Protocol used to send interpersonal messages between User Agents
P3	Protocol used by User Agents to send messages to Message Transfer Agents
P7	Protocol used to send messages between User Agents and the Message Store
P22	1988 version of P2 protocol
Public Management Domain (PMRD)	A set of privately managed Message Handling Systems

Table. X.400/X.500 glossary.

GATEWAYS BRIDGE ISLANDS. The first, and still dominant, method of integrating mail systems is through gateways. There are gateways from virtually every mail system to one or more other systems, and some combination can usually be found to get mail from one specific system to another.

A mail gateway sits between two systems and provides translation services required to send a message from one system to the other. At the most basic level, a gateway must perform three functions which equate to the three essential mail functions.

E-Mail Interoperability Evolution (See Illustrations 7-11 for Details)

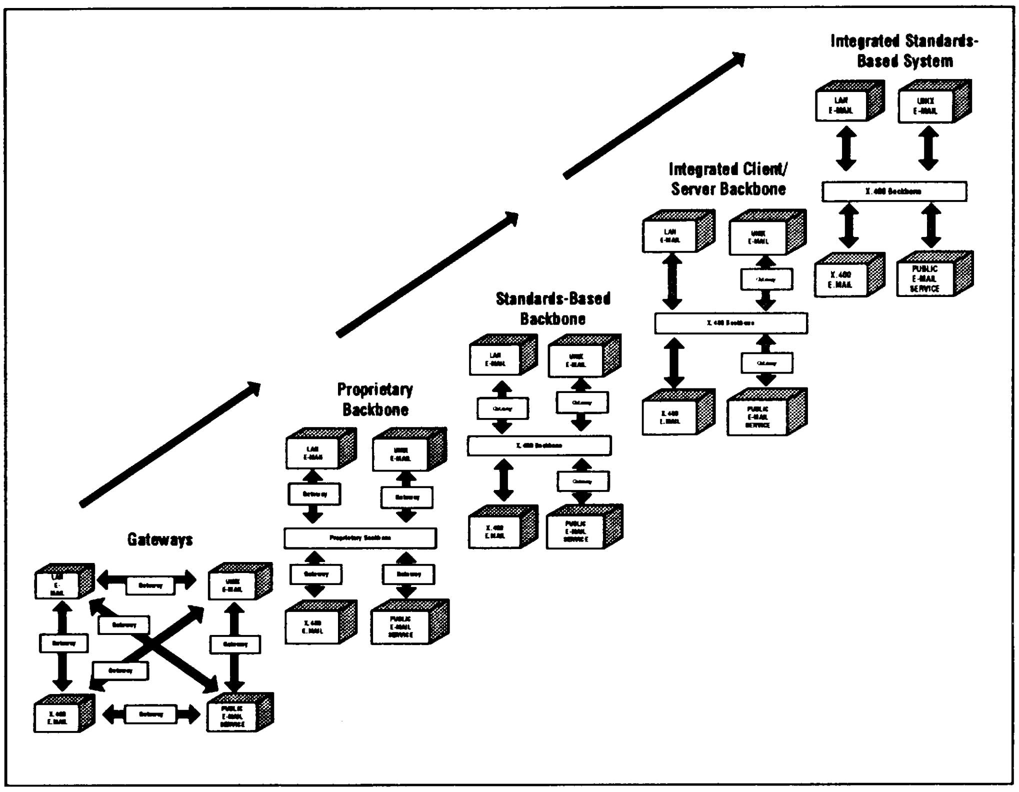

Illustration 6. E-Mail interoperability is evolving from gateways that connect islands of information on a point-to-point basis to fully integrated, standards-based mail systems that provide the user with a choice of clients and services.

First, the gateway must move the message from the jurisdiction of one system to that of the second system. This equivalent of mail *transport* can be as simple as copying a file from one subdirectory (where it was placed by the first mail system) to a second subdirectory (where it can be picked up by the second mail system). Gateways can also be instructed to perform more complex transport acts, such as dialing up another computer to deliver the message to the second mail system.

Second, each gateway must also provide a translation service between the addressing schemes of each system. This equivalent of a *directory* service must ensure that the contents of the "To:" field from the first system can be used by the second system to deliver the message. Translation of the "From" field is also required if the gateway is to support such features as "Reply."

Third, a gateway must translate the message format from that of the first system to that of the second system. This equivalent of a *message store* allows the message to be put into a mail box of the other system. Most gateways can only deal with the simplest aspects of the message format—generally, only plain text.

Gateways may also provide more advanced services, such as management functions (e.g., low-cost routing, notification of non-delivery, etc.) and document translation (e.g., taking a DG CEO document and putting it into Digital's All-In-1 format). The most important added service is directory synchronization, where users of one mail system have continually updated access to the names and addresses of users of the other system. With these added functions, mail gateways can provide a robust interoperability between dissimilar mail systems.

