Patricia Seybold's Office Computing Group

Editor-in-Chief
Michael A. Goulde

INSIDE

UNIX IN THE OFFICE

Guide to Open Systems

Vol. 7, No. 4 • ISSN: 1058-4161 • April 1992

The X Window System

Where Is Its Future?

By Andrew D. Wolfe, Jr.

IN BRIEF: If the late 1980s marked the rise of open systems, they were also the years in which personal computing began driving the computer industry. The mainframe market contracted severely, and many vendors of proprietary computing systems have been forced out of that business. But, as the decade waned, open systems were still only emerging—they had not yet taken over the computing market. If anything, personal systems were capturing it from below. Fundamentally, PCs—and especially the highly usable Apple Macintosh—had given the industry the new mission of empowering the end user. Now, potential buyers would measure software and systems as much by simplicity and ease of use as by price and functionality. The Unix community responded to this mission with the X Window System. This article will examine the X Window System, the controversies around it, its flaws and virtues, and its directions for the future.

Report begins on page 3.

Now That the GUI Wars Are Over

Users Must Pick Their Winner

IT WASN'T SO long ago that we wondered if Motif, OpenLook, Windows, the Macintosh, or even NeXTStep would become dominant on the tens of millions of desktops across the world that could benefit from a graphical user interface (GUI). The initial battle for dominance gave way to attempts at agreeing on a single standard. The funny thing is, no user I ever spoke to wanted one standard GUI. Developers were hoping for one Application Programming Interface (API), however, to simplify the arduous task of porting event-driven GUI applications to different platforms.

The industry never achieved the goal of a single standard, and it doesn't look as if it ever will. But developers now have options that they didn't have a few years ago. A variety of toolkits, ranging from Solbourne's Object Interface to Neuron Data's Open Interface, allow them to work from a single code base and deliver applications that run on multiple GUIs. It wasn't so long ago that writing a GUI application for any environment, whether Windows, Motif, or Macintosh, was a grueling process, requiring hundreds of lines of C code to implement the most basic dialog box. The new tools are making the GUI part of application development a lot simpler. Since developing a portable GUI, even one that adopts the native platform's look and feel, is easy, the demand for a single GUI API has subsided.

Now, the focus must shift to user organizations that must choose a GUI standard for their architectures. Not only do they have the same choices they had three years ago, they now have combinations of choices. They can have MS Windows running in an OpenLook window on SPARC workstations, either through software or hardware emulation from SunSelect. Macintosh application code can be recompiled with libraries from Quorum to run on Unix workstations. DESQview X from Quarterdeck Office Systems can make DOS applications become X Window clients. IBM's primary selling point for OS/2 2.0 isn't Presentation Manager; it is that 2.0 runs Windows applications better than Windows.

Where does this leave users? They still need to decide whether they want an X Window-based GUI or not. If not, do they want one that is available on platforms from multiple hardware vendors, or from only one? Or do they want one that apologizes for its presence and offers another as a substitute? Now that the GUI wars are over and there is no winner, the smart user takes advantage of the range of choices that are available and (hopefully!) makes an informed decision.

X Window-based GUIs have important advantages, offering access to heterogeneous systems and applications across networks, irrespective of the operating system. But there are potential disadvantages as well. Windows offers great performance in its current incarnation, but it is far less flexible in its ability to interoperate with other systems.

Users must understand their applications and environments before making their decisions. They also need to understand the underpinnings and technological foundations of the respective GUIs as well. Since the existence of open systems means freedom of choice, it also means that choices must be made. ◐

UNIX IN THE OFFICE

Editor-in-Chief
Michael A. Goulde

MCI:
MGoulde
Internet:
mgoulde@mcimail.com

Publisher
PATRICIA B. SEYBOLD

Analysts and Editors
JUDITH R. DAVIS
ROSEMARY B. FOY
DAVID S. MARSHAK
RONNI T. MARSHAK
JOHN R. RYMER
ANDREW D. WOLFE, JR.

News Editor
DAVID S. MARSHAK

Art Director
LAURINDA P. O'CONNOR

Sales Director
RICHARD ALLSBROOK JR.

Circulation Manager
DEBORAH A. HAY

Customer Service Manager
DONALD K. BAILLARGEON

Patricia Seybold's
Office Computing 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

Unix in the Office (ISSN 0890-4685) is published monthly for $495 (US), $507 (Canada), and $519 (Foreign) per year by Patricia Seybold's Office Computing 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 *Unix in the Office*, 148 State Street, 7th Floor, Boston, MA 02109.

The X Window System

Where Is Its Future?

Introduction

The X Window System, or "X," as it is commonly called, provides a generic windowed graphics capability as a network service, accessible to any machine on the wire. In this article, we will examine the structure of the X Window System, its strengths and weaknesses, its current implementations and future directions, and we will contrast it with other window systems in order to determine whether, and under what circumstances, the X Window System is a worthwhile long-term investment in graphical user interfaces.

Why Is X Important to Open Systems?

Graphical User Interface (GUI) technology was pioneered in the late 1970s by researchers at Xerox's Palo Alto Research Center (PARC) and elsewhere, but it wasn't really popularized until Apple's introduction of the Macintosh personal computer in 1984. The Macintosh demonstrated that *windowing*—that is, the division of a display into independent, moveable regions—was a critical feature in building a graphical interface. Vendors in the Unix community began to construct GUIs also; however, they faced difficulties not encountered by personal systems vendors. Multitasking, multiple processors, proprietary graphics hardware, networks, and time-shared operation greatly complicated the task. Sun, Apollo, and others successfully sold proprietary solutions, but it took the MIT X Consortium to establish a workable, standardizable framework for Unix graphical user interfaces.

X Window System Origins

X had its origins in Project Athena at MIT, a long-term effort to develop a comprehensive heterogeneous computing environment. (The Kerberos network authentication facility is another offshoot of Project Athena.) X had its genesis when researchers Bob Schiefler and James Gettys adapted a Stanford windowing system, called *W*, to Unix. It was clear that their alterations had fundamentally changed the software; thus, they renamed it X. Versions followed in rapid succession, and industry interest grew quickly. X Version 10 was the first to be widely used in commercial products, but Version 11, or "X11," was the watershed. Each of these major versions embodied a significant change in the makeup of X; since the first release of X11, the X Consortium has made four additional minor releases, the most recent denoted X11R5. Each added functionality to the underlying X11 structure. The X Window System created a unifying force for windowed GUIs in the interoperability-conscious Unix community, and it has achieved a dominant position in the Unix market that is unlikely to be challenged anytime soon.

Refined, but Not Perfect

However, problems with X were apparent even at its introduction. It uses a bit-mapped rendering model that makes the display of graphics and text subject to wide variation on different monitors. Its graphics primitives only handle 2-D graphics. Alternate media, such as sound and video, are completely left out. Perhaps the worst problem is that its client/server architecture doesn't optimally use workstation resources in many applications, imposing performance penalties on applications and on the network.

