Patricia Seybold's

VOL. 4, NO. 8
ISSN: 0887-3054
AUGUST 1989

UNIX IN THE OFFICE

PRODUCTS • TRENDS • ISSUES • ANALYSIS

INSIDE

EDITORIAL

Office as a Utility ....Page 2
Vendors will stop selling Office as a set of applications. Rather, traditional office functionality will become a set of utilities underlying vertical software.

NEWS ANALYSIS

A comparative look at **Digital's RISC and VAX/VMS processors** • **WordPerfect Office moves from the LAN to Unix** • Report from **Xhibition '89**: networking, graphics, standards, and other issues • The RISC market offers a multitude of choices from **Motorola, Sun, Intel, MIPS Computer, Hewlett-Packard, and IBM** ....Page 14

Graphical User Interfaces

A Developer's Quandary

By Laure Brown

NO DOUBT YOU'VE been inundated lately with information on graphical user interfaces: the picks, the perspectives, the right graphical metaphor, the comparisons and contrasts, the distinction between look and feel, the standards, the lawsuits. Admittedly, we've contributed to that inundation. However, behind all the discussions about intuitive displays and color schemes and menu *(continued on page 3)*

• E D I T O R I A L •

The Brave New World of Office

By Judith S. Hurwitz

FOR MORE than three years, we have been explaining where the Unix office market and its vendors are headed. Sometimes, we have been impressed with the innovation we have seen. Other times, we have chided these vendors for going for the lowest common denominator. We have gotten used to thinking of Unix office applications as a set of vertical applications sort of like accounting packages. But watch this space, folks. It's all about to change.

Not that you'll wake up one morning to find that all the familiar and well-loved office software products are gone (heaven forbid!). No, the transition will be much more subtle than that. The first signs of change can actually be seen in IBM's OfficeVision product, announced a few months back. Rather than position OfficeVision as a vertical package, IBM positioned the functionality within OfficeVision (mail, calendar, etc.) as underlying utilities that could be integrated with third-party application packages.

So what does this have to do with Unix? A lot. In the coming 18 months, we expect vendors such as Hewlett-Packard, Data General, IBM, and others to begin announcing and delivering their next-generation office products for the Unix platforms. Digital has already begun to offer a rich set of office tools (mail, calendar, compound document editor) for Ultrix under its DECwindows system. The one characteristic that these products will have in common is that office functionality will become an underlying service. What does this mean? Well, if I am an accountant and want to send a monthly report, I simply hit the Mail key from within the accounting application. If the report must be approved by three managers, I can set up a procedure that sends it to each manager, who, in turn, adds an electronic signature. The idea is that individuals can work in their native environments and still have full use of the office functionality that is part of the system.

There is another interesting twist to this story. Traditionally successful Unix office vendors have software that runs on a wide variety of platforms, and they use this general availability as a key selling point. They have been successful because there has not been a flood of available products to compete with them. They probably still feel secure. No matter how good Hewlett-Packard's or Data General's new office software might be, it will be proprietary, right? Wrong. We expect that these vendors will play the Unix software game and freely license their new office software to any vendor or user who wants to buy. If this scenario unfolds, it will have a profound impact on the Unix office software marketplace. The winners will be the users, who will have a lot more good applications to choose from. But there may well be losers. If vendors currently selling Unix office software rest on their installed bases and do not change fast enough, they could find their comfortable positions in the marketplace eroding slowly over time.

The drama that will be unfolding in the Unix office software arena is a microcosm of the entire Unix marketplace. Unix is becoming a hot and valuable market. The powerful hardware vendors from the proprietary world are moving in fast, causing the traditional hot box vendors to find new niches and technological innovations to distinguish themselves.

The '90s will be an interesting decade. Hardware will be presumed to be fast and cheap. Software will be the focus of our attention. We expect that, at the same time, a new definition of office will emerge. Office will be the stuff that makes vertical commercial applications work. In the not-too-distant future, office will be viewed as an underlying technology, as networking is today. It will be a challenging and exciting transition, and Unix will be at the forefront. ◐

Patricia Seybold's Office Computing Group

Publisher PATRICIA B. SEYBOLD
Editor-in-Chief JUDITH S. HURWITZ

Managing Editor
RONNI T. MARSHAK

Senior Editors
MICHAEL D. MILLIKIN
JUDITH R. DAVIS
Telephone: (617) 861-3926

Associate Editors
JOHN R. RYMER
LAURE BROWN

News Editor
DAVID S. MARSHAK

Contributing Editor
GARY J. NUTT
Telephone: (303) 444-0341

Sales Director
RICHARD ALLSBROOK JR.

Circulation Manager
DEBORAH A. HAY

Customer Service Manager
MICHELLE MANN

148 State Street, Suite 612, Boston, Massachusetts 02109 Telephone: (617) 742-5200 FAX: (617) 742-1028

Patricia Seybold's UNIX in the Office (ISSN 0887-3054) is published monthly for $495 (US), $507 (Canada), and $519 (Foreign) per year by Patricia Seybold's Office Computing Group, 148 State Street, Suite 612, Boston, MA 02109. Application to mail at second-class postage rates is pending at Boston, MA and additional mailing offices. POSTMASTER: Send address changes to *Patricia Seybold's UNIX in the Office*, 148 State Street, Suite 612, Boston, MA 02109.

•GRAPHICAL USER INTERFACES•

(continued from page 1) implementations sit bewildered software developers staring at an unsettled world of multiple user interfaces. So, although we considered taking on a discussion of the main Unix user interface contenders for this article, the developer's dilemma seems a more pressing issue.

The heart of the problem is portability. Graphical user interfaces (GUIs) will critically impact next-generation computing environments, especially distributed computing and object orientation. But with so many GUI options out there, developers are stymied. There's X and its many flavors; there's PM (and eventually PM/X); there's NextStep; there's Macintosh; there's Metaphor; there's NewWave. They're all contenders in the GUI game, and they share no common application programming interfaces (APIs). You have to completely rewrite an application to run it on each platform. Which do you write to? All of them? Not quite, although that's the ideal option. Writing GUI applications is tricky—very tricky—and it's all the more so if you're a developer coming from a text-based DOS world. And the cost of re-porting an application from system to system is simply too expensive.

The heart of the problem is portability. GUIs will critically impact next-generation computing environments, especially distributed computing and object orientation.

Solutions? Few. Neuron Data has come up with one: the Open Interface Toolbox. However, it's only used internally by Neuron Data. A similar tool is available from Advanced Programming Institute. Both technologies are worth looking at, so we'll do that, but not without first putting the developer's position into better perspective. We'll begin by painting in the background of GUIs: the benefits and the motivations behind them. Then we'll zero in on windowing programming environments: the levels of complexity involved, the dominant window systems, and the portability issues.

More Than Meets the Eye

Graphical user interfaces are more than a slicker, easier way to interact with computers. They offer significant advantages to users over character user interfaces (CUIs), including:

- Non-modal operation. An application doesn't take over your entire screen.
- Direct manipulation of information on screen.
- Task orientation (as opposed to tool orientation). Rather than worrying about what application you need to load and how to work with it to get at those budget figures, for example, you grab the budget icon and run.
- Easy data-sharing between applications.
- Most GUIs run on larger displays—more real estate for viewing multitasking applications.
- More consistency across applications and platforms. This is where a lot of the standardization issues come into play. Consistent behavior (the "hows"—things like moving windows, changing icons, scrollbar movements) across platforms is somewhat more critical than a consistent appearance. As long as you know how to manipulate GUI components, it doesn't really matter how they look on screen.
- Transparency among applications.
- Ease of learning—with a caveat. Too many graphical user interfaces with different interface principles will only confuse users.
- Coprocessing support. With a windowing environment, a GUI can give you the proper front-end tools as well as a view to the back-end processors.

A number of changes in the marketplace are driving the transition to graphical user interfaces. As businesses continue to thrive on change and globalization, getting information from a number of sources quickly, the computer industry is moving from standalone, single-tasking, proprietary, disposable solutions to networked, multitasking, integrated, multivendor, scalable solutions. Clearly, distributed network computing (DNC) and object orientation have roles here.

But do GUIs? Yes, in that they support DNC and object orientation. Graphical user interfaces are inherent in object-oriented environments. Consider, as an obvious example, compound document/object architectures. You sure can't see a raster on a character-based screen. In DNC, a distributed windowing environment, such as X-Window, is in order. Thus, you can open a window and manipulate an application that lives on another machine on the network.

Although GUIs have much to contribute to the commercial users, right now, the market remains in scientific and research communities, where workstations prevail. No doubt, that has something to do with the fact that innovative developments often come out of these environments. Of course, the work of many scientists and researchers demands investments in new technology.

For the typical commercial user working with 1-2-3 all day long, the investment doesn't seem quite as appropriate. His computer most likely doesn't support GUIs, and the ones that do are pricey. Be that as it may, everyone agrees that future interfaces will be graphical. And that's the not-too-distant future. Because next-generation computing is upping the user interface ante, the commercial world has taken a marked interest in windowing environments during the past year.

X-Window: Just a Review

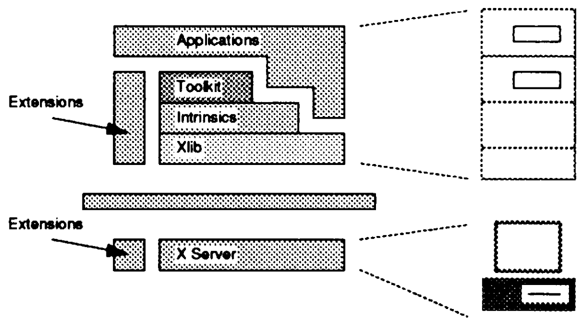
BY NOW, you're probably familiar with X, so we'll just give you a quick run-through of the X fundamentals. (For an in-depth look, see Vol. 3, No. 10.)