The Trouble with Gateways. For all their usefulness, gateways have two serious limitations. The first is that they rarely, if ever, can do complete translations. Almost always, something is lost. Examples include:

- Message content, particularly any rich information, be it formatted text, multimedia, or executable, can be lost.
- Addressing or routing information becomes incomplete, making messages undeliverable.
- Mapping one system's naming scheme to another's cannot be complete (for example, it is difficult to map both *John Smith @ Marketing* and *John Smith @ Sales* to a system which accepts only a single level of unique names).
- Receiving information about a specific message when it crosses systems can be incomplete. Non-delivery messages within one system may not get back to the originator in the other system.
- Functions such as Registered Mail or Return Receipt rarely work across gateways.
- Translations in one direction do not get translated back in the other direction. Certain characters that are not acceptable to the receiving system (a notable example here is spaces within names) may be translated for the receiving system and may not be retranslated, causing undeliverable messages which are particularly difficult to trace.
- Mail-enabled applications which rely on API calls generally cannot be run across gateways.

And, of course, gateways add cost and points of failure to a messaging system—though there is a counter argument that they add points of management.

The second limitation of gateways occurs despite all efforts to correct the problems listed above. This limitation is due to the very nature of gateways: They are binary translation engines between two—and only two—systems. (We occasionally get into arguments with our proofreaders who like to change "gateways *between* systems" to "gateways *among* systems." The correct understanding, both technically and grammatically, is that a gateway can only run between two systems.)

This means very little if you only have two systems that you want to connect. However, as you add systems, the number of gateways required increases, as does the complexity of managing them. The example in Illustration 7 shows that six gateways are required to connect four systems, and we are aware of companies that internally support over 20 different mail systems. One could eliminate one or more of the gateways by sending messages from one mail system to another via a third system. And, in fact, many companies do just that, specifically when a binary gateway is not available between the two systems in question. The result of this approach is the multiplication of the problems that can occur with a single gateway, and, though this can work, the reliability and fidelity of the system may be in doubt.

BACKBONES AS BLACK BOXES. To eliminate the need for the high number of gateways where multiple systems are involved, the concept of a mail backbone (analogous to a networking backbone) was developed. The backbone architecture provides for each message to be first translated into the backbone's format and then to the format of the receiving system. The backbone is thus a sort of black box where the message enters in one format and comes out in any other format supported by the translation engine. The advantage to this architecture is that it requires only one additional gateway for each new mail system added. (See illustration 8.) The disadvantage is that two translations are required where only one was required in the point-to-point system.

Backbones can also offer additional services such as file conversion and directory synchronization. In addition, backbones can be configured to identify messages bound for a compatible system and to pass through the messages without altering them, thus preserving a higher level of fidelity.

E-Mail Gateway Approach

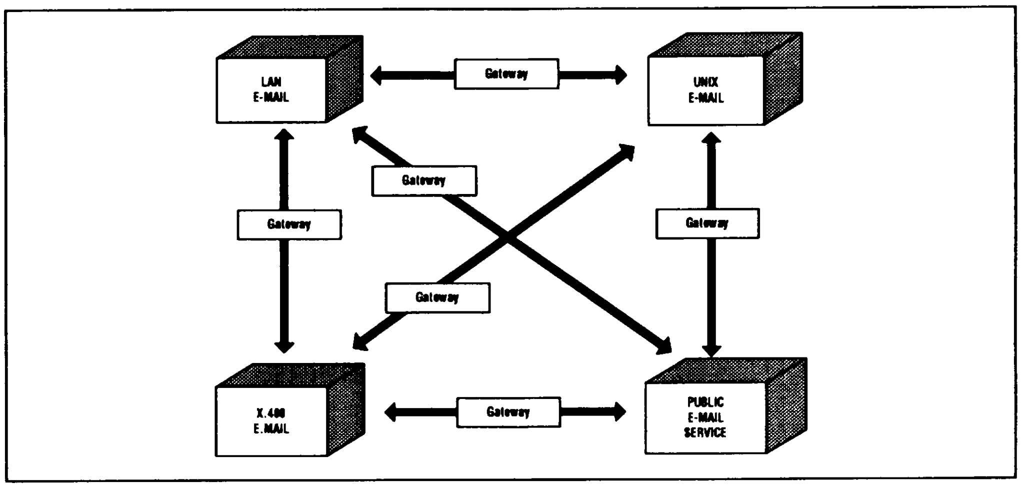

Illustration 7. The number of mail gateways between systems increases dramatically as the number of systems to be connected increases. Here, six gateways are required to connect four mail systems.