The Structure of X

The X Window System implements a hardware-independent, networked, bit-mapped, windowed graphical user interface. It provides the basic mechanisms for partitioning a physical display into windows and for arbitrating among the various user-input devices attached to the display—keyboard, mouse, digitizer, or whatever. An X display can

simultaneously show numerous applications from anywhere on its network, as depicted in Illustration 1.

X generally operates as a networked-based software system using a client/server architecture—but perhaps the term server/client fits better. In the X environment, the locations of clients and servers are reversed from what one expects. X's network perspective led the X Window designers to consider the display device as the server; application programs using the display, therefore, are called clients, even though the user probably thinks of them as servers. This architecture is depicted in Illustration 2. As we will discuss later, this partitioning introduces some performance considerations that are related to X's "backwardness."

X Window Display with Running Programs

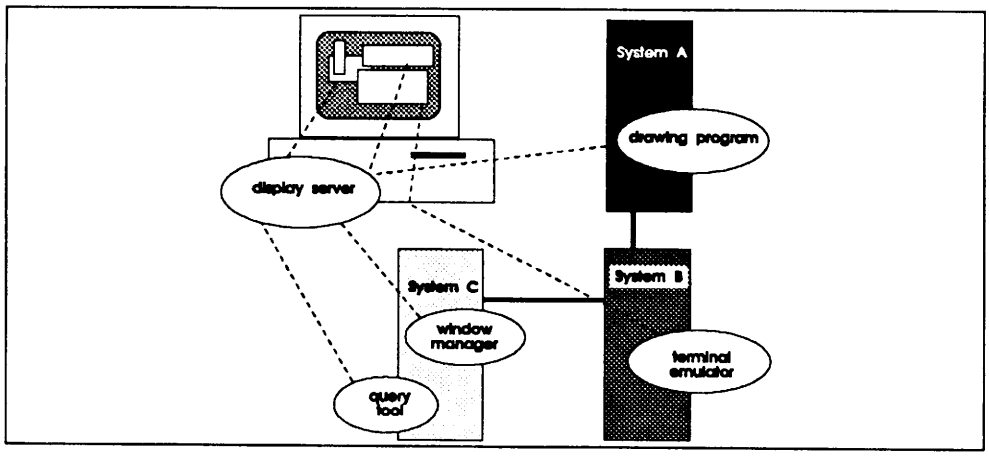


Illustration 1. A typical X Window display, showing three running programs from three remote hosts. The user has also popped up a menu against the display background; this menu is under the control of the window manager program on the host, System C.

The typical X Window client application has a very definite event-driven structure, unlike the usual nongraphical application. Instead of the ordinary subroutine calling-tree control structure, there are dozens or perhaps hundreds of user-interface events that an application may need to handle. To process these, the program must use an "event loop"—that is, a program structure that repeatedly accepts input events one at a time and dispatches them to designated event handlers.

To display objects on the screen, the program makes calls to various X Window run-time libraries. The highest level of object functionality is provided by a library called a widget set or toolkit. Widget sets implement facilities like pull-down menus and text windows. The most important widget sets are the *Xm* Motif toolkit, and the *Xol* OpenLook toolkit. At a lower level come the Intrinsic functions, which provide basic drawing and display capabilities; almost all X Window programs use the X Consortium's *Xtk* intrinsics. The lowest-level interface is *Xlib*, a subroutine library that performs the actual interaction with the server over the network connection. Illustration 3 shows the way a programmer might look at his or her X application. The main procedure (and perhaps some initialization routines) assembles a set of event handler routines, then passes control to an event loop routine. Event handlers accomplish their objectives by using the widget, intrinsics, and Xlib libraries.

X Window "Server/Client" Architecture

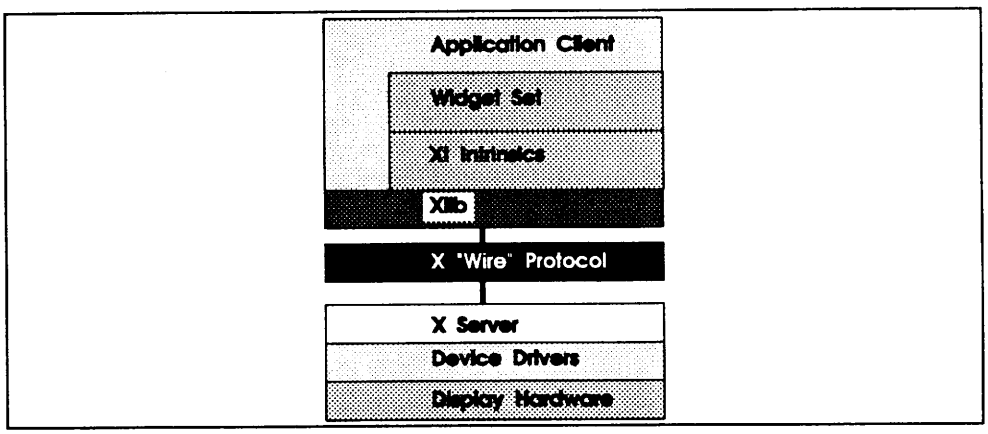


Illustration 2. Under X Window, the display hardware is owned by a display server program. This server is available throughout the network. The user accomplishes his or her work by starting up application programs that connect as clients to the display server. Typical users consider this terminology to be backwards from what they expect. Ordinarily, the desktop machine and the desktop software are considered to be "the client side," and the applications or various functional services, like database managers, are considered to be the "server side."

X Window Managers and "Look-and-Feel"

It is worth noting the Window Manager client program in Illustrations 1 and 2. Under X Window, not all clients are created equal. Rather, one program can identify itself to the display server as the Window Manager, after which it has special privileges in using the display. Its role is twofold. First, a window manager allows the user to manipulate the user interface as a whole—windows and connected applications. It provides a mechanism for manipulating the windows on the screen, allowing the user to select a window, to place and resize windows, and to close them or collapse them into iconic form. When the user has selected a window, the window manager routes subsequent input traffic to the client program. For example, when the user clicks in a word processor window and starts typing text, the keystrokes are sent to the word processing program. The window manager also allows the user to start up and to abort applications that use the display.

The second role of the Window Manager is to implement look-and-feel at a basic level. *Look-and-feel* is a hotly-debated topic in the GUI community, so it is well worth considering what it entails. *Look* is simply the appearance of the display, including decorations, button shapes, borders, and colors. *Feel* is the way the interface behaves in response to user input. There is no inherent meaning in moving the pointer to a box and clicking a button; if GUI software responds in some way to a mouse-click, it is only because the programmer has selected and implemented some meaning for it. Before we had Zoom and Iconify boxes, scrollbars, or any of the usual GUI paraphernalia, someone had to invent them. Some aspects of look-and-feel are totally out of developers' hands. The way a window is selected by the user and its basic framing, such as the title bar and close boxes, is applied by the Window Manager.