Client/Server Model. X is client/server based, but it's backwards. The client is the machine that's actually running the application, and the server is the workstation that's providing display services. In other words, the X server runs on what most people generally consider a client: the workstation.

Windows. X has the familiar hierarchical, layered window structure that allows windows within windows within windows...(you get the picture). The main workstation window, called the root window, displays the background. Its children are shell windows and are associated with a particular application.

Architecture. X-Window is made up of four main components (see illustration):

- The X server is the workstation program that the user interacts with; it creates the user interface, displays graphics, etc.
- The X protocol defines the data structures that pass requests between clients and servers. It's independent of operating system, network transport, and application language, thus letting users access applications in a typical smorgasbord network.
- The X library (Xlib) is X's interface to the X protocol. It is represented by the bottom layer in our model, translating messages between client and server into protocol requests and providing very primitive graphics primitives.
- The vanilla Xtoolkit (Xtk) has a full set of intrinsics, but only a few widgets. The range of widget sets fluctuates from toolkit implementation to toolkit implementation.



X architecture.

Windowing Environments

While GUIs may be helping users reach a higher computing plateau, they're giving developers a tough time. Graphical systems are relatively new, and writing to a them is a whole new ball game for most programmers. Actually, few even use them. Today, most Unix programmers—like users—are still sitting at character-based terminals. GUI and CUI (character user interface) programming models are worlds apart. Developers have their hands full just migrating traditional character-based applications, let alone new applications. Furthermore, because they're richer, GUI environments are more complex than CUI environments. You have to worry about things like graphics display, multifonts, and interapplication connections between windows. Writing to a bit-mapped screen is a much more exacting exercise than is writing to a character-based screen. Lots more coding is required. Hunting down a bug can take relentless effort.

A REFERENCE MODEL. From a programmer's perspective, a generic, X-based, graphical user interface can be made up of four programmatic interface layers, which are hierarchical in terms of coding complexity: the base window system interface, the toolkit, the window manager (a.k.a. the presentation or style guide), and the user interface management system (UIMS). As you move up through the layers, GUI programming gets less convoluted. Writing applications at the window system interface will put you down in the guts of the GUI, whereas, with a UIMS, programming is simpler.

So, you ask, if complexity is such a big deal, why not simply use the UIMS instead of mucking around at a lower layer? Basically because (as we'll point out below) UIMSs are, by-and-large, still immature. Some GUIs don't offer UIMS capabilities at all (see illustration, page 6). Furthermore, each layer depends on what's underneath it. The window manager, for instance, relies on components built by toolkit programmers. So it comes down to a question of programming expertise. Generally speaking, the person fiddling with toolkit mechanisms to create GUI components is not the same person who uses the components to create an interface. And the person who creates the interface is not going to be the one who writes the application code. And none of them will be the user who wants to configure existing applications and interfaces to meet his or her whim. The programmatic interface depends entirely on aptitude. If you have your hands on a window system whiz—a rare commodity—you're not going to waste that talent on interface design; you're going to have him or her programming toolkit components.

Moreover, you may write at the lower layers to keep more control over the look and feel of the application at hand. The deeper you are in the layers, the less confined you are by the style conventions and functionality of the GUI.

Base Window System Interface. The lowest rung on the GUI programming ladder is the base window system interface, which converts messages between the application and the display device. Think of it as the middleman between the application and display. When an application makes a window management call, the base window interface changes the call into a format that the display can interpret. It then changes event messages (e.g., mouse clicks or window changes) into values for the application.

This interface provides only minimal window operations such as primitive graphic functions, print string, and I/O events. Therefore (as mentioned), writing applications at this level is a painstaking process.

Toolkit. A less demanding interface is the toolkit, a set of development resources that makes life easier for programmers. The toolkit itself is made up of two layers: intrinsics and widgets. Intrinsics are the basic building blocks of the toolkit—the toolkit's toolkit, so to speak—which are used to create widgets. The toolkit spares programmers from the coding drudgery involved at the base window system interface. They don't have to write their own routines to generate things like menus, pop-up panels, and windows.

Window Manager. The window manager is where appearance and behavior enter the picture. Its guidelines specify the style of user interaction: where functions should be placed on screen and how features such as pull-down menus should be implemented.

UIMS. The industry is still pretty hazy about user interface management systems (UIMSs). Ask 10 different people what a UIMS is, get 10 different answers—anything from a screen painter to an interface layout tool to an interface customization facility. A more clear-cut definition needs to be found—what belongs in a UIMS, what doesn't, and evaluation criteria. In the meantime, we'll give you a working definition:

- Paint, not code. A UIMS lends the developer some GUI advantages. Consider it a graphical programming interface. Whereas toolkit programmers need an intimate relationship with widget design and intrinsics, a UIMS spits out code as you paint. (However, the functions of toolkits and UIMSs are blurring as toolkit implementations pick up UIMS capabili-

UI Glossary of Terms

Server. The display device.

Library. In X terminology, the base window system interface is the X library (Xlib), which consists of primitive graphics routines (e.g., for window functions and graphics resource functions).

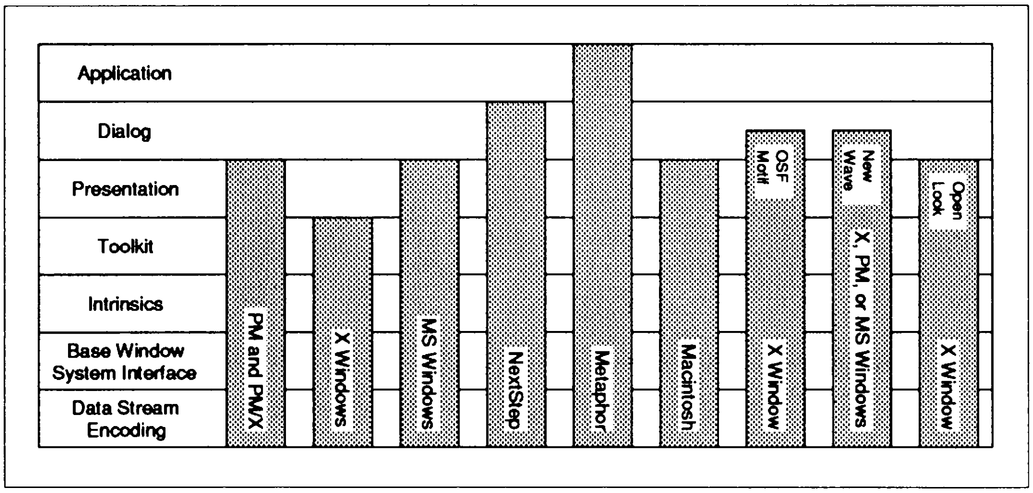
Intrinsics. Intrinsics are low-level user interface development tools. They are the general utility routines for implementing code and work in conjunction with higher-level, screen-based components called widgets.

Widgets. Widgets, basically, are specialized windows with I/O capabilities. They provide graphical interactive components like scrollbars, dialog boxes, text fields, and such. Widget sets can also include more sophisticated components such as multifont text and rasters.

Gadgets. Gadgets are essentially widgets, but are more limited functionally. However, they give applications better performance by creating fewer windows and using less memory. Gadgets include labels, push buttons, toggle buttons. You use them just the way you use widgets, with similar programmatic access. They are just streamlined implementations.

Application	*Application Specification*
Dialog	*User Interface Management System*
Presentation	*Style Guide (Look and Feel)*
Toolkit	*Function calls using Intrinsics*
Intrinsics	*Function calls using Interface*
Base Window System Interface	*Function calls*
Data Stream Encoding	*Data structures*

GUI Components per the National Institute of Standards and Technology (NIST) draft reference model (which is based on X-Window).



GUI capabilities. The various environments mapped against the layered GUI model. Few adequately address the UIMS issue.

ties, and UIMS tools incorporate traditional toolkit characteristics—like widget-building.)

- Dialog management. A UIMS deals only with interaction between the user and the application; it doesn't touch application code. To that end, there needs to be a clear split between function and interface. Why? Because we're looking for a more modular way to design user interfaces—a plug-and-play solution. It should be possible to borrow interface functions from other applications. The split also makes sense in terms of performance. Toolkits are huge and, to keep things from being unbearably slow, they should be stored locally (in X terms, on the server).

- Portability. Ideally, you should just be able to say, "Hey, I want this to be a NextStep application," pick NextStep from a menu, and the UIMS will bang out the right code and style conventions. However, the modular approach we hope for doesn't lend itself well to multiple systems. Unless the developer is very familiar with each target platform (and, with the newness of GUI systems, this is just not the case), modules will carry nonportable code of the interface system that the developer knows best. Currently, most tools are strapped to a single window manager. This fact also stifles creativity, because you're locked into that window manager's look and feel.

- Testing, prototyping, and evaluation tools. A UIMS can execute a user interface (UI) without it being tied to the application. In other words, you can test and refine the behavior of the UI before actually binding it to the program. And we don't just mean testing the way it looks and interacts with the user. It's also essential that a UIMS spot any hidden domino effects. Just deleting a button might cause a ripple of unforeseen repercussions and screw up the whole application.

The UIMS to Come. Today, UIMSs are programmer's tools, and that's too bad. Interface designers are just that—designers, not programmers. They know about the cosmetics, the composition, and the psychological implications of user interface design. UIMSs should be geared for them. The current UIMS tools still demand some knowledge of widgets and toolkits. Refinement is in order.