Proprietary vs. Standards-Based Backbones. The first backbones, such as those from Soft*Switch and Digital, translated each mail system's format into its own proprietary format and then translated it from its own format into the receiving system's format. More recently, the concept of a standards-based backbone has gained great favor. Here, the internal format of the backbone is no longer proprietary. Rather, it is built on a standard (generally, X.400, but sometimes SMTP or even Novell's MHS, which, though proprietary, can be obtained from multiple sources). This provides a level of investment protection, since the backbone can be swapped out for another backbone supporting the same standard without purchasing new gateways. In addition, a standards-based backbone can provide mail services directly to any system written to that standard, reducing the requirements for gateways and paving the way for the next levels of standards-based messaging. (See Illustration 9.)

Proprietary E-Mail Backbone

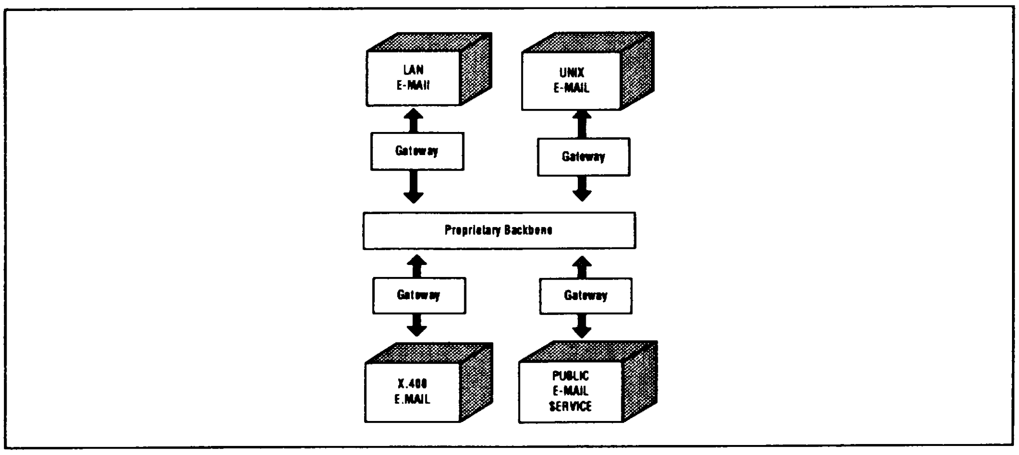

Illustration 8 . Backbone approaches require only one gateway per mail system.

Standards-Based E-Mail Backbone

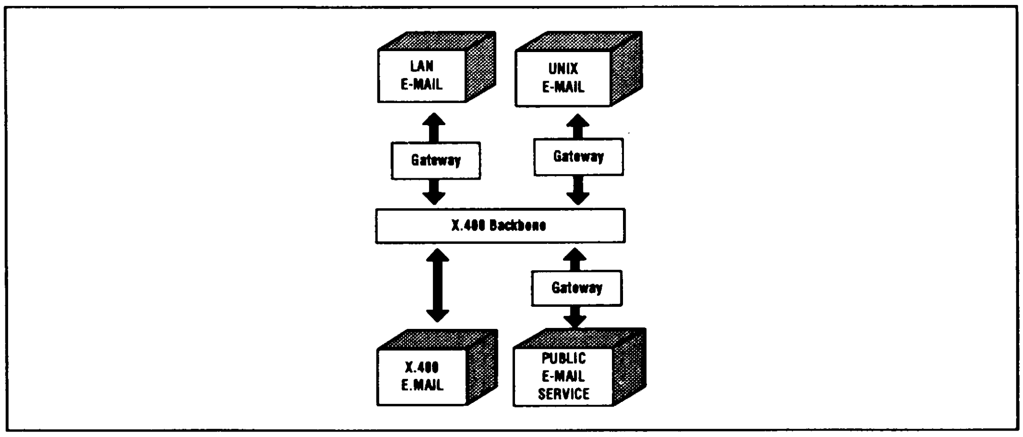

Illustration 9. An X.400-based backbone can directly provide services to an X.400 system without a gateway.

INTEGRATED CLIENT/SERVER BACKBONE. The next level eliminates the requirement for a gateway between the client (e.g., LAN-based) system and the backbone services. Here, the desktop mail clients are separated from the servers, so they can use either their own servers or the standards-based servers that make up the backbone. (See Illustration 10.)

A good example of this architecture is provided by the recent agreement between Microsoft and Hewlett-Packard which offers Microsoft Mail users a driver that switches the transport, message store, and directory services to Hewlett-Packard's OpenMail service. The Microsoft Mail users stay within their own interface. The only indication that they are using different services is any additional functionality offered by the new back end (e.g., in the case of OpenMail, Microsoft Mail users now have access to new directory search capabilities such as wildcards and soundex).