A particular GUI's look-and-feel is a (hopefully) consistent set of display ornaments, along with presentation and behavioral conventions, that allow the user to work with numerous applications in a consistent fashion. This includes menu bars, dialog boxes, alerts, scrollbars, resizing handles, text regions, and selection buttons. A GUI's look and feel is commonly specified in a style guide; this provides application developers with often elaborate rules for consistency with a GUI style.

Understanding X Window Program Structure

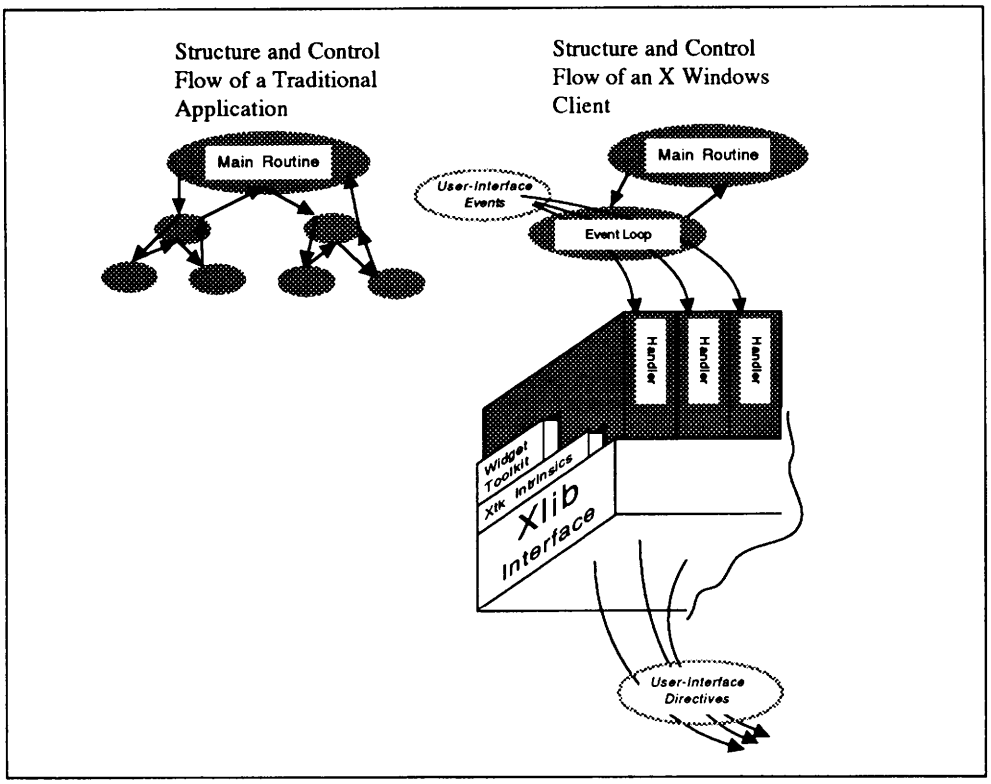


Illustration 3. In a traditional program, control passes through a hierarchy of subroutines in a more or less set order, and the program has a definite state of execution at all times. Under X, the real control flow occurs per event—the event is received by an event loop routine and passed to a handler, which executes the appropriate function and (normally) passes on its response as a directive to the display server. For precision, it should be noted that the main loop routine is normally built into the widget or intrinsics library, but it is here depicted outside that layer to illustrate that it is a high-level routine sequencing and dispatching the execution of the handler functions.

Limitations of the X Window System

The X Window System, unfortunately, has some genuine problems, but it is important to emphasize that these problems do not detract from its value as a platform for development today. This isn't just because X is the only game in town for Unix GUIs. It is also the only networkable and machine-independent windowing system available in the industry. X runs on proprietary platforms such as VAXs and Macintoshes; even IBM plans to support X (clients) on its mainframe and AS/400 product lines. X is extremely adaptable to host environments. And, finally, X has important value as a crucial standard for interoperability in graphical computing.

Limitations of the Imaging Model

X Window is explicitly designed for bit-mapped interfaces, and all of its operation assumes bit-mapped imaging. However, this poses some severe difficulties in several areas because of the significant variations among bit-mapped displays. Different vendors' displays have different pixel densities and different aspect ratios, that is, the ratio of horizontal and vertical pixel densities. To begin with, there is no such thing as an absolute square under X Window. What appears square on one display is likely to be rectangular on another. The bit-mapped imaging causes real problems with printing, since the user sees one thing on his or

her display and what is printed is almost certain to differ significantly. This problem seems to worsen as printer resolution increases. Text display is a serious problem also, because the typefaces are subject to distortion. The visual effect of a typeface is both subtle and highly sensitive to alteration. Fortunately, the most recent release of X Window, X11R5, implements a scalable outline font service. Notwithstanding this improvement, however, X remains a bit-mapped solution and obliges application programmers to program around display differences. The bit-mapped model also makes it difficult for programmers to manipulate display objects as objects; moving text or shapes is subject to "bit spill," where odd pixels on the display retain an object's color even after the object has been moved away.

Network Windowing Interface

For traditional programmers, the techniques necessary for engineering any type of graphical user interface are thoroughly alien and require a steep learning curve to acquire. Instead of a rigid and deterministic control flow, the programmer must cope with event loops, nondeterministic program states, and the need to handle innumerable display details. Unfortunately, the problem is even more difficult in the X Window environment than in personal systems. To begin with, an X Window application cannot simply reach into some device and start displaying. The user may want the application to use a display on some other machine, at the next desk or around the world. It must customize itself according to the display in use and the user's preferences. When it starts interacting with the user, it encounters an entirely new problem: To allow for heterogeneous display and client machines, X Window uses a machine-independent client/server protocol. In order to activate a pull-down menu, for example, the program must build a number of values (in the local machine format) into machine-independent toolkit structures, define the menu with those structures, and then link the pull-down menu into the overall user-interface structure. The X Window application interfaces should use a true network presentation layer to take care of machine data formats and such issues, but they do not. This is clumsy enough, but it is also often desirable to use one definition as the basis for another. One might want a whole set of push buttons in a row, with near-identical characteristics, for example. The X Window toolkits have only rudimentary support for an object-oriented approach that would provide such capabilities. These deficiencies unnecessarily complicate the programming process and give rise to inconsistency and error with X Window user interfaces.