End-user UIMSs aren't out of the question, either. Eventually, users should be able to customize an existing application for the way they want to interact with it. Or they can design their own applications, or overlap them, or reuse functions they like from other programs and libraries.

Standard Foundations: X-Window and Presentation Manager (...and NextStep?)

Though plenty of GUI options exist, the industry is hovering around two window systems: X-Window and OS/2's Presentation Manager (PM).

From a Unix perspective, we're obviously most interested in Presentation Manager's influence on Unix user interfaces. Because of its MS Windows heritage and its IBM SAA (Systems Application Architecture) affiliations, PM is setting the user interface tone in the office. (IBM was rather quiet in its recent SAA announcement about how OfficeVision would fit into its AIX strategy. There isn't an easy short-term answer. Porting SAA to AIX would certainly take a while. IBM could always port the look of OfficeVision to AIX, but it wouldn't have the same SAA underpinnings. In the interim, IBM has stated that it will provide for interoperability between SAA and AIX, its two strategic environments. In addition, IBM has promised a native office implementation under AIX, but no time frame was mentioned.)

OSF sensed as much when it chose a PM look and feel for Motif. But look and feel are as close as PM- and X-based GUIs come to each other, and, for the time being, X owns the Unix market. There's also PM's Unix version, PM/X, and, though it's a work in progress, it, too, may affect the direction of Unix user interfaces. The fact that PM/X hasn't reached the market yet puts it at somewhat of a disadvantage, and X has jumped out as a standard.

X: SOME GOOD POINTS. The industry has certainly taken a shine to MIT's X-Window system, and its very acceptance is one of its biggest advantages. The fact that MIT put X in the public domain didn't hurt its chances. Just in the past year, X has completely transcended its academic heritage. Witness, for example, the Xhibition conference in June. Attendance more than doubled over last year's, and a good many of those attendees were commercial developers.

X is significant for a number of reasons:

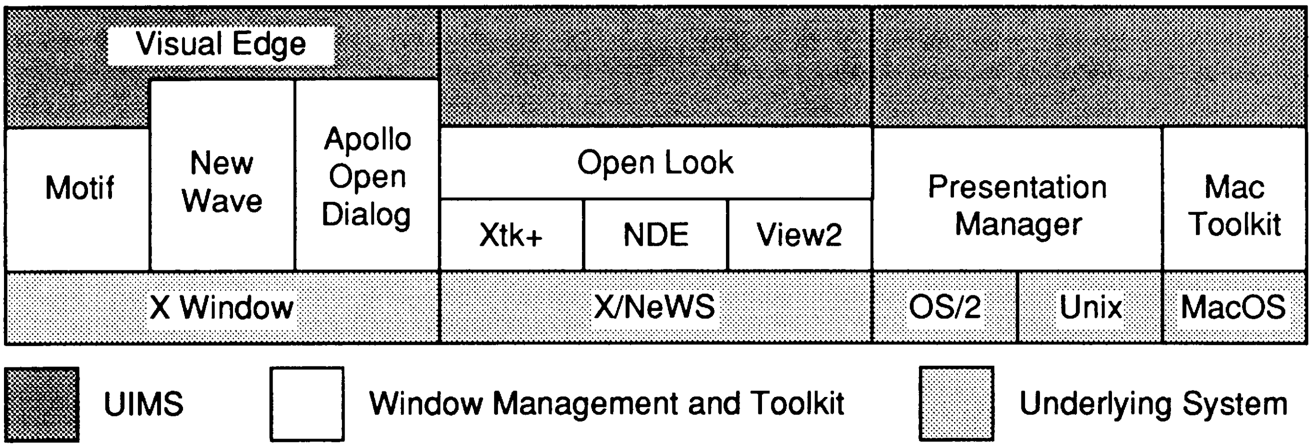
- It's free.
- It's portable—hardware-, software-, and operating system-independent. While X has taken off in the Unix world, it supports any system: DOS, Macintosh, VMS, MVS—you name it.
- It's distributed, and, as such, is an enabling technology for distributed network computing.
- It's a framework for object orientation. Not only does it provide for the essential graphical capabilities of object orientation, it also has an object-oriented programming model. Widgets are objects. They have specific properties, and an extended widget inherits the properties of its root widget.
- Its client/server model encourages the split of application and interface that we talked about with UIMSs.

SOME BAD. X has an inherent paradox. It was designed to be highly portable (and it is). To that end, it's a very low-level facility. Plain X doesn't address the presentation or UIMS aspects of a GUI. Its designers made a conscious decision not to impose any conventions on top of X, but rather to let individual implementers develop and extend widgets to satisfy their specific goals. And that's just what's happened. Organizations have developed individual toolkits, but none of them has been commonly accepted. As a result, no common API has been established. From a developer's standpoint, the portability of X has flown out the window.

The lack of a common API is X's most notable problem, but it has other limitations as well:

- Unless the client lives on the server, X is slow—even when you're using powerful machines as servers. So-called low-end X terminals have Motorola 68020 and 68030 chips.
- X is expensive. Not X itself—that's licensed free—but the hardware it requires. Not everyone who wants to use X has the right equipment to do so.
- Most of the applications being written to X are Unix applications and don't port to PCs. ("Most" is the operative word here. X applications are being written to other platforms; look at VMS.)
- X doesn't address security at all. Any jerk go in and tamper with or spy on your work.
- There seems to be a misconception about X and its relationship to DNC. DNC encompasses the notion of distributed applications. X only separates the display from the application and lets the application run in terminal emulation; it has nothing to do with parsing up the application. Therefore, additional application distribution tools are necessary.

PRESENTATION MANAGER. OS/2's Presentation Manager is a more complete system than X. Whereas no-frills X stops before window management enters the picture, PM defines all the graphics functions. Unlike X, PM has a single API, and all PM applications portray the same look and feel. More to the point, however, is that X is a distributed application environ-



GUI capabilities. Products mapped to three layers.

ment. Presentation Manager (and PM/X) isn't. That's a fundamental difference between them. A PM application lives on your machine—or at least a component of it does; an X application can be anywhere on the network.

Like X, PM has an object-oriented, hierarchical windowing paradigm. Main windows are independent of each other and encapsulate an application. These windows can have child windows, which inherit the class characteristics of the main window. Those child windows can spawn children of their own, and so on. Each window has a class handler that knows its interrelationships to other windows. For example, in a typical windowing scenario, several parts of an application may be spread over several windows. In this situation, the class handler keeps track of the relationship between parts of the application.

A PM application lives on your machine—or at least a component of it does; an X application can be anywhere on the network.

Presentation Manager's graphics programming interface is comparable to Display PostScript in function, and supports graphics (including kerning, bezier, cubic splines, and polygonal clipping), bit maps, and text editing. In addition, PM has the advantage of OS/2's dynamic link library, which allows for extra extensions.

CUA 2. First, let's get one thing straight. PM is not CUA; PM uses CUA. CUA (Common User Access) is part of IBM's SAA; it gives users a single interface across all IBM platforms. CUA provides basic guidelines for things like panel types and formats, selection techniques, and function keys. PM, on the other hand, is a full graphical user interface. You can consider CUA a subset of PM's functionality.

CUA has recently been extended from the text orientation of level one (which provided graphical menus) to a more graphical orientation based on an iconic representation of the desktop—a more object-oriented approach. IBM has dubbed this second level "workplace extensions." Its early releases will have limited functionality. For example, users will not be able to add a new icon to the desktop, nor will they be able to manipulate existing icons. However, releases in the first quarter of 1990 will provide much more flexibility. Users will be able to add new applications to the desktop and to register applications and devices. (Registering is an administrative process that establishes relationships between applications and devices, or objects.) For example, a document can be registered with a printer so that the user can print the document simply by dragging a document icon to a printer icon. Additionally, through the direct manipulations of objects, users will be able to cut and paste between windows without the use of a clipboard. We expect that, over time, IBM will add more functionality, including the ability to create live links between applications.

PM OBSTACLES. PM is new. Granted, this is a temporary setback, but, at the moment, few developers are familiar with it. It may look like MS Windows to a user, but not to a programmer. In addition, because it defines appearance and behavior, it has a lot more—and completely different—rules than X. Although multiple implementations leave X without a common API, once you get to know X, it's not hard to move from, say, DECwindows to Open Look. With PM, you're working with a separate and more complex model. Couple its newness with its inherent sophistication, and PM is intimidating.

Dialog Manager. Compounding the problem is the fact that PM hasn't had the same sophisticated development tools that X has (i.e., the various widget sets and toolkit implementations), and implementing windowing PM applications has been a difficult task. To alleviate some of the complexity, IBM is shipping the PM Dialog Manager in September, which will help programmers take advantage of PM facilities, including windows and graphics. Specifically, the Dialog Manager provides APIs for display services, variable handling, and session control. Programs written with the Dialog Manager conform to SAA/CUA conventions.

THE PROMISE OF PM/X. PM/X sounds like Presentation Manager on top of X, doesn't it? Wishful thinking, perhaps, because it isn't. PM/X is the Unix version of PM. PM is being rewritten from Assembler to C. Although it's still under development, when PM/X does arrive, it will mean a single API set for both OS/2 and Unix applications. With Motif, OS/2 and Unix applications can share the same look and feel—great for the user, but not for the developer. A common programming interface, on the other hand, is great for the developer. Furthermore, given X's status in the market, we suspect that PM/X may eventually support both local and remote X-based applications in addition to OS/2 applications (see illustration, page 9). Considering the growing demand for DNC solutions under Unix, PM/X may even become distributed itself. However, neither distribution nor support for X would be available any time soon. (We're talking years.)