Integrated Client/Server Backbone

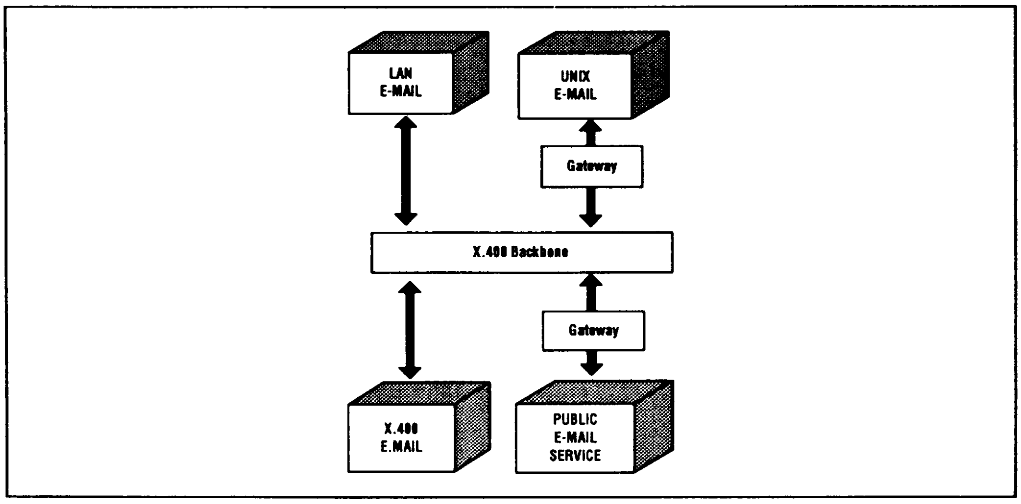

Illustration 10. This level allows clients to connect directly to the backbone services.

Currently, systems such as this use a proprietary protocol to deliver the message between the client and the standards-based services. However, this architecture is designed to migrate to a standards-based approach in both ends of the process. First, the calls from the client will be made in a "standard" format, using the desktop APIs such as MAPI and VIM. Second, the X.400 systems are evolving to support P7 calls so that they can be used directly as a client/server message store. Then mail clients will be able to make their calls via P7. Additionally, we can expect some MAPI and VIM conversions to P7. Both of these efforts will allow additional mix-and-match options for the user.

The greatest advantage of this system is the elimination of gateways. With the same services being used by multiple clients, the fidelity of messaging is assured and requirements for services such as directory synchronization disappear. However, this works only for those systems connected to the backbone. For full interoperability, one more step is required.

INTEGRATED STANDARDS-BASED MAIL SYSTEM. The final stage is the fully integrated standards-based mail system. Here, all clients communicate with their services via standards-based protocols. All services also communicate with each other via standards-based protocols, thus forming one virtual system. (See Illustration 11.) Naming and directory services are now handled on a federated basis—directory synchronization is not required—so users can query across systems on a hierarchical basis.

Integrated Standards-Based E-Mail

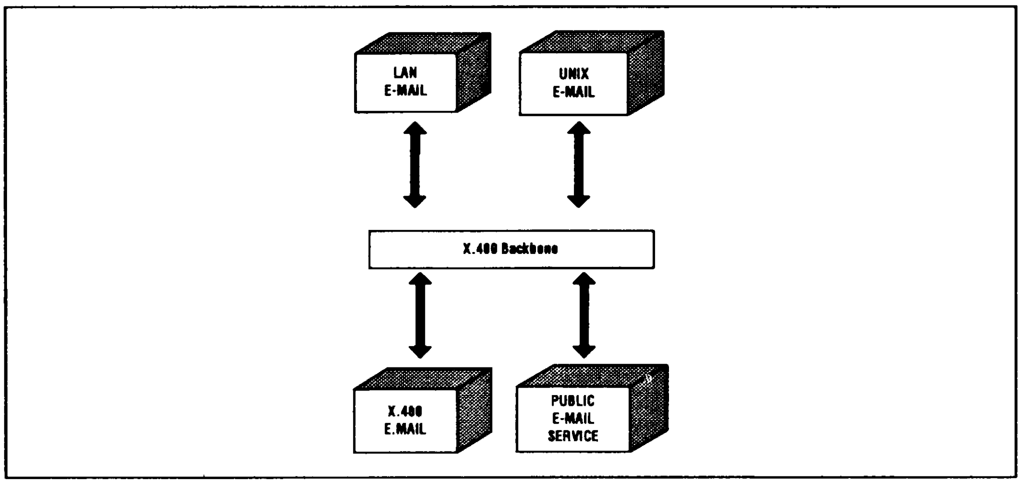

Illustration 11. In a fully standards-based system, all clients can reach all services without using gateways.

This system is only a promise. However, the promise may just be compelling enough for users to force the mail vendors to make it a reality.

What Can Be Done Today?

Plan for the Evolution

The first step is to look at today's system and see how it can be moved closer to where the architectures are moving. A gateway approach should be migrated to a backbone approach, and standards-based backbones should be considered for these backbones, as well as for existing proprietary backbones. Here, we recommend X.400, since all the vendors and most large customers are moving in that direction. Robust implementations of X.400 are becoming more and more available on an almost-daily basis. Vendors offering X.400-based products include: Digital, Hewlett-Packard, NCR Corporation (Dayton, Ohio), Groupe Bull (Paris, France), Data General Corporation (Westboro, Massachusetts), Soft*Switch, OSIWare (Vancouver, British Columbia), Retix (Santa Monica, California), ISOCOR (Los Angeles, California), Worldtalk (Los Gatos, California), Enterprise Solutions (Westlake Village, California), Alisa Systems (Pasadena, California), NeXor (Nottingham, England), ISODE Consortium (London, England), and many others.