Performance Factors

The client/server partitioning under X is an abstraction of the traditional host-terminal setup—except that the host has become the client, and the terminal, the server. Instead of exchanging characters and blocks of characters, X Window clients and servers exchange input events, such as mouse-clicks and keystrokes, plus display directives. An X client program, such as a file system browser, must handle all of its display interaction I/O. Because of this, an X Window client has *more*, rather than less, user-interface activity to handle than a character-mode application. This causes the client to use correspondingly more CPU resources and network bandwidth. An important need for users planning adoption of an X environment is to assess what level of traffic will be associated with every X Window client. It is quite possible that, for some applications, a network of X Window workstations could have poorer performance than a similar network of Windows 3.0 PCs, even if the workstations are more powerful.

X Is Susceptible to Security Breaches

The subtle difficulties of creating a networkable windowing mechanism are daunting. Unfortunately, one of the most difficult qualities to retain when using a distributed facility like X is security. In order for a program to interact with an X Window display, it needs very little information. X Window display service is provided at a public TCP socket address on each host that has a display. Each display is numbered on its host, so that, in a workstation network, anyone could expect to find displays named "systema:0," "systemb:0," "systemd:0," and so on. The X server can identify its clients only by their host name, so the only access control possible is on the basis of client host—not by the client program user's identity. Two alternate display-access control schemes are provided, but both require client-program functionality that is basically never implemented. That users can connect to other

users' displays may not seem terribly serious, but, when the display's owner doesn't know about the connection, it becomes so. Second, the unseen foreign program could completely saturate the display server with requests, essentially knocking the display out of service. Finally, the X server is quite content to dump the entire display's contents to any connected program. Thus, one of the most insidious types of invasion is one that will proceed by simply eavesdropping.

Resource Handling Is Inconsistent

One of the characteristics of a graphical user interface is that, because it intrinsically conveys and handles a lot more information than a character-based interface, there are numerous general and specific attributes of an application interface that must be customizable. Differences in keyboards, pointers, and display characteristics as well as the preferences of users or groups have to be addressed; for example, if a user has a blue display background, he or she probably wants window framing and pop-up menus to be a complementary color. Under X, all of these particulars are addressed through "resources." A resource is practically anything, so long as it has a name, but is usually attached to some component of the user interface. Background colors, for example, can be resources.

Customization is done through resource files; the normal user-customization resource file is called ".Xdefaults" and is read from the user's home directory. Each line will set some value; for example, the xterm terminal emulator can have its normal font set on one line, its boldface fonts set on another, its line-wrap behavior on yet another. Resource files can get colossal. Not only do most applications have numerous resources that can be set, but, in general, all resources for all applications are read from the single resource file. This becomes very difficult to maintain, what with allowing for host and display differences. Because of this, customization is often either skipped entirely, or else it fails to provide true interface consistency. When more versions of X11R5 become available, this will be remedied somewhat by the resource-file inclusion capability; however, this does not address the fundamental lack of structure in X Window resource handling. User interfaces are highly structured, and, without some means of addressing that structure, resource customization is likely to remain a hit-or-miss proposition.

The Great X Window Debates

The Toolkit/Look-and-Feel Debate

When Apple introduced the Macintosh, it had developed a very definite set of look-and-feel guidelines and provided run-time software (Macintosh Toolbox) to assist developers in writing to these guidelines. Over the next few years, third-party software vendors complained about the pressure Apple placed on them to adhere to these guidelines. However, as the Mac gained in popularity, it was precisely this consistency to which many attributed the Mac's success. Consequently, as the X Window System began to solidify as the basis of Unix GUIs, many recognized the need for a set of style guidelines to provide coherence and consistency to X Window applications. Such a specification had to include not only the compliant window manager, but also a toolkit of run-time routines that applications would use for their own scrollbars and so forth. Although the X Consortium at MIT provided some basics of look-and-feel, the need for more elaboration gave rise to one of the many contests between the Open Software Foundation (OSF) and Unix International: Motif versus OpenLook.

Motif Streaks to Dominance...

It seemed that Motif was a winner almost from day one. The Open Software Foundation issued one of its first Requests for Technology (RFTs) for a look-and-feel specification for X Window GUIs. The objective was to find a specification that would define a highly usable look-and-feel for X Window applications; secondarily, the OSF wanted to have a lot of commonality between the final specification and Microsoft Windows and OS/2 Presentation Manager. Two main contenders appeared: OpenLook, developed by AT&T and now owned by the spin-off Unix System Labs, and a prototype developed by Hewlett-Packard in conjunction with Microsoft; its window manager was contributed by Digital Equipment.

The latter proposal won, primarily because it met the Windows/PM-similarity requirement. The OSF consortium dubbed it Motif, and OSF members immediately announced intentions to adopt it. Since then, with only a few OpenLook holdouts, Motif has become a near-universal assumption when the topic is the X Window System.

...But Is It Really Effective?

Motif may have carried the day, but that does not exclude it from scrutiny. Essentially, the Motif definitions are long on user-interface components, but short on environment. Some Motif implementations are so bare-bones they can scarcely be called GUIs. Motif is also sketchy when it comes to defining conventions for making use of the components. The Motif Style Guide says practically nothing about when to use pop-up menus, for example. And while its members tout its Windows-like, three-dimensional appearance, there are numerous examples where official Motif style is visually confusing or ambiguous. The usual reason for this is the consistent use of stylized shading for visual indicators rather than something more direct, like OpenLook's check mark. Among these, the most consistent offender among Motif implementations is the radio button, depicted in Illustration 4 along with corresponding Macintosh and OpenLook ornaments. In our opinion, Motif's use of heavy 3-D renderings of common GUI components does not always really benefit usability.

Which Motif Button Is "On"?

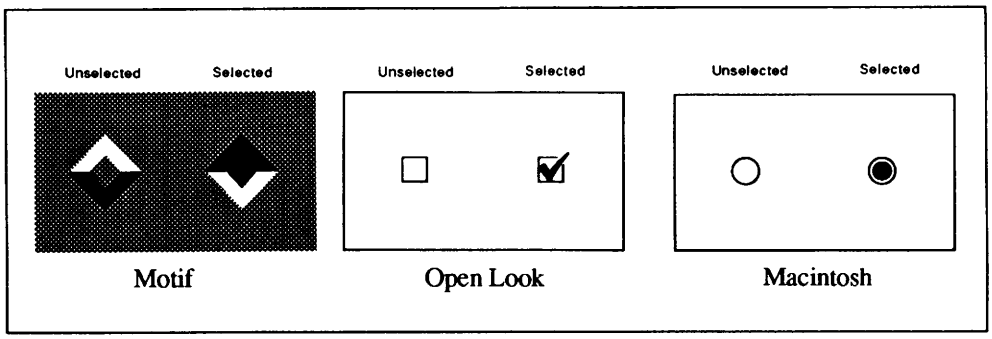


Illustration 4. HP and the Open Software Foundation put a lot of effort into Motif to make it similar to Microsoft Windows and to make it extremely visually appealing, especially through 3-D renderings. However, the artistry is often highly ambiguous. In this case, the user could easily wonder which button is actually on. Those familiar with artistic techniques would observe that on the right button, the upper shadow and bottom highlight, plus the darker inner shading, indicate it is concave—that is, pressed inwards—that is, on. Ordinary people might find it difficult to make this distinction. In contrast, the Macintosh radio button and OpenLook check box, performing the same function, are unmistakably clear.