... AND NEXTSTEP. Ah, NextStep—the dark horse—an enticing development environment that could greatly influence the future of Unix, especially since IBM has licensed it. (IBM, though, has a number of GUI options: CUA 1, CUA 2, AIX Motif, AIX NextStep, and even Metaphor.) What makes NextStep such a seductive environment is its tools. NextStep reaches into the UIMS layer. In fact, it was created especially to reduce the level of programming expertise necessary for application development.

The Pros. NextStep has a number of advantages over both X and PM:

- NextStep was designed from the bottom up as an object-oriented system. It's actually written in objective C, a C variant that implements support for classes, encapsulation, inheritance, and so forth.

- NextStep fronts PostScript with a window server, which gives software developers a strong front end to their applications.

- NextStep encourages users to build their own applications using graphical, system-provided building blocks. Class libraries give users and developers a strong foundation to build on. NextStep includes 38 object classes, including a PostScript manager, a text editor, system controls, ScrollView, a programming interface, and a storage object. Because of their object-oriented structure, these libraries make it easier to build sophisticated interfaces.

- NextStep applications cooperate with each other using Mach's object-oriented message-passing scheme, an application-to-application communication foundation for distributed environments. (For a close look at the Mach operating system, see Vol. 4, No. 6.)

The Cons. But NextStep is not the ideal environment (what is?). The folks at NeXT are a cocky crew, and they didn't really address compatibility. NeXT is its own environment and demands yet another API. Furthermore, since NeXT uses Mach (hardly a standard operating system), other systems won't be able to communicate with NextStep applications in a heterogeneous network.

A few other gotchas: We mentioned that NextStep has UIMS capabilities, but it makes no separation between display and underlying data. Also, NeXT has focused primarily on the developer at this point. Its user environment, on the other hand, is still immature. For instance, users can't embed objects within other objects. In other words, users can't stick a spreadsheet object into a document object—pretty limiting.

Issues and Options

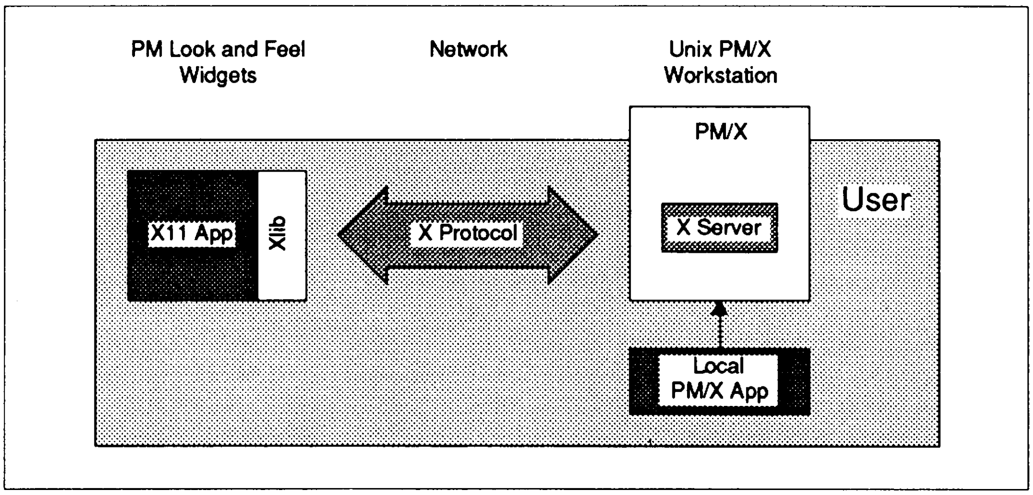
We hope you weren't expecting some happy ending here, where some whiz-bang tools come bounding in to save our developers in distress. The API problem is no small one. There are a bunch of APIs under X alone. An X application generated under one window manager will crash another. But the multiple API plight is not being ignored. In the X world, standards bodies are zeroing in on X intrinsics. The X Consortium is considering developing a common programming interface based on X intrinsics, and X/Open and the National Institute of Standards and Technology (NIST) have decided to adopt X intrinsics as a basis for standardization efforts. Frankly, these attempts are a little too little and a little too late. It's not as if they'll have any solutions soon, and they only address X.

In the meantime, commercial software houses have a couple of options:

- Sit and wait for the standard GUI to surface. But let's face it—you might miss the boat. The GUI market might be troublesome, but it's also opportunistic. Many predict that the window system that takes off will be the one that sits under hot new applications.

- Go for one. Developers could also just choose the platform they deem most promising and keep their fingers crossed (Russian Roulette style). You'll notice a blatant risk factor. But there's another consideration here. If most developers choose this path, it will severely damage Unix's commercial chances. Since things aren't settled with X, many developers would wind up choosing a more stable UI environment, like OS/2, Mac, or VMS, and postpone Unix until things calm down. And one thing the Unix market doesn't need is software houses postponing Unix application development.

- Go for them all. While that may be an option for the bigger software companies (and even they only have the resources to write to a choice few), it would blow away small ISVs. (Small companies are very often the source of innovation. We've been disappointed with the blase, state-of-the-market stuff we're seeing coming out of the big software shops.)



Possible PM/X model.

ABOUT THAT LAST OPTION. Let's rethink that last option. Rewriting applications for X, Mac, PM, etc., may be impossible, and we've already talked you blue in the face about the lack of a common API. Can't there be another way besides support for all the GUIs?

Actually, yes. We've seen a scant few approaches—pathfinders in the frontier of open systems portability technology. One example comes from Neuron Data's Open Interface Toolbox (OIT), an internally-used, GUI-independent development environment that supports a number of systems. Advanced Programming Institute (API), a Boulder, Colorado-based organization, provides another example—the Extensible Virtual Toolkit (XVT). Both tools feature similar design: an intermediate, portable library that's independent of the system it's running on.

Open Interface Toolbox

ABOUT NEURON DATA. Neuron Data, a four-year-old company based in Palo Alto, California, peddles expert system shells and artificial intelligence (AI) design and programming environments. Its marketing premise has been to provide AI products that are not hampered by lofty development costs, esoteric languages, extensive customization, and special purpose hardware (hurdles sometimes tripped over by AI vendors in the past). So far, it seems to be the right strategy. Since the company began shipping products in 1985, Neuron Data has sold over 6,500 software licenses—and that during a time when the AI industry has been on the decline.

Nexpert Object. Neuron Data's premiere product is Nexpert Object, a C-based, graphical, rule- and object-oriented expert system shell that runs on various machines, including VAXs, Macintoshes, PCs, and Unix systems. Nexpert uses objects and an inference engine to pool software resources for expert system development. (Essentially, an inference engine is responsible for the reasoning capabilities of an expert system.) Neuron Data has taken the "why don't we let customers integrate the applications they already have" approach with Nexpert. Currently, the product provides access to major Unix databases—Oracle, Sybase, Informix, and Ingres (as well as DB2)—and tools such as 1-2-3, Excel, and HyperCard.

WINDOW TO WINDOW TO WINDOW. Nexpert is interoperable across a number of different machines and window systems. You can develop an application on an X terminal, recompile it, and run it on a Macintosh. The product is conceptually window-system and operating-system independent. Clearly, this company has done something to leap over interface swamps. The bridge is the Open Interface Toolbox (OIT).

Neuron Data submitted the OIT as a candidate for OSF's user interface component last September, but it's not a commercial product. It's only used internally by Neuron Data's own developers for graphical user interface design. The toolbox has an extensible widget set and interface design tools. To its credit, Nexpert has a nice interface, giving the user colorful, graphically rich tools for visualizing the structure of data and the sequences and relationships of rules. From the outset, the company was working against the notion of expensive hardware for AI applications; portability became a necessity. OIT can potentially map to a whole slew of systems. Right now, the toolbox supports VMS/UIS, DECwindows, X, Macintosh, and PM. Support for NeXT, MS Windows, and Display PostScript are in the works.

The structure of Neuron Data's OIT targets two major concerns: portability and productivity.

OIT DESIGN. The structure of OIT targets two major concerns: portability and productivity. Portability is handled mainly by an intermediate layer called the virtual graphics machine (described in "Portability: The OIT Library" below). Productivity gets a shot in the arm from an object-oriented programming paradigm. OIT's extensible widget set is based on object-oriented principles. In addition, Neuron Data distinguishes among various levels of programming skills and has architected the toolbox accordingly.