What Can Be Done Today?

You should make new purchases of mail systems with an eye toward your architectural goals. If support for specific desktop APIs is high on the list, then one set of options presents itself. If the availability of an X.400 MTA from the LAN-based mail provider is important, then the choices today are more limited, though both Microsoft and Lotus have committed to providing X.400 back ends for their LAN mail products.

Conclusion—Is Standards-Based Mail Real?

Standards Are Becoming the *Only* Basis for Mail Interoperability

It is clear that we are not there yet. Most companies still have a maze of gateways through which messages try to find their way. The exchange of rich content across systems is uncertain at best. And real mail-enabled applications are in their infancy.

Yet it is equally clear that taking the road the industry is following, the road that customers are forcing their vendors to travel, the only logical road, is that of achieving the promise of ubiquitous, rich electronic messaging through standard protocols and APIs.

The question is no longer **if** standards are the answer. The only question is **when**. That when is not too far off, and the pace of getting there is picking up.

Next month's *Open Information Systems* will address
SAP, Europe's Software Giant.

For reprint information on articles appearing in this issue,
please contact Donald Baillargeon at (617) 742-5200, extension 117.

Open Systems: Analysis, Issues, & Opinions

FOCUS: INDUSTRY CONSORTIA

Open Software Foundation at Age Five

The Open Software Foundation (OSF) celebrated the fifth anniversary of its founding with a birthday party at the Massachusetts State House and a challenge event with demonstrations of applications built on the Distributing Computing Environment (DCE), arguably its most significant contribution to the industry to date. The potential value to customers of DCE and Distributed Management Environment (DME) is immeasurable. However, the roles of OSF's other technologies—Motif, OSF/1, OSF/1 microkernel (MK), and Architecture Neutral Distribution Format (ANDF)—are less clear. Over its first five years, OSF has begun laying the foundation for tomorrow's open distributed computing. What and how it builds on that foundation will be watched closely by vendors and users alike.

One of the important reasons for the OSF to celebrate at its birthday party was that a funding plan for the next two years has been put into place. This plan assumes that the OSF's license revenues will continue to increase over that period, requiring lower contributions from members. While a seven-year break-even point would be unacceptable at an ordinary software startup, comparing the OSF to a software startup is inaccurate and unfair. Process is almost as important as product at the OSF, and its vendor-neutral process takes time to follow. Also, as a technology supplier to vendors and not end users, OSF's range of options available for raising revenue is limited.

Challenge '93

Challenge '93 was divided into three sections: OSF/Motif and applications, OSF/1 implementations, and DCE implementations/ applications. Twenty-five vendors demonstrated interoperability or portability using OSF technologies on more than 80 platforms ranging from PCs to mainframes.

The entry criterion was that products shown had to be either available or announced for general availability in 1993. The OSF verified that products met this criterion through interoperability testing prior to the Challenge event. Vendors demonstrating DCE implementations went though testing at an Interoperability Festival (I-FEST) in the month prior to Challenge. I-FEST consisted of a battery of DCE quality assurance tests that required each platform to operate as both a client and a server. Each platform also had to interoperate with at least two other platforms.

CHALLENGE DEMONSTRATIONS. The DCE demonstrations included:

- The University of Michigan's (Ann Arbor) Center for Information Technology Integration (CITI) Demographics Demonstration Program (DDP). DDP uses DCE services to provide information about a selected geographical area from services distributed across servers throughout a network. DDP makes use of DCE's Threads, Cell Directory Services, Security Services, and Remote Procedure Call (RPC).

- Sybase Incorporated (Emeryville, California) demonstrated a prototype of its Client/Server Architecture based on DCE, using RPC, Directory, Security, and encryption. The demonstration showed how a developer could use standard SQL and the Sybase application programming interface (API) to transparently use DCE services without having to rewrite and recompile applications in the DCE environment when the format of the data changes.

- Oracle Corporation (Redwood Shores, California) had a number of demonstrations showing how Oracle tools use DCE services to connect to Oracle7 databases. Cell directory services were used to locate the appropriate database, and threads were used on the server to handle RPC calls and as the client/server transport method.

- The OSF Distributed File System (DFS) was demonstrated running across implementations from Transarc Corporation (Pittsburgh, Pennsylvania), IBM (Armonk, New York), Digital Equipment Corporation (Maynard, Massachusetts), and Hewlett-Packard Company (Palo Alto, California). The operation and management of DFS are based upon the DCE RPC, and the demonstration showed how DFS provides a unified, globally distributed file space with DFS files available from any DCE machine.