...And How Consistent Is It?

If someone were to shop for a standard Motif workstation, he or she might conclude that there was no such thing. HP's Visual User Environment (VUE) has its Workspace Manager, while IBM has a generally plain presentation, and the Santa Cruz Operation's Open Desktop has its own highly compacted look. Menus and functionality, and even screen representations of common Motif ornaments, vary dramatically. Again, since the Motif Style Guide doesn't provide as much direction for when to use particular interface components, they may be used inconsistently.

OpenLook: An Alternative Definition

Although Motif has won the trade press war, the OpenLook definition remains a serious alternative. While Motif is certainly dominant in the breadth of its market support, OpenLook still holds a solid position as the basis of Sun Microsystems' Open Windows GUI—and Sun still ships more Unix workstations than any other vendor. OpenLook was developed by AT&T and bequeathed to Unix Systems Laboratories at the time of its spin-off. Visually, OpenLook takes a different direction from Motif/Windows or the Macintosh. It has a more open appearance than either; it makes only sparing use of 3-D-style shading, and functional decorations like scrollbars and pull-down menus are not built into the

window framing as they are under the Macintosh and Motif/Windows. Menus and dialog boxes can be fixed to the screen using a push-pin ornament; under other GUIs, these disappear after the user has made a choice. Buttons are rounded, rather than squared like those under Motif. In most cases, visual feedback to user operations is extremely clear.

This author has used Motif, OpenLook, and Macintosh GUIs, each for a year or more, and considers OpenLook superior to Motif in usability because it is uncluttered and operation is generally very straightforward. It also seems easier to handle large numbers of windows under OpenLook than under Motif. However, OpenLook also has definite drawbacks. Certain functions need revision or debugging, in particular, selection of a window for keyboard input, and icon handling. In addition, user alerts are visually obtrusive, and OpenLook has its own cases where visual cues are too subtle to allow the user to really find his or her place.

Do the Style Definitions Miss the Point?

While the Motif and OpenLook camps trumpet their accomplishments, an important component has been left out of the equation. The original GUI work at Xerox PARC made clear the importance of creating a solid metaphor by which graphical displays could simplify users' work by implementing their tasks as commonplace actions. Most X GUIs implement a means of printing by dropping one icon onto another, but this is the limit of metaphor implementation. This was the paradigm shift embodied in the Macintosh GUI—an orientation to graphical objects, rather than to functions. For example, when printing a file, the Unix user typically takes a functional command and applies it to the file, for example "`lpr file`" or "`nroff file.nr | lp`". Under an object-interface paradigm, the user chooses the file and executes its print function. Similarly, one chooses a rectangle in a drawing program and then executes the "fill" function, rather than starting a "fill" and then finding the region to fill. This type of usage, arguably the key innovation of the Macintosh GUI, is followed loosely in most X Window programs if at all. In our view, both Motif and OpenLook fail to really stress this key approach to GUI design.

Will Tool Developments Make This Debate Irrelevant?

In the last year or two, the process of X Window GUI development has evolved radically. X and the look-and-feel toolkits really provided low-level functionality, and that only through run-time libraries. Coding X applications was, at first, an excruciating process of tinkering with display parameters, positioning, and sizing, often through editing pixel values in C source code. However, several tools are now available for building GUI programs which allow the developer to edit the application's appearance graphically. These include templates for components like sliders and pull-down menus that can be inserted right into the interface and adapted for their intended use. TeleSoft's TeleUSE tool for Motif, Sun's GUIDE for OpenLook, and ICS's Xbuild are examples of such tools, which greatly accelerate the development process. However, there are also tools and mechanisms allowing a single application to be built for multiple look-and-feel specs, and even for multiple platforms; these include XVT (by XVT, Incorporated), and Solbourne's Object Interface. Such facilities make it unnecessary for developers to choose between environments, and they also allow users to choose their own look-and-feel behavior, independent of vendors.

The Desktop Debate

It should be obvious that a comprehensive graphical environment consists of more than the components discussed up to this point. Scroll boxes and text windows don't do anything by themselves. While specific applications, such as Island Draw, are fine for their purposes, it is also important to allow the naive user some means of handling all his or her files and all his or her work through the GUI. The X Window response to this need is a desktop manager program, following the example of the Macintosh Finder program.

The Desktop Debate

Is a Standard Desktop Necessary?

Relatively few hardware vendors have implemented their own desktop managers. Hewlett-Packard, Sun, and Digital all have, but most other Unix vendors ship a third party's or none at all. IBM and SCO use X.desktop from IXI, Limited. Data General ships Looking Glass from Visix, Incorporated. IXI and Visix both suggest that their products are suitable for corporate standardization as mechanisms for keeping consistency among computer sites. However, while both products have some exceptional strengths, in central aspects of functionality, they are fairly similar. And while each supports some form of drag-and-drop program operation, neither provides such a high degree of application integration as to make it a compelling option. Indeed, it is limitations in X stemming from the X Consortium's "Policy against setting policy" that make arguments about desktop managers moot.

What Developers and Users Really Need

In order to implement comprehensive desktop functionality, developers have a few critical needs. First of all, X-mediated interprocess communications must be amplified considerably. Several vendors, including HP, Sun, and Digital have developed X Window session managers in order to implement drag-and-drop between files and applications, files and objects like printers, and simply between application windows. However, there is little or no interoperability across these implementations. The X Window Inter-Client Communications Conventions Manual (ICCCM) doesn't really provide any standard way to handle drag-and-drop. Its definition of the X Window "clipboard" facility needs to be extended, including the ability to handle additional data types.

Finally, specifically in the Unix realm, it is difficult to manage files as objects when Unix files have no real typing. All of the desktop managers, whether vendors' or third parties', rely on various heuristics and half-measures to deduce the type of file that the user wants to manipulate. This makes printing and even editing a file from the desktop manager a pretty chancy operation. The uncertainty of file typing undermines the consistency of the desktop metaphor, making it difficult for users to trust the interface and use it effectively. Some standard mechanism for determining and coercing file types is really necessary to implement effective desktop managers and to significantly improve interoperability among applications.

The PC vs. X Terminal Debate

There is an ongoing debate about the relative merits of using X Terminals or PC-based X Servers. The X Terminal is often touted as a cost-effective option to the PC, offering greater security and easier management. However, for many end-user organizations, the debate is not which implementation of X Window to use, but whether to use it at all. PC desktops are still a bargain, and users are accustomed to the productivity tools under DOS, Windows, or the Macintosh. While X Terminals are less expensive than PCs, productivity applications under X severely lag PC or Macintosh environments. At least, the PC-based X server allows users the choice of accessing X applications on a server, or using their familiar PC applications.