More Layers. Yes, we're going to talk programmatic interface layers again. The toolbox has three (and you'll notice resemblances between Neuron Data's model and ours):

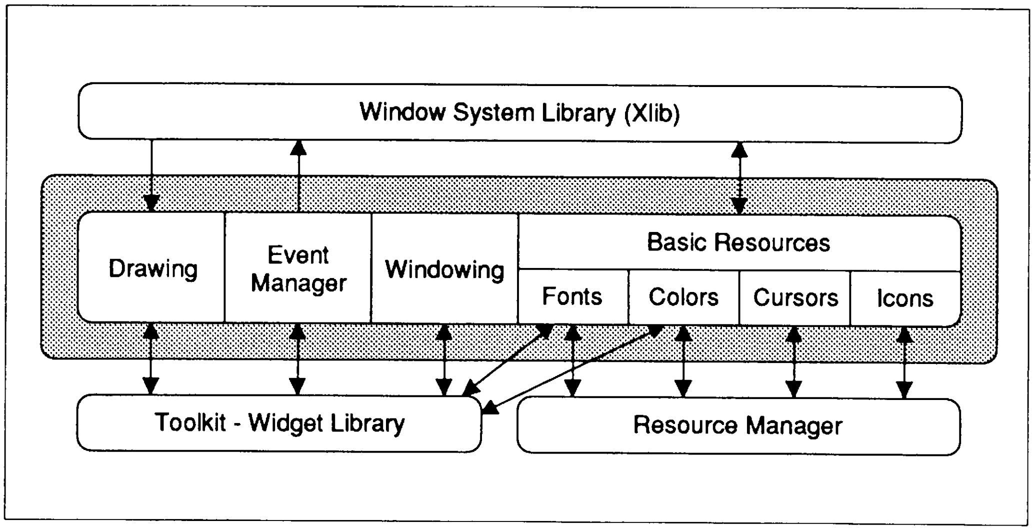
- Widget programming. Programmers create and extend widgets with the OIT toolkit. Again, Neuron Data took an object-oriented approach to widget-building and widget extensions. In other words, core widgets have specific properties that can be modified for customization (and most graphical applications require custom widgets, such as the drawing area of a paint program or the cell manager of a spreadsheet). Because OIT is not merely an X implementation, the toolkit widgets are not related to Xtk widgets. Instead, widgets are built on top of the virtual graphics machine.

- Application programming. The person doing the actual application design is not the widget builder. The designer works with a tool called the resource editor, which allows drawing or painting interfaces without much coding (see "Productivity: The Resource Editor" below).

- End-user programming. For end users, the OIT is essentially an editing tool. Users can tailor programs visually to suit their preferences.

Productivity: The Resource Editor. The crux of OIT operation for the application programmer is the resource editor. Developers generate user interfaces as if using a paint or draw program. Parameters such as the type of text button to implement (e.g., push button, toggle, check), its shape, its font type, and its behavior to mouse events are handled visually. In other words, developers determine the style of a widget by drawing it, not by coding in its parameters.

The resource editor produces two types of files: OIT ASCII resource files that describe the layout of the UI (its form) and programming source code (its function). While programmers can draw the look of an OIT-generated user interface, they still must do source-coding for its functionality. OIT gives some guidance for source-coding by providing templates of the functions in the OIT library (described in "Portability: The OIT Library" below). Basically, coding is reduced to a "fill in the blank" situation. You fill in the proper function call, and each screen you paint is linked to a library of OIT functions.



VGM architecture.

OIT also has a style guide, so applications have a consistent look and feel. Although extensions and variations can be made to the style guide, Neuron Data doesn't exactly recommend it, assuming (correctly) that your typical applications programmer is not going to know a whole lot about user interaction design. The style guide, we were told, was designed to maximize efficiency, "which requires a good understanding of the psychological issues involved."

Portability: The OIT Library. If the crux of OIT operation for the applications developer is the resource editor, the widget programmer depends on the virtual graphics machine and the OIT library. Actually, the library has various versions—one for each system OIT supports. OIT works with the concept of an intermediate layer between the underlying window system (Macintosh, PM, X, etc.) and the OIT development platform. You port an OIT application to another system by moving source code and resource files between machines and linking the appropriate OIT library.

The library has a partner in portability: the virtual graphics machine (VGM). The VGM is crucial to the overall design of OIT. Programmers actually use the VGM interface rather than the window subsystem to create and extend graphical components (or widgets; Neuron Data uses X terminology), so the functionality of OIT isn't dependent on any one system. The VGM interface supplies low-level functions and drawing operations. New widgets are stored in the OIT library, which sits on top of the VGM. The VGM interfaces to the underlying system with a VGM driver (like the library, one driver per system).

The VGM has a number of responsibilities, including:

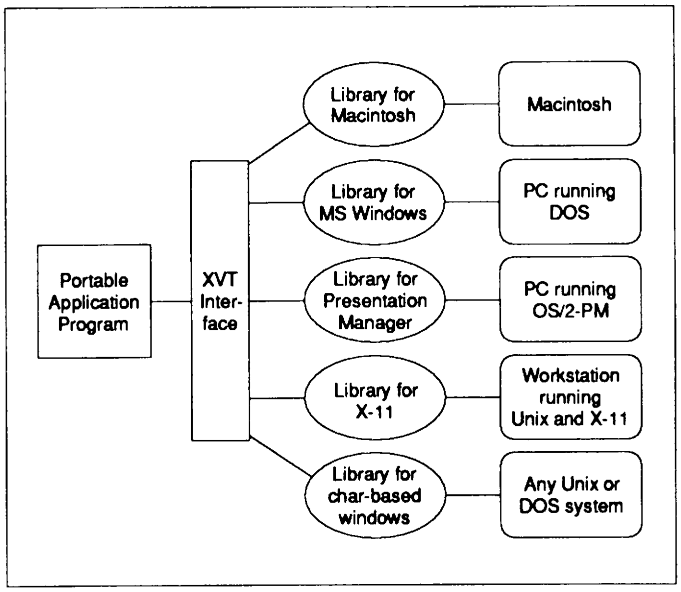
- Encapsulating windowing system implementation details
- Defining core widget structure
- Dispatching events to widgets
- Processing exposure events clipping
- Dispatching timer events
- Managing window operations (move, resize, iconify, etc.)

COMMENTS. The toolbox is a very workable solution for Neuron Data. However, the company obviously jumped through a few hoops to develop it. We've already mentioned the difficulty of finding developers who have a solid relationship with multiple GUIs. Apparently, Neuron Data did just that. The library contains abstractions of several window systems. However, many companies do not have the resources to go through these gyrations.

Extensible Virtual Toolkit

Advanced Programming Institute's (API's) Extensible Virtual Toolkit (XVT) may spare companies those gyrations because it's an actual product. Like OIT, it's a portable user interface programming environment. It was first shipped in January 1988 for the Macintosh and MS Windows, and API is just finishing versions for PM, X.11, and character-based terminals.

WORKING ASSUMPTIONS. API started developing XVT with a number of ground rules. Because portability was the main goal, the toolkit had to abstract the features common to a number of systems. API wanted its function calls not only to look like those of a window system and toolkit (with events, windows, graphics, fonts, and so forth), but also to be simpler



XVT architecture.

than the underlying system. In order to hit the right level of abstraction, the company set out with six assumptions about XVT's design:

- Commercial viability is a must. To API, the only way to really test XVT was to release it commercially, exposing it to scrutinization under real-world applications.
- XVT should support all standard environments, especially X, MS Windows, PM, and Macintosh.
- The interface must be thin. That is, it must use whatever the host window system offers for window management, menus, resources, dialog, and so on.
- The programming language had to be plain C—without any preprocessing or interpretation.
- The developer shouldn't have to dive into the native window system. Portability must be achieved through abstraction, not through multiple paths.
- The developer should be able to dive into the native window system if he must (i.e., if XVT doesn't support a certain feature).

XVT DESIGN. With these rules in mind, API, like Neuron Data, adopted a middle layer concept for XVT. The XVT library has a single interface that is identical across all systems. It contains about 200 function prototypes along with type definitions for abstract objects, such as windows and fonts. Therefore, applications developers just write to one interface. The library itself has multiple implementations—one for each host system it supports (see illustration at left).

COMMENTS. The features XVT provides are, by nature, lowest common denominator, and that may prove to be a stumbling block for some developers. (Incidentally, the same can be said for OIT.) Yet, developers rarely need 100 percent portability; they need tools that reduce the cost of porting, and that's what XVT does. Certainly, 90 percent portability is better than no portability at all.

Conclusion

PORTABILITY, NOT STANDARDIZATION. At first, you'd think it would be good to see the industry standardize on a user interface—just mandate something like Motif or Presentation Manager. But, aside from political ramifications (let's face it, would AT&T stand for industry standardization on Motif or PM?), such a move would severely cramp innovation of better, more seductive, streamlined interfaces.

A common API is a more practical solution. Developments such as Neuron Data's OIT and API's XVT are precisely the kinds of tools developers need. Additional portability techniques would also help: portable UIMSs and modular user interface libraries. What we're after is interoperability. We want to be able to mix and match applications. We want our applications to plug and play together. It isn't the difference in appearance that's annoying; it's the inability to jump back and forth as quickly as we would like, and to perform operations as easily across applications as we can within applications.

THE SOFTWARE SANDBAR. At the Executive UniForum Conference in April, Bill Joy, the ever-blunt VP of Research and Development at Sun Microsystems, dismissed the recent user interfaces hullabaloo as "much ado about not much." He touted application development instead. If it takes five minutes to learn a different interface for a really neat application, who cares?

The user interface wars, though, are getting in the way of application development. As software vendors continue to stress choosing a development environment, we continue to wait for better applications. In time, as GUIs become more mainstream, programmers will be more competent to build portable toolkits that abstract the functionality of a number of systems (à la XVT and OIT). In the meantime, users are pulling their hair out looking for solutions for their emerging needs: coprocessing, interoperability, modularity, object orientation, portability. We need to look several levels below the surface, not at the look and feel of computer systems.