- The Volpe National Transportation Center demonstrated the Enhanced Traffic Management System (ETMS) developed for the Federal Aviation Administration (FAA). This system lets air traffic controllers access data on traffic and flying conditions throughout U.S. air space. It compiles real-time flight and weather data and presents them in an easy-to-use format. ETMS runs on systems from HP, IBM, and Digital using DCE services.
- Mitre Corporation (Bedford, Massachusetts) demonstrated several prototype U.S. Army distributed Command and Control applications that make use of DCE secure core services, including Threads, CDS, RPC, Distributed Time Service (DTS), and Security, to access information on DCE servers.

There were also examples of DCE applications being shown from Charles Schwab, Citibank, Owens Corning, Paris Metro RATP, and Radio Paging Service. Hewlett-Packard, Digital, and Gradient Technologies Inc. (Hudson, Massachusetts) all demonstrated their DCE development tools.

DCE Release Timetable

DCE Release 1.0.2 was delivered to licensees in May. It consisted primarily of bug fixes, but security replication was added as a major feature enhancement. DFS was enhanced with advanced file set operations, enabling operations to be carried out simultaneously on blocks and groups of files on a distributed basis. DCE Release 1.0.3 is scheduled for delivery at the end of 1993. It will contain features from Release 1.1, such as Internationalization, that are not dependent on other components of Release 1.1 and that will be ready before the mid-1994 scheduled delivery of Release 1.1 One of the key objectives of Release 1.1 is to improve the maintainability of the code by building in instrumentation and auditing capabilities.

Motif: Competing with Volume

The Common Open Software Environment's (COSE's) standardization on Motif for the Common Desktop Environment formalized its position as the standard Unix graphical user interface (GUI). However, Unix still has to compete with Microsoft, and Motif is outsold 20:1 by various incarnations of Microsoft Windows. Motif will never supplant Windows as the industry's dominant interface, so the OSF must concentrate on strengthening its position in its workstation niches to withstand the potential onslaught of Windows NT. As long as the underlying X Window System lags Windows in support of critical user environment capabilities, like Dynamic Data Exchange (DDE) and Object Linking and Embedding (OLE), X Window is threatened by NT. The best hope for the X world to successfully stave off complete dominance of the desktop by Microsoft is to take its network heritage to heart and leapfrog Microsoft with a network-based but OLE-compatible implementation.

Although the OSF does not wholly control its own fate in this area, it is working to enhance Motif beyond the X standard. It can add functionality to Motif in some areas, but it must largely stick to the work of the X Consortium and concentrate on rapid implementation of the upcoming Release 6 of X11.

OSF/1: Kernels Aren't Relevant

Although the original rallying point for the formation of the OSF was the creation of a vendor-neutral, Unix-compatible operating system that would, eventually, be unencumbered by AT&T licenses, we believe that this mission is less important today than it was in 1988. True, the OSF recognizes a tidy revenue stream from OSF/1 (approximately one-third of its revenue), but it is not clear to us that, in the long term, operating system development is the best use of OSF resources. Customers are looking to the OSF not as a provider of operating system technology, but as a provider of critical technologies for distributed applications and management.

In the meantime, the OSF formally announced that it has begun working on the convergence of the current OSF/1 technology with the OSF/1 MK technology being developed by OSF's Research Institute. The result of this work will be the next version of OSF/1, Release 1.3, available in mid-1994. This operating system will be based on Carnegie Mellon University's (Pittsburgh, Pennsylvania) Mach 3 microkernel technology and will be used by IBM and others to support multiple operating system personalities. It will have facilities to support real-time applications and will also support B3 level security. An important part of the work of the Research Institute has been the industrialization of Mach 3 through the definition of a stable set of features and interfaces. Maintainability of the code has been another important priority.

A key issue for future generations of OSF/1 technology is whether the OSF is the right vehicle for its continued evolution. The ability of OSF/1 MK to support multiple sets of operating system services, commands, and libraries on a single platform makes it potentially valuable for the time when users will expect each platform to support multiple execution environments. We believe that the OSF should consider contracting an outside source to continue OSF/1 MK development and clear its own plate of an issue that has generally been a public relations disaster for the consortium.

ANDF: Compelling Idea—Hard to Implement

The industry needs a technology like ANDF. Both users and software developers would benefit greatly from the distribution of packaged software in an architecturally neutral format that protects intellectual property while simplifying the distribution, purchase, and installation of applications. Between Windows NT and the work of COSE, the requirement for a consistent execution environment is rapidly becoming a reality. Over the next few months, we anticipate that the OSF will be announcing some important advancements in ANDF technology, support, and licensing. After that, the task will be to create an atmosphere in the industry that ANDF is a standard method of software distribution and an understanding on the part of users that ANDF is the best way to distribute software.