X Loses Client/Server Benefits

Another issue that could stall X Window acceptance is the way it partitions clients and servers. In the personal systems environment, the desktop is always a client, or, to put it more directly, the end-user component of an application always runs on the desktop as a client if it is in a client/server setup. The meat of the GUI logic runs on the personal machine, while the databases and other facilities to be used *through* the GUI reside on the server. This is straightforward and also allows the desktop machine to offload the server from resource-intensive, event-driven user interface processing.

However, under X, it is more common for the client application to run on the host machine. A site may have running on one machine a database, an X Window database browser application that also runs on the database host computer, and X display servers from a number of vendors. Even though all the X display on the net can serve as a user interface for that program, the database host still has to handle a large amount of user-interface processing. Where are the benefits of client/server? Most X installations lack the balanced allocation of processing that is enjoyed in a PC client/server architecture. In fact, distributed

processing under X Window is probably more burdensome than character-mode terminal configurations and weighs down the network itself as well as the server.

Workarounds Lead to Machine Dependencies

Since it is possible for an X client and X server to run on the same machine, some of the resource burdens of X could be circumvented by having interface-handling code run on the X server workstation. However, this approach is device dependent and cannot work when the X server is running on an X terminal.

Notwithstanding these relative disadvantages, the Unix and X Window environment brings a number of advantages to the table—including interoperability, robust multitasking, and multiuser protection.

How Systems Vendors Implement—and Customize—X

Hewlett-Packard

Hewlett-Packard's X Window implementation contains some interesting addenda. HP provides a scalable typeface technology and coordinates multiple virtual displays on the same physical display. X11R5 functionality is being ported and productized. HP provides a Motif look and feel but has added a lot of value above the window system level. HP VUE provides extensive session management with the ability to keep session state between logins—all open applications and files are brought to their last state when the user logs in. A workspace manager panel at the bottom of the screen keeps important functionality in front of the user, including buttons by which the user can identify the current virtual display and change to any other.

Sun

As mentioned earlier, Sun provides its GUI, called Open Windows, under the OpenLook style specification. Sun has added PostScript windows to its X server, providing a migration path from the older PostScript-based NeWS windowing system. Sun also allows client applications a direct path to its graphics accelerators, bypassing the basic X server and protocol which are not designed to use them; applications using this facility can realize substantial performance improvements. Much of Sun's work is above server level, however. It developed a rich complement of applications and implemented some session management capabilities with Open Windows.

Digital Equipment Corporation

Digital Equipment Corporation was one of the earliest X Window implementers. Digital developed a set of tools with a distinct look and feel and packaged it as DECwindows. Although an Open Software Foundation member, Digital retains its original software as the X Window environment for its Ultrix (Unix) workstations. But on its VAXstations, running the VMS operating system, Digital now provides a full Motif implementation. Digital is a Display PostScript licensee and provides some demos and tools that employ Display PostScript. DECwindows provides a session manager, and several of its accessory programs have basic session-state maintenance.

As Digital migrates from Ultrix to OSF/1, it will provide a Motif-compliant user interface instead of DECwindows.

IBM

IBM provides a standard implementation of X Window with some extensions. Value additions to its X server include Display PostScript windows and the ability to run multiple virtual displays on the same physical display. However, the virtual displays must be opened by hand, and X must be independently started on each; there is no ability to coordinate work between the virtual displays. Multiple physical displays can also be used as a single virtual display by a single display server. IBM also is a Motif implementer and ships the Visual Edge AIC interface builder and IXI's X.desktop desktop manager with its AIXwindows product. IBM provides an extremely thorough set of file and behavior definitions in its adaptation of X.desktop.

How Systems Vendors Implement—and Customize—X

Data General

Data General is working to complete its port of X11R5 by the summer of 1992. DG's X Window implementation is fairly basic but includes a shared-memory Xlib interface. For clients running on the same machine as the display server, this greatly speeds performance by cutting out much of the Xlib encoding and TCP/IP transmission overhead. Data General is actually taking a lead on some X Window futures; it has contributed its shared-memory Xlib to the X Consortium, and is developing the sample implementation of a multithreaded X server for X11R6. (See the description of this effort below under "Ongoing X Windows Developments.") The Data General GUI includes Motif and the Looking Glass desktop manager from Visix.

Does Differentiation Make a Difference?

Sun and HP have done the most to differentiate their X-based GUIs, and yet Open Windows and HP VUE don't seem to be what drive customers to purchase Sun or HP workstations. It just doesn't seem that customers select a vendor based on their implementation of X Window. In fact, many customers purchase the platform and then purchase the GUI of choice from third parties. While special features are nice to have, third party enhancements, like those from IXI and Visix, seem to be sufficient to meet user requirements in this envrionment.

Development Directions with X—Release 5 and Beyond

Font-Imaging

Release 5 of X11 includes a font server that should solve some of the font-imaging problems encountered in prior releases. The X11R5 display server reads a path of locations to search when a client application requests a font; unlike the earlier releases, however, the locations can now include a network address as well as a file system path. The display server can connect to a font server at that address in order to get font images. The font server can handle both bit-mapped fonts and outline fonts, and will perform scaling on either type to match desired point size and screen resolution. Non-square pixel geometries can also be accommodated. This should improve font consistency considerably, and also improve the correspondence between screen and printer font renderings. However, the outline font technology is somewhat crude; unlike Adobe PostScript or Apple TrueType fonts, no scale "hinting" is provided to adapt typefaces for display at various point sizes.

Resource Management

Another X11R5 improvement is in the area of resource management. Resource files have been made more flexible, and some additional customization capabilities are provided. To begin with, resource files can now include other resource files. Also, separate groups of an application's resources can be divided into separate resource files through a revision of filename construction. This greatly eases customization, particularly when there are independent groups of customizations desired—colors, screen geometries, character sets, and so forth. Some form of resource-database structuring can be accomplished using these facilities, but we feel that the text-file mechanism for storing application data has outworn its welcome. Few users will feel comfortable with editing their resource files. If a vendor could either replace this facility or layer it with a robust graphical editor, then site-customization and personalization would become much simpler.

Device-Independent Color

Just as there is no consistent shape between two X displays, there is also no consistent color. X11R5 includes a color manager in the Xlib client library to allow applications to specify colors by absolute intensities, rather than device-specific intensities. This library uses screen characterization data stored in the server from a configuration file. Thus far, little characterization data is available, but as vendors deliver X11R5 ports, files should start appearing for various vendor displays. This capability should largely rectify color differences among X Window displays. (However, exact color display is extremely difficult to achieve and should not be expected of X Window displays, or even of most color displays used for desktop publishing.)