Announcing the Seventh Annual

SEYBOLD EXECUTIVE FORUM

Preparing for the Global Information Age

SESSION AND SPEAKER HIGHLIGHTS:

INFORMATION SERVICES IN THE 1990s & BEYOND

Dow Jones & Co., Inc.—William C. Dunn, *Executive Vice President*
McGraw-Hill—Thanos Triant, *Senior Vice President, Product Development and Technology*
Reuters

CLOSING THE GAP: New Tools For Research & Development

Digital Equipment Corporation—Samuel H. Fuller, *Vice President, Corporate Research*
Merck, Sharpe and Dome—Dr. Myra Williams, *Exec. Director of Information, Research and Strategy*

BLURRING ORGANIZATIONAL & NATIONAL BOUNDARIES: Beyond EDI

Chase Manhattan Bank—James T. Whelan, *Vice President, Office Integration Services*
Soft•Switch—Michael Zisman, *President*
State of Florida—Edwin A. Levine, *Staff Director*

WORLDWIDE FINANCIAL TRADING: The Prototypical Electronic Market

Institutional Investor Magazine—Larry Marion, *Senior Editor*
Merrill Lynch & Co.—Gerald Ely, *Director Information Systems Division*
Morgan Stanley—Alan Trager, *Managing Director*

THE PAYOFF FROM OPEN SYSTEMS

American Airlines—Chicke J. Wright, *V. P., Technology Support*
E.I. DuPont de Nemours & Company—Danny C. Wigley, *Senior Systems Consultant, Information Systems, Fibers Division*
Warner Communications—Donald Winski, *Chief Information Officer*
Data General—Jan-Pieter Scheerder, *Director, Open Systems Marketing*

THE WORLD'S EVOLVING COMMUNICATIONS INFRASTRUCTURES

ICL House—Robert Smith, *Director of Networks Industry Division*
Northern Telecom
Nippon Telephone & Telegraph

DIRECTIONS IN MULTIMEDIA TECHNOLOGY

Apple Computer
Intel
MIT Media Lab—Muriel Cooper, *Director of Visible Language Workshop*

COMPUTERS & COMMUNICATIONS: A Converging Market

AT&T—Robert M. Kavner, President, *Data Systems Group*
IBM
NEC

October 30, 31, November 1, 1989

The Royal Sonesta Hotel, Cambridge, Massachusetts

Sponsored by: Patricia Seybold's Office Computing Group

REGISTRATION: $1,095.00

Advance registration is required. To register, call Deborah Hay at 1-800-826-2424 (in Mass. call 617-742-5200) or send a fax to 617-742-1028.

NEWS ANALYSIS

PRODUCTS • TRENDS • ISSUES • ANALYSIS

• INSIDE •

Lower-Cost Performance. Digital's RISC Processors Offer More Bang for the Buck than Does Its VAX Line. **Page 14**

WordPerfect Poses a Threat from Below. **Page 15**

Overcoming Hurdles at Xhibition. **Page 16**

Enjoying a Choice of RISCs. **Page 17**

•DIGITAL•

Caught in a Squeeze

In launching a full line of systems based on RISC (Reduced Instruction Set Computers) processors last month, Digital Equipment Corporation further exposed the gulf in price/performance between its VAX and RISC lines. Digital finds itself caught in a squeeze: Although VAX/VMS is its engine of profitability, its own RISC/Unix platforms offer an increasingly attractive alternative.

Digital's response has been to rely on VAX/VMS's strength in applications and networking to beat back the challenge from within. However, given the rapid progress of its RISC platforms, Digital will soon have to elevate them to a position equal to the VAX in networking, applications, and marketing.

The irony of Digital's position is that it has done such a good job of preparing for a shift from reliance on VAX/VMS as its engine of profitability to open systems. The growing repertoire of services in Digital's Network Application Support (NAS) provide a unified foundation for both VMS and Ultrix applications. Digital has built a full line of systems based on the MIPS Computer Systems RISC architecture in a very short time.

Yet Digital hesitates to make a full commitment to open systems. VAX/VMS is just too lucrative. Digital still bids VMS first whenever possible, even on Unix RFPs. VMS, Digital executives point out, will become an open system to rival Unix as it is made compliant with the Posix interface specifications. But this strategy is of no help to customers who want open systems now. Posix compliance for VMS is years away.

A WIDENING GAP. The question of when and how to shift from reliance on VMS to aggressive promotion of open system platforms is the subject of internal wrangling at Digital and fumbling in the marketplace. The recent price/performance differentials between Digital's new RISC/Unix platforms and their VAX equivalents make it clear that the uncertainty must end. For the sake of comparison, we use VAX 11/780 equivalents as the unit of price/performance measurement.

- DECstation 2100. One of the three new RISC systems, the DECstation 2100 is based on a MIPS R2000 processor rated at nine VAX units of performance (VUP), or 10 MIPS. In a typical configuration, it costs $14,000 ($1,555/VUP), although prices start at just under $8,000. An equivalent VAXstation 2000 gives the user less than two VUPs, at about $5,000 apiece.
- DECstation/System 3100. Announced last January, the 3100 is rated at 12 VUPs. A typical single-user system costs $30,000 ($2,500/VUP); a typical server costs $34,400. An equivalent MicroVAX/VAXserver 3100 offers 2.5 VUPs at $7,400 per VUP.
- DECsystem 5810/20. At the high end of Digital's RISC line are uniprocessor and dual-processor models based on a 25 MHz R3000/R30010 complex in a VAX 6000 cabinet. In a typical configuration, the uniprocessor DECsystem 5810 delivers 14 VUPs for $300,000, or $21,500 per VUP.
- DECsystem 5400. This is a MicroVAX cabinet with a 20 MHz R3000 CPU and R30010 floating point processor. It is rated at 13 VUPS, $7,000 each in a typical configuration. Comparable figures for the equivalent MicroVAX weren't available.

These numbers reiterate what we all know: the RISC and Unix pairing delivers performance at a much lower cost than VAX and VMS. Digital's response today is to position its RISC/Ultrix platforms as high-performance/low-function products. If you want applications, network management, etc., the party line goes, choose VAX/VMS. What Digital could be doing is revamping VAX/VMS to bring it into line with the economies of RISC and Unix and aggressively building and marketing its DECstation/server line in the meantime.

THE GOOD NEWS IS... Fortunately, while the marketing strategists at Digital fiddle, the development and support shops aren't burning. Development of NAS as a robust set of cross-platform services continues. VAX/VMS always gets new services first, but at least Digital is, in fact, adding support for its RISC/Ultrix platforms. With NAS services available to both, VAX and RISC platforms can participate as cooperative peers in network applications. Applix Incorporated is developing a version of its Alis office automation software for VMS to leverage Digital's cross-platform integration services. Alis already runs under Ultrix.

IBM is working toward this goal in connecting and integrating its Systems Application Architecture (SAA) systems and AIX systems. But it is two to three years behind Digital.

Digital is also building a base of software for the new platforms. Three Digital Porting Centers are helping developers port their applications over to the new platform, and Digital also has founded a special program to encourage development at universities for its RISC platforms. Digital expects to have some 250 software packages for the new machines by the end of next year. As this number grows, VAX/VMS's edge over the DECstation/server line shrinks.

THE BYTE-ORDER ISSUE. To promote its cross-platform software strategy, Digital has taken a calculated risk with its MIPS Computer RISC processors. Digital is not using MIPS's binary compatibility standard that ensures out-of-the-box software compatibility for all implementations of its processors. Rather, Digital reversed the byte order of MIPS standard, bringing it into conformance with its VAX byte order.

Digital believes it is more important to promote commonality between its VAX and RISC lines than it is to participate in a general MIPS-based software-swap meet. The risk is that users will demand cross-platform binary compatibility, in which case, both Digital and MIPS could find themselves out in the cold. This is unlikely, however. Moving from DECstations to other MIPS-based computers will, in most cases, require a simple recompile.

CONCLUSION. As Digital sells its dual platforms, the price/performance advantages of its RISC/Unix platforms will become more and more apparent to users. Dangling VAX/VMS's rich applications and networking base as an alternative to open systems will work for only so long. Digital's brass must shift attention to building the DECstation/system platform with lots more applications, robust networking services, and aggressive marketing. The longer they delay, the more bids they'll lose to Unix offerings from Sun, Hewlett-Packard, and others. — *J. Rymer*

•WORDPERFECT•

LAN Office System Moves to Unix

At the same time that Unix office system vendors are feeling pressure from above as the major systems vendors threaten to invade their turf, they are about to face another challenge from below: the extension of the LAN-based office system to Unix.

WordPerfect Corporation (Orem, Utah) has announced the release of WordPerfect Office for SCO's Xenix System V/386, Release 2.3.1, with other Unix versions to follow. The Xenix version is priced at $995 per processor, i.e., per host in a multiuser system and per workstation in a networked system. In addition to Unix, WordPerfect Office currently runs on DOS LANs, Digital VAX/VMS, and Data General AOS/VS, with a Macintosh version due later this year.

WordPerfect Office consists of Mail, Calendar, Calculator, File Manager, Notebook, and Program and Macro Editors that can be called from a menu-based Shell. Individual applications, including, but not limited to, the WordPerfect word processor, can also be accessed from the Shell. The Shell also provides a way to integrate dissimilar applications via a cut-and-paste utility. Even closer integration is possible for those applications written to WordPerfect Office specifications, whose APIs will be published by WordPerfect Corporation.

WORDPERFECT OFFICE FEATURES. The Unix version of WordPerfect Office is very similar to the PC LAN version, which we examined in the *Office Computing Report*, Vol. 12, No.5.

The Shell hides the traditional Unix environment, providing easy access to the services and applications and permitting hot-keying between them. It is not, however, graphical in nature, and although WordPerfect Corporation has stated that it will write to the predominant graphical user interfaces, the company has not announced a timetable for Motif and/or Open Look versions.

The electronic mail system allows users to send any type of document to any local WordPerfect Office user or to users on multiple Unix systems via Ethernet (TCP/IP) or UUCP. The notebook is a flat-file database for applications such as online address books. The file manager allows file manipulation (retrieve, delete, rename, copy, or move) without using Unix commands.