DME: Someday

Of all of the areas the OSF has been involved in, none is more critical but has met with more resistance than the Distributed Management Environment. The concept of a unified system and network management framework for mixed environments is high on the priority list of virtually all users. Yet, because of the strategic importance of management and the close association of management and platform, vendors have been strong to resist surrendering proprietary frameworks and applications. While all vendors have pledged to converge on DME compliance, the vagueness of these commitments are only exceeded by their commitment to Object Management Group (OMG) Common Object Request Broker Architecture (CORBA) compliance.

OSF: The Next Five Years

The OSF must face many challenges successfully if there is to be a 10-year celebration. The foremost challenge is continuing to supply value to the industry in order to justify its existence to both its vendor sponsors and to users. From our perspective, the OSF must:

- Continue to advance the technological foundations for open, distributed computing
- Continue to act in ways that do not favor any one vendor or group of vendors
- Continue to avoid the pitfall of being thrust into the role of carrying out unprofitable development activities to meet special interests
- Avoid entering into competition with for-profit corporations.

If not met head-on, any one of these challenges could lead to a devaluing of the OSF and its activities by its members and by its ultimate franchise, the customers.

—M. Goulde

The Only Comprehensive Guide to Relational DBMSs for Unix Covering:

•

Oracle

•

Sybase

•

Informix Software

•

Ask/Ingres

•

Progress Software

•

Borland/Interbase

For More Information, Call Donald Baillargeon at (617) 742-5200 ext. 117

New In-Depth Research Report from Patricia Seybold Group

Unix Relational Database Management *Third Edition*

Vendor Strategies, DBMSs, and Applications Development Tools

By Judith R. Davis

Full Report, Third Edition, Available July, 1993	**$595**
June 93 Report plus Four Quarterly Comparison Matrices—starting September 1993	**$795**
Four Quarterly Comparison Matrices starting July 1993 *or* starting September 1993	**$595**
One Comparison Matrix of Your Choice July 93, Sept. 93, Dec. 93, March 94, June 94	**$200**

The third version of this well-respected and popular report continues to cover the product and marketing strategies of the major Unix Relational database vendors. In addition, this In-Depth Report includes detailed chapters for each vendor on database server architecture and functionality, and on tools available for developers, end users, and system administrators. The final chapter provides a reality check on overall strengths and weaknesses of each vendor from the perspective of the experienced professional developer. The report also includes a comprehensive comparison matrix of the features, functions, and architectural characteristics of each product. The matrix includes such categories as: computing environment, server architecture, database parameters, intelligent server features, security, distributed database, interoperability, tools, and more.

Report Highlights:

Informix: Focusing on Database Application Technology. Informix is positioning its OnLine database engine and its comprehensive development tools as the ideal foundation for the development, deployment, and evolution of database applications. Planned are OnLine enhancements, a graphical facelift for the company's venerable 4GL products, and sophisticated end-user tools. Will these enable Informix to compete in the open client/server database market?

Ingres: Can it Develop a Successful Marketing Strategy? In spite of good technology—especially in its state-of-the-art graphical development tool, Ingres/Windows 4 GL, users still feel that Ingres has a continuing inability to market its technology effectively. What are Ingres and Ask together doing to regain market visibility for Ingres, and to keep Ingres ahead on the technology curve?

InterBase: Can Borland Be a Player in the RDBMS Server Market? InterBase has some impressive RDBMS server technology but has weak tools. Borland, on the other hand, has been phenomenally successful selling shrink-wrapped PC applications for the desktop. Is Borland's acquisition of Ashton-Tate/ Interbase Software a marriage made in heaven, or a serious mismatch of product architectures, marketing strategies, and support requirements?

Oracle: Can Oracle7 and a New Tools Strategy Keep Oracle on Top? Oracle has made sweeping technology enhancements in its entire product line over the past year. The Oracle7 database server implements a significant number of new features and functions, and Oracle's new suite of tools will finally roll out this summer. Like Sybase, Oracle has also set its sights on the entire client/server infrastructure. Will these moves enable Oracle to maintain its dominance in the RDBMS market?

Progress: Counting on Version 7 to Keep It in the Race. Progress still has an impressive RDBMS application development environment. But the company is feeling the competitive heat to provide both a graphical development environment and to match the competition in the server features war with options like stored procedures, triggers, and standard SQL compliance. Will Version 7 do the trick?

Sybase: Can Its Open Client/Server Architecture Succeed? Sybase is clearly positioning itself as a client/ server company. It is aggressively addressing difficult client/server issues with its new System 10 product line, and intends to bolster its lagging development tools in the future. Does System 10 give Sybase the ammunition it needs to fight off and surpass archrival Oracle?