Development Directions with X—Release 5 and Beyond

3-D Image-Rendering

With X11R5, the PHIGS 3-D rendering standard is available under X. This effort has been underway for a number of years as PEX, that is, the PHIGS Extension to X, and the current version is the first "sample implementation." PEX is provided as a subsidiary protocol in the X server, and clients employ the run-time library PEXlib. While interesting and operational, the PEX implementation under X11R5 is not very usable. It is relatively inefficient, and little usable PEX software exists. Worst of all, the computationally-intensive 3-D graphics operations can lock up the display server for minutes at a time—not even the mouse will move. PEX needs some significant changes to make it practical, and the X Consortium has made these a priority in X Window development. The protocol and PEXlib are also being revised to meet the PHIGS+ standard and its binding in the C programming language.

Internationalization

X11R5 has internationalized X Window both in the server and the Xlib client run-time. This release supports alternate character sets for input/output and also works with ANSI C internationalization for use of language-specific error and status messages. An internationalized application operates by determining its locale or its desired language and loads resources appropriately. This actually includes loading the resource database with every string to be displayed to the user. In addition, the application must use an "input method" appropriate to the user's language. Work on internationalization is continuing at the X Consortium.

Authorization

X11R5 addresses a serious X Window problem in the mechanism for authorizing remote clients to use a display. The X Window server code has no reliable means of obtaining user identification from a client. It decides whether or not to accept a client on the basis of the host name from which the client is attaching. However, in the Unix environment, that approach is terribly unreliable—many unwanted users could run clients from an approved host. X11R5 provides two new access control mechanisms for identifying approved users, one based on DES secret-key cryptography, the other on RSA public-key cryptography as implemented in Sun Secure RPC. However, to use this, all client applications must be rebuilt to use one or both of the new access control methods, and a mechanism for integrating user identities into the X Window access control system is also needed.

Ongoing X Window Development Efforts

It seems that there will always be a new release of the X Window System under development. Release 6 is well underway with one main objective—a multithreaded X server—and a host of additional efforts, to delivery of which the X Consortium has not committed X11R6. Unfortunately, the work in progress does not completely align with the critical needs outlined earlier in this article, but it does address some other valid requirements for the X Window System. In the remainder of this section, we will survey the X projects that are in progress.

Multithreaded X Server

When PEX was implemented, the functionality was great, but the PEX server in the display server needed a lot of cycles to complete 3-D rendering operations, thereby locking up the display. The main objective of X11R6 work is to rework the X server as a multithreaded program. This would allow part of the server to plod along on a PEX operation while the rest of the server continued to service other clients and the user at the keyboard. In a similar but independent revision, the client-side Xlib run-time is being cleaned up to allow clients to run in multithreaded mode. The POSIX thread design (p-threads) is the model for this work, since the POSIX specification allows one thread to preempt another.

Inter-Client Communication Conventions Manual—Issue 2

Heretofore, vendors like Sun, Digital, and HP have been somewhat at a loss in handling important user-interface functionality like the drag-and-drop supported on the Macintosh. These three vendors all added to the basic functionality of X, but in ways that were almost guaranteed to be incompatible with work of other vendors. In fact, many of the accessory programs written by these vendors run erratically on others' platforms—or not at all. The X

Ongoing X Window Development Efforts

Consortium is addressing the needs that drove these vendors with a new revision of the Inter-Client Communication Conventions Manual (ICCCM) that addresses drag-and-drop and user session management. This document is absolutely one of the most important projects for the X Consortium.

X Window Extensions

An image extension project is underway, extending the Xlib protocol to handle fax and JPEG-compressed images, and implementing simple image manipulations in the X server. Multimedia could be facilitated by a synchronization extension prototyped at Olivetti; this would allow display activities to be synchronized with each other and with external software—sound and video drivers are the obvious first targets. A user-contributed mechanism for extending event types, available on the X11R5 tapes, is under consideration as a means of resolving all the new kinds of events with which X must deal on both the server and client sides.

Low-Bandwidth X Protocol

Taking a cue from NCD, which provides X terminals that can communicate over 9600-baud asynchronous serial lines, the X Consortium is exploring ways to create a standard lightweight X protocol to work over slow connections in the same way—even dial-up lines. Areas of exploration include intensive image compression and approaches for resource-caching.

Binding for C++

X Window programmers long frustrated by the need for object-oriented capabilities in the X toolkit may get some relief when binding for C++ comes to fruition. This work lies entirely on the client Xlib side. The X Consortium does not plan for this C++ X library to completely replace the X toolkit, but there should be significant improvements for programming complex display structures.

Conclusions

X Is Here Today

Whatever one thinks of the X Window System, it is certainly not a vaporware product. Despite its ongoing starts and stops in deployment, X is actually a fairly mature piece of software, and it meets its original requirements effectively and with good reliability. It succeeds in providing a hardware-independent facility for windowed graphical user interfaces, spanning not only CPU architectures but operating systems, and X software can be run over a network at will.

Have GUI Tools Brought X to the Threshold of Success?

A point well worth noting about X Window regards the development of X Window applications as compared to development of PC or Macintosh applications. The worst aspects of PC software development are absent from the Unix/X environment. Not only can a Unix developer run as many test programs as he or she pleases, but any or all can fail without the display or the program host crashing. Moreover, a number of excellent GUI builders for X afford high productivity in GUI programming. These factors, considered with Unix's strengths in group software development, could give rise to the development of powerful X Window applications—perhaps even the "killer application" that will bring Unix and X to dominance in the computer market.

Will Open Systems Enshrine X?

X is the only GUI that will run in native mode on heterogeneous Unix (as well as non-Unix) machines, and it supports all of the networking and interoperability features that are expected in the Unix environment. The distinct advantages of X in the vendors' battle for the desktop are directly derived from Unix: peer-to-peer networking, multiuser security, vendor-independent standardization, scalability, maturity and reliability, high flexibility, a solid development environment, and the ability to use emerging processor and display technologies quickly and effectively. These qualities are difficult to compete with.

Display PostScript—The Solution for WYSIWYG under X?

Adobe Adapted Printing Language for Active Displays

PostScript, as most of our readers know, is the printer language devised by Adobe for rendering graphics and scalable typefaces. Its first application was in the Apple LaserWriter—PostScript was essentially the final key to the Mac as a desktop publishing platform. In fact, the Mac plus the PostScript LaserWriter essentially *created* desktop publishing. However, the Mac was not able to meet the exacting type-setting requirements of commercial publications—the inconsistencies between the Mac's QuickDraw screen imaging model and the PostScript printer imaging model used in the LaserWriter would cause minor slips and overlaps not acceptable in those environments. Adobe began adapting its PostScript print imaging model for display purposes, and Display PostScript (DPS) was born. The most notable of Adobe's few Display PostScript licensees is NeXT, Incorporated. *All* imaging on the NeXT computers is done through Display PostScript, with results that are stunning and with high fidelity between displayed and printed output. Other licensees include Digital, IBM, and Silicon Graphics.