The Calendar is strictly a personal utility, with the group-scheduling function that is available on the PC LAN platform not yet implemented in Unix. Implementing the Scheduler has been a consistent problem for WordPerfect; bugs in the DOS product have delayed its porting to other platforms. WordPerfect will not begin porting the Scheduler to Unix and the other systems until a more stable version of the DOS Scheduler is ready later this year.

WHAT, ME WORRY? When compared to the current range of Unix office products such as Uniplex, Alis, or Q Office, WordPerfect Office tends to fall short in both functionality of the modules (with the exception of the word processing module) and integration between them. Why, then, should anyone worry about this upstart from below, particularly when the major threat seems to be coming from above in the form of distributed networked office applications being touted by the major players such as Digital, Data General, and Hewlett-Packard?

WordPerfect Office will pose a threat to the current array of Unix office system vendors for two specific reasons: (1) the great popularity of its DOS products, and (2) the multiplatform strategy being pursued by WordPerfect.

Familiarity Breeds Sales. WordPerfect has already shown that it can move onto other platforms and leverage users' familiarity with the company and its products. Arguably, after Lotus 1-2-3, the second most popular application on all platforms combined (and certainly on DOS), WordPerfect has set the standard for PC word processing and is rapidly gaining market share on Unix and on the VAX. This "brand name" association itself creates an advantage, as does the fact that there are a great many users who are comfortable with WordPerfect's user interface and many more who have had at least some exposure to it.

Multiple Platforms. WordPerfect Office holds its greatest promise as an integrating force in a multiplatform, multi-OS environment. However, this was not the company's original intention. WordPerfect initially saw Office as an extension to its other offerings (individual word processing, database, and spreadsheet applications), with the major opportunity on the DOS LAN, where comparable office offerings did not yet exist or had not yet taken hold. Following its corporate policy of trying to be on all of the major platforms, the company then committed to move Office to VMS, AOS/VS, Unix, and the Macintosh.

WordPerfect was then faced with the discovery that its large customers were looking toward it to provide them with a method to integrate their disparate systems, particularly the PC LANs, with the rest of the corporate environment. A common user interface, along with electronic mail, calendar/scheduling, and document management were all high priorities for these customers. In reaction to this demand, WordPerfect is now beginning to position Office as the glue to bind a number of systems, be they multivendor PC LANs or any mixture of PC and Mac LANs, VAXs, Data General machines, and Unix systems. Therefore, the next set of releases will enable the mail systems on each to interoperate, with integrated group scheduling to come later. The company is also working mail connectivity through gateways to many external systems (including Action Technologies' Message Handling System—MHS, MCI Mail, X.400, and fax) in addition to the current UUCP and Digital interfaces. ◐ *—D. Marshak*

•TRADE SHOWS•

Xhibition '89

Xhibition '89, the second Xhibition conference, seemed to be more of a confirmation of X's establishment as a commercial standard than anything else. Twice as many people attended as last year. And the show was by no means dominated by Unix gurus. During one session, commercial developers in the audience were asked to identify themselves by a show of hands, and at least 40 percent of the hands in the room flew up. However, while vendors are jumping on the X bandwagon, X has issues yet to be settled.

COMMERCIAL VIABILITY. In this month's feature, we talk about some of X's hurdles (one speaker preferred to call them opportunities) in commercial environments. Those hurdles—or opportunities—became the focus of this year's Xhibition conference.

Networking. Much of X's appeal comes from the fact that it's a distributed window system. However, X only splits an application and interface. In a true distributed environment, you want to spread your application across the network, and X offers no help here. That's not to say that distribution tools are X's responsibility, but that X alone is not enough. The need for products that help developers effectively distribute applications is bigger than ever.

Another problem of X's networking scheme is security—or the lack thereof. X doesn't address security whatsoever. And, apparently, it won't address security in its next release, either. During Xhibition's session on security, Jim Fulton, from the X Consortium, offered as a solution something along the lines of the Kerberos authentication system, developed at MIT's Project Athena. Essentially, the system won't recognize you unless you answer an encrypted message (every message, incidentally, is unique). Whether or not the answer is Kerberos, security in X is a vital concern, and solutions need to be worked out.

X is New. Most commercial developers aren't familiar with X. No doubt, that's why so many of them attended Xhibition. In addition to technical sessions, the conference featured tutorials on things like X programming, widget-writing, object-oriented programming, X intrinsics, Motif, XView, Display PostScript, the Andrew system, and even one on user interface design (it's nice to think that programmers might know something about the cosmetics of user interface design). These tutorials were designed specifically to help bring programmers up to speed on X and its surrounding issues.

Obviously, however, a few Xhibition tutorials won't do the trick. The X world has taken a keen interest in speeding the learning curve for developers. User Interface Management Systems (UIMSs), which we look at closely in this month's feature, should help. Today, most UIMS products come in the form of interface layout and design tools—tools that let you paint and customize a screen. Ideally, a UIMS should be easy enough for a graphics designer or even an end user. That way, it won't be a techie who's designing your interface (techies don't really make the best designers); you can do it yourself.

Display PostScript. It's been said that the X graphics primitives are just that—primitive. The DPS extension provides a supplement to basic X, bringing to it some of the flexibility you find in NextStep.

Standardization: The API Issue. X has no common API. This was the concern most often discussed at Xhibition. Vendors have implemented individual toolkits with individual widget sets and window managers that won't support other toolkits and widget sets and window managers. What a mess. Standards bodies seem to be looking at X intrinsics as a source for standardization. There was also talk of a higher-level API that's not intrinsically tied to any window system.

In the meantime, Unix developers need only consider two graphical user interfaces—Motif or Open Look. And, of the two, Motif seems to have the advantage.

AND ON THE FLOOR. The exhibition floor offered no surprises: X applications and terminals (some with color and big screens—the kind you'd like to watch a football game on).

One repeated complaint of X is its slowness. To that end, X hardware suppliers are pumping up their terminals with more powerful chips (68020 and 68030 chips). In other words, you've got a PC on your desk, not a terminal.

Next, we'd like to see X hardware become less expensive. Few can afford the equipment necessary to run the X Window system, which not only limits the number of users capable of running X applications, but interferes with X development as well. ◐ —*L. Brown*

•RISC•

Some Semblance of Order

The RISC market may appear to be a free-for-all, but it actually has begun to coalesce. It has become clear that, unlike the CISC microprocessor market, all of the major players—Motorola, Sun, Intel, MIPS Computer, Hewlett-Packard, and IBM—will be winners. Most vendors are not locking themselves into a single RISC technology.

ENOUGH FOR EVERYONE. Given the big price/performance advantage of RISC when it is paired with Unix, we expect the market for RISC-based processors to grow rapidly in the near term. Digital Equipment's latest RISC announcement is the latest—but surely not the last—testimony on the differential between RISC/Unix and proprietary architectures (see "Caught in a Squeeze," in this issue).

This growing market will support multiple architectures for two reasons. First, there's simply a lot of demand for RISC/Unix systems, enough to support lots of vendors.

Second, Unix's presence dramatically simplifies the software picture. This operates at two levels, binary compatibility and porting. All of the major architectures—with the possible exception of Hewlett-Packard's Precision Architecture—have their own intra-architecture binary-compatibility standards. Binaries allow users to run software out of the box on conforming implementations of the individual chip sets. For users, binaries provide a level of cross-vendor compatibility.

For ISVs, porting from one RISC implementation to another is straightforward. If software vendors honor their commitments, they'll very quickly build libraries of applications for all of the major RISC architectures. For example, Oracle will be available on MIPS, SPARC, 88K, and so on. Users will be able to pick and choose from among RISC architectures with a fair amount of confidence that applications will be available that span platforms, masking the differences of the underlying RISC architectures. In the long term, compilers that allow easy moves from one architecture to another will make porting from one architecture to another as simple as a recompile.

HOW THE OPTIONS SHAKE OUT. All of the major participants in the RISC market will win by succeeding in niches. Here is the way the market looks at this point in its development:

- SPARC. As the earliest RISC implementation targeted at general availability, Sun Microsystems' SPARC has attracted a strong following. It leads the field in terms of OEMs, with one, Solbourne Computer, soon to announce its second generation of SPARC-based workstations. SPARC has the largest library of applications as well. If SPARC has a weakness, it is in its scalability. Major system vendors have tended to use SPARC to build workstations, and other RISC designs to craft supermini- and mainframe-class machines.

- 88000. Motorola's RISC chip set is quietly gaining momentum. Unisys, the largest commercial Unix vendor, has just signed on to produce a full line of systems—from workstations to mainframes—using the 88K. Data General's 88K-based Aviion systems became generally available this month. Meanwhile, Data General has spearheaded a drive to sign up ISVs that is proving successful. Expectations are that Apple and AT&T will also use the 88K in future systems.

- MIPS R Series. Digital has quickly built a full line of systems based on MIPS's R Series. There are two wins for MIPS in this. First, Digital has demonstrated that MIPS's architecture supports very large systems as well as workstations. Second, Digital will sell a lot of RISC/Unix systems and, indirectly, help attract ISVs to the platform. The downside is that Digital will not be promoting MIPS's binary compatibility standard—it has gone its own way—and that it professes no special commitment to use only MIPS. MIPS risks losing ground to the 88K and eventually becoming a second choice.

- Intel i860. Intel's RISC chip has a dual role in the market. First, it is being seriously considered as the successor to today's 80386-based Unix systems. Unisys's Server Products Division is working on such a product, and IBM has also expressed an interest. The i860's other life will be as a coprocessor to either Intel 80X86s or other RISC chips. Software availability for the i860 lags behind that of other architectures. The reason: It was introduced just six months ago.