Receive the information in this report in a flexible and timely manner that suits a variety of database research needs:

- **In June 93 receive the Third Edition RDBMS Report which includes the latest product matrix.**
- **Receive just the Product Comparison Matrix.** The matrix information is updated on a quarterly basis. Buy one matrix or choose a set of 4 quarterly matrices from June 93 through June 94.
- **Receive the Full Report in June 93 *and* 4 Quarterly Matrix Updates from Sept. 93-June 94.**

Patricia Seybold's Computer Industry Reports Order Form

Please start my subscription to:		U.S.A.	Canada	Foreign
☐ *Workgroup Computing Report*	12 issues per year	$385	$397	$409
☐ *Open Information Systems*	12 issues per year	$495	$507	$519
☐ *Distributed Computing Monitor*	12 issues per year	$495	$507	$519

Please send me a sample of: ☐ *Distributed Computing Monitor* ☐ *Open Information Systems* ☐ *Workgroup Computing Report* ☐ *Paradigm Shift—Case Studies in Distributed Computing*

Please send me information on: ☐ Consulting ☐ In-Depth Research Reports ☐ Conferences

Total $________ ☐ **My check for $**________ **is enclosed.** ☐ **Please bill me.** ☐ **Please charge my subscription to: Mastercard/Visa/American Express** (circle one)

Name: ______________________ Title: ______________________

Company Name: ______________________ Dept.: ______________________

Address: ______________________

City, State, Zip code, Country: ______________________

Fax No.: ______________________ Bus. Tel. No.: ______________________

Card #: ______________________

Exp. Date: ______________________

Signature: ______________________

Checks from Canada and elsewhere outside the United States should be made payable in U.S. dollars. You may transfer funds directly to our bank: Shawmut Bank of Boston, State Street Branch, Boston, MA 02109, into the account of Patricia Seybold Group, account number 20-093-118-6. Please be sure to identify the name of the subscriber and nature of the order if funds are transferred bank-to-bank. IO-693

Send to: Patricia Seybold Group: 148 State Street, 7th Floor, Boston MA 02109; FAX: (617) 742-1028; Phone: (617) 742-5200; MCI Mail: PSOCG

1992 and 1993 Feature Reports from Patricia Seybold Group
To Order these Back Issues, Fax (617) 742-1028, or Call (617) 742-5200.

Office Computing Report

Volume 15 issues—$40

- ☐ 6/92 Apple's Macintosh—Can It Become "the Cadillac of Collaboration"?
- ☐ 7/92 Business Intelligence—A Framework for Data Analysis Applications
- ☐ 8/92 The Quest for Common Mail APIs—Clearing up the Confusion
- ☐ 9/92 BeyondMail for Windows—Epitomizing the Mail-Enabled Application
- ☐ 10/92 Microsoft's Workgroup Strategy—Moving Group Functionality into Windows
- ☐ 11/92 Visual Programming—Application Design for End Users and Professional Developers
- ☐ 12/92 The Notes Phenomenon—The Industry Reacts to Lotus Notes

Volume 16 issues—$50

- ☐ 1/93 Microsoft Access—"Cirrus" Database Project Gets down to Earth

Workgroup Computing Report

- ☐ 2/93 WordPerfect Information Systems Environment—WordPerfect Reveals Its Blueprint for Workgroup Support
- ☐ 3/93 Lotus Notes Release 3—Extending the Notes Paradigm
- ☐ 4/93 Can Windows NT Meet the Challenge?—Microsoft's Next Generation Operating System Stirs the Industry
- ☐ 5/93 Action Technologies' Workflow Products—Coordinating the Activiteis of People as They Work Together

Back Issue Total $________

Distributed Computing Monitor

Volume 7 issues—$50

- ☐ 6/92 Distributed Printing—Major New Approaches Begin to Relieve One of Distributed Computing's Most Frustrating Problems
- ☐ 7/92 The New E-Mail APIs—Finally, Real Progress toward Mail-Enabled Applications
- ☐ 8/92 Distributed Object Computing—The Merger of Distributed Computing and Object-Orientation into a New Architecture
- ☐ 9/92 Gradient DCE for PCs—PC-DCE Spurs OSF to Make PCs Peers to Unix and Other More Powerful Platforms
- ☐ 10/92 Database Interoperability—A Comprehensive Approach to Database Access for the 1990s
- ☐ 11/92 Transarc Encina—Will Distributed OLTP Systems Overrun the Mainframe's Last Stronghold?
- ☐ 12/92 UI-Atlas Distributed Management—Object-Oriented, Distributed Management for the Unix System V World

Volume 8 issues—$50

- ☐ 1/93 Component Software—A Market Perspective on the Coming Revolution on Solutions Development
- ☐ 2/93 Switched Internets—The Coming Gigabit Revolution in Enterprise Networking
- ☐ 3/93 IBM's System Object Model—Cornerstone of an Open Distributed Object Computing Environment
- ☐ 4/93 Encapsulating Databases—Practical Uses of Object Technology to Inprove the Value of Relational Data
- ☐ 5/93 OMG's CORBA 2.0—Industrial Grade Standard for Distributed Object Computing?

Back Issue Total $________

Printed on recycled paper.