Display PostScript and X

In examining the imaging difficulties of X, what becomes apparent is that some serious sort of graphics language is needed. X drawing primitives are just that—primitive—and complex image rendering is laborious and prone to error. Display PostScript naturally fills this need. It can be implemented under X as an alternative imaging protocol for X drawing regions. The DPS protocol is an extension to the standard Xlib protocol. When a PostScript-aware X server encounters the Display PostScript protocol, it forwards it to the DPS interpreter for rendering. One might wonder why Display PostScript would be used, rather than an adaptation of, say, Hewlett-Packard's PCL language. Both are graphics languages, after all, and both started in the same printing market. However, the dominance of PostScript in the personal systems market and in high-resolution printers makes DPS a natural choice for an X Window graphics language.

Display Postscript has always been criticized for its performance because of its interpretive nature. Adobe has developed some new technology that allows an application developer to precompile routines that can be called to greatly improve performance.

Will DPS Be the Next Piece of X?

Adobe has contributed to the X Consortium a client-side library for Display PostScript. This will work with servers matching Adobe's protocol specifications for DPS, and it at least, sets up the mechanism by which vendors can begin to use DPS. However, the key part of the equation is missing: the DPS interpreter. This would ordinarily be licensed from Adobe. Alternatively, a vendor could develop its own clone DPS interpreter. This is a reasonably feasible undertaking, especially since Adobe published comprehensive specifications for the language; some printer vendors have gone this route with original (printer) PostScript. Display PostScript is also capable of handling color and, unlike the standard X facilities, is completely independent of the pixel and color characteristics of the display hardware. Does this mean DPS is a shoo-in for X Window? Perhaps, but only if a DPS server/interpreter can be obtained for the X Consortium.

Or Will Multimedia Kill It?

As valuable as they are, the virtues of Unix have not empowered the end user as much as personal systems have. The venue for the 1990s is the desktop, and, in this market, the personal systems reign supreme. Not only have millions of Apple Macintoshes been delivered, but actual users of Microsoft Windows probably now exceed the Macintosh installed base. Non-graphical PCs of the IBM architecture still number in the dozens of millions. The personal productivity revolution is continuing with no intention of waiting for Unix to catch up. Nowhere is the gap more serious than in multimedia. Unencumbered by the protections built into Unix, or by the Unix market's requirement to support multiple platforms, input and output devices by the score have been developed for the personal systems. X Window, in the meantime, is only just addressing vendor-independent mechanisms for alternate user-input devices. Nothing exists in X or in the wider Unix community that approaches a consensus on alternate output media like sound or video. It is quite legitimate to debate the real need for multimedia, but if it takes off soon, the X Window System may miss the boat.

Interoperability Improvements Will Protect Investments in X Window

The GUI wars have cooled, and no single winner is likely to emerge at this point. Most segments of the computer market have been penetrated fairly extensively by now, so no one platform is going to grab a dominant position. Market evolution is occurring far more slowly now than a few years ago. This includes the desktop market; personal systems still sell at a high rate, but the installed base is huge, slowing the rate of market development. It is difficult to imagine any platform storming the market and destroying its competition as IBM did with the IBM PC 10 years ago.

In the meantime, the improvements in interoperability that have emerged over the last few years are spreading to the GUI scene. X Window display servers are available for the Macintosh and for the PC under both DOS and Windows 3.0. The new ICCCM will improve interoperability within the X/Unix market. Numerous critical applications have migrated to Unix from the personal systems market, most notably WordPerfect and Lotus 1-2-3; among those that haven't, many can be run in DOS emulations or virtual "DOS boxes" on Unix workstations. In addition, Unix boxes can provide file service for Macs and PCs, plus gateways for electronic mail and other network facilities. In our view, interoperability has passed a critical threshold that will make it unnecessary to throw away Unix and X, even if personal systems do achieve long-term ascendancy in the desktop market.

Still the Best among Deficient Solutions

X Window can be inadequate and frustrating. It can be clunky or underpowered. But when you can bring up on one display dozens of windows from dozens of programs running on hosts all over your network—Unix and non-Unix hosts—you know that you have a facility for getting your work done, not just in the personal environment, but in the workgroup environment and the entire corporate environment. X Window has a long way to go, but, in the near-universal situation of heterogeneous networks of workstations and personal systems, it's the right solution and it's working right now.

We believe that many customers migrating to open systems will find themselves choosing MS Windows PCs running X Servers applications as their desktop of choice. MS Windows gives them a broad selection of graphical applications, along with local processing for client/server applications. Adding X gives them graphical access to information sources throughout the enterprise and beyond, irrespective of its source. ◐

Next month's *Unix in the Office* will address
Hewlett-Packard's Master Plan.

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

Open Systems: Analysis, Issues, & Opinions

VENDOR FOCUS: IBM

IBM RECASTS SNA AS AN OPEN, PEER-TO-PEER NETWORK

APPN Becomes IBM's Strategic Network Protocol

The concept that customer requirements for interoperability, especially interoperability between SAA and Unix, represent an opportunity, not a threat, has finally sunk in at IBM. Since January, the company has been at work rolling out a new positioning for Systems Networking Architecture (SNA) and Advanced Peer-to-Peer Networking (APPN) in a series of announcements that now have culminated in a "blueprint" for IBM's peer-to-peer and multiprotocol direction.

Although APPN was first announced over six years ago as a departmental networking solution, it has been growing in importance over the years. Within the new blueprint, APPN becomes one of the peer networking cornerstones in this age of open distributed systems. To underscore the importance of the recent announcements, IBM is calling them the most significant since SNA in 1974. APPN's role in IBM's blueprint is to provide the base for building the networks of the 1990s and beyond. While hierarchical SNA is still supported, it is clearly passé.

KEY POINTS. There are five major points in IBM's networking offensive:

1. IBM is licensing APPN Network Node specifications, thereby removing constraints on its availability on other vendors' platforms. If this leads to widespread support and adoption, it could lead to APPN's becoming a de facto networking standard for peer-to-peer networking.

2. The availability of APPN across all SAA and AIX platforms makes SNA networking far easier and less costly to configure and support.

3. IBM is recasting SNA as a peer-to-peer rather than a hierarchical networking scheme, making its entire network architecture less dependent on mainframes.

4. TCP/IP is now being included in SAA as part of the Common Communications Support structure, further enhancing the interoperability between SAA and Unix systems.

5. IBM is attempting to strengthen its position as a multiprotocol vendor. It has targeted concurrent support for OSI, TCP/IP, and SNA APPN, in addition to supporting Novell's IPX.

The Networking Blueprint