- HP Precision Architecture. One of the earliest RISC architectures, HPPA was recently made available for licensing to OEMs. Hewlett-Packard (HP) has bolstered its original design with some of the features—multiprocessing support, in particular—of Apollo's Prism RISC architecture. HP acquired Prism when it bought Apollo earlier this year. HPPA promises to be a strong contender. It is mature, having been the foundation of a full product line for two years. It has a healthy library of software as well. And HP's position as the largest supplier of Unix-based workstations won't hurt, either.

Given the close relationship between AT&T and Sun, you'd expect AT&T to be cheerleading for SPARC. Not so. AT&T will employ multiple RISC architectures.

- Son of RT. The RISC design in IBM's RT PC is the wild card in the market. IBM is working on a second-generation RT line that will include both workstations and rack-mounted large systems. Unlike the lackluster RT, the new products will at least equal the performance and features of competitors, IBM promises. If they're hot, the new RTs will also blow away IBM's current Unix platform, the System/370 mainframes, as the primary choice. The questions for IBM are: Will it suffer by being the only RISC design not generally available to OEMs, and can it build a big software library quickly?

THE MULTIRISC STRATEGY. Unisys, AT&T, and Digital have indicated that each will use two or more RISC architectures to build future Unix systems. Unisys will use the 88K as the basis for successors to its S/Series, and is likely to use the i860 in follow-ons to its Server/PC and SPARC as the basis for workstations. Unisys already sells SPARC workstations to its federal customers.

Given the close relationship between AT&T and Sun, you'd expect AT&T to be cheerleading for SPARC. Not so. James Clark, new chief of AT&T's Minicomputer and High-Performance Systems Division, is telling the world that AT&T will employ multiple RISC architectures. Clark won't reveal his plans, but the smart money says he'll choose SPARC at the low end and either MIPS or the 88K in the midrange and up.

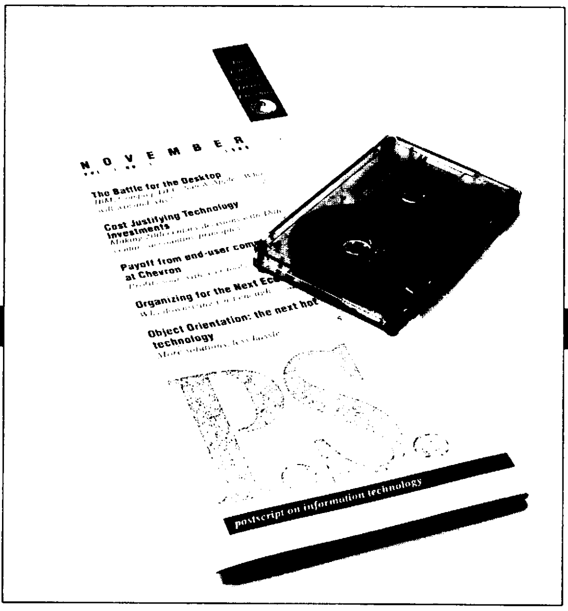
Beyond Digital's declaration of independence from MIPS, little is known about its plans to use other RISC architectures. Digital president Kenneth Olsen recently revealed that the company is exploring the possibility of an overhaul of its VAX architecture, which presumably would utilize RISC technology.

CONCLUSION. As the RISC market settles into a pattern, marketing has become the key to success for the players and their customers. Building software libraries and stables of quality OEMs will determine which of the RISC architectures is the biggest winner. Meanwhile, end users are likely to enjoy ample choices and the freedom to exercise them. ◐ —*J. Rymer*

INTRODUCING

P.S.

postscript on information technology


N O V E M B E R
The Battle for the Desktop
Cost Justifying Technology Investments
Payoff from end-user computing at Chevron
Organizing for the Next Ec
Object Orientation: the next hot technology
P.S.
postscript on information technology

Patricia Seybold Launches a New AudioNewsletter

CURRENT TRENDS IN NONTECHNICAL TERMS

This innovative monthly service, both an audiocassette and a printed newsletter, is designed for you, the technology watcher, and for your nontechnical colleagues. It comes to you from Patricia B. Seybold, publisher of *The Office Computing Report, Unix in the Office,* and *Network Monitor.* Every issue presents discussions of current issues and technology trends in nontechnical terms so that you and your associates can find out what is going on and make more intelligent decisions.

FASCINATING INTERVIEWS

Each month, Patricia Seybold will bring you fascinating interviews on a variety of topics, including:

- Executive Information Systems
- Cost Justification
- Object Orientation
- Evolving Standards
- Desktop Platforms
- End-User Computing
- Managing Change
- Partnering for Progress

She will also give you her personal view of where things are headed. . .what to watch. . .and what you can safely ignore.

INNOVATIVE AUDIOTAPE REVEALS EVEN MORE

But that's just the beginning. . .because an exclusive audiotape that accompanies each printed edition of the newsletter reveals even more:

- One-on-one interviews with industry insiders
- Technology trends explained in plain English for busy executives who must make fast, intelligent decisions
- Firsthand success stories—how your peers manage change in their corporate culture

Ordering Information

CALL FOR YOUR FREE TRIAL SUBSCRIPTION TODAY!

Call Toll-Free: 1-800-826-2424 to order your trial subscription to **P.S.** ***postscript on information technology***, or write to:

Patricia Seybold's Office Computing Group
148 State Street, Suite 612, Boston, MA 02109.

There is no obligation. We'll send you a FREE ISSUE—both newsletter and audiocassette—of **P.S.** ***postscript on information technology.*** Pay $295 when your bill comes in and you'll receive an issue once every month for a full year—12 issues in all. If you do not wish to subscribe, simply write "cancel" on your bill.

You can also benefit from **P.S.** ***postscript on information technology*** by ordering the $95-a-year newsletter only.

P.S.

Listen to it in your car on your way to work or plug into it as you exercise!

Patricia Seybold's Computer Industry Reports

ORDER FORM

Please start my subscription to:		U.S.A.	Canada	Foreign
☐ *Patricia Seybold's Office Computing Report*	12 issues per year	$385	$397	$409
☐ *Patricia Seybold's UNIX in the Office*	12 issues per year	$495	$507	$519
☐ *Patricia Seybold's Network Monitor*	12 issues per year	$495	$507	$519
☐ *P.S. postscript on information technology*	12 issues & tapes per year	$395	$407	$419
☐ *P.S. postscript on information technology*	12 issues per year	$ 95	$107	$119

Please send me a sample of: ☐ *Office Computing Report* ☐ *Network Monitor* ☐ *UNIX in the Office* ☐ *P.S. postscript on information technology*

Please send me information on: ☐ Consulting ☐ Special Reports ☐ Conferences

☐ **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: ____________ 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's Office Computing 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.

IU-0889

Send to: Patricia Seybold's Office Computing Group: 148 State Street, Boston MA 02109; FAX: 1-617-742-1028; MCI Mail: PSOCG
To order by phone: call (617) 742-5200

Topics covered in Patricia Seybold's Computer Industry Reports in 1988/1989:

Back issues are available, call (617) 742-5200 for more information.

Office Computing Report

1988—Volume 11

#	**Date**	**Title**
10	Oct.	Document Processing
11	Nov.	Upper CASE Tools—Getting Users to Help Design Systems
12	Dec.	Executive Information Systems

1989—Volume 12

#	**Date**	**Title**
1	Jan.	Digital's Office Strategy—Going After the Desktop
2	Feb.	Compound Documents Take the Stage
3	Mar.	Wang's Freestyle—Not the Paperless Office, But an Office with Less Paper
4	Apr.	Focus on Training Technologies—Office Systems Training from DG, Digital, HP, IBM, and Wang
5	May	The LAN Office Emerges—WordPerfect Enters OA Race
6	June	Objects at Floodtide—Object Orientation Permeates New Development
7	July	Decwrite—Advanced Applications under DECwindows, Part 1

UNIX in the Office

1988—Volume 3

#	**Date**	**Title**
10	Oct.	DECwindows
11	Nov.	Data General's Unix—A New Beginnning
12	Dec.	Document Processing with Interleaf *Second Feature:* Apollo and Sun Service and Support

1989—Volume 4

#	**Date**	**Title**
1	Jan.	Unisys Ofis Ensemble—In Search of Cohesion
2	Feb.	A Tale of Two Operating Systems—System V.4 and OSF1
3	Mar.	FrameMaker 2.0—The Art and Science of Document Processing
4	Apr.	CMU's Andrew Project—A Prototype for Distributed, Object-Oriented Computing
5	May	Sybase—The Next Generation
6	June	Thoroughly Modern Unix—The Shape of New Operating Systems
7	July	Surveying the Users—Taking the Pulse of Commercial Unix

Network Monitor

1988—Volume 3

#	**Date**	**Title**
10	Oct.	ISDN: We Still Don't Know
11	Nov.	Data General's PC Networking Strategy
12	Dec.	3+Open—3Com Makes Its Move

1989—Volume 4

#	**Date**	**Title**
1	Jan.	APPN—Peer-to-Peer Networking Arrives at IBM
2	Feb.	Bridges & Routers—Dividing to Conquer
3	Mar.	BBN Builds Big Networks—Designing Private Packet-Switched Networks
4	Apr.	Northern Telecom DNS—Trying to Build Intelligent Networks
5	May	NetView—IBM's Enterprise-Wide Manager
6	June	Apple's Networking Strategy—Bringing the World to the Desktop
7	July	Distributed Object Management—Networks Made Easy... And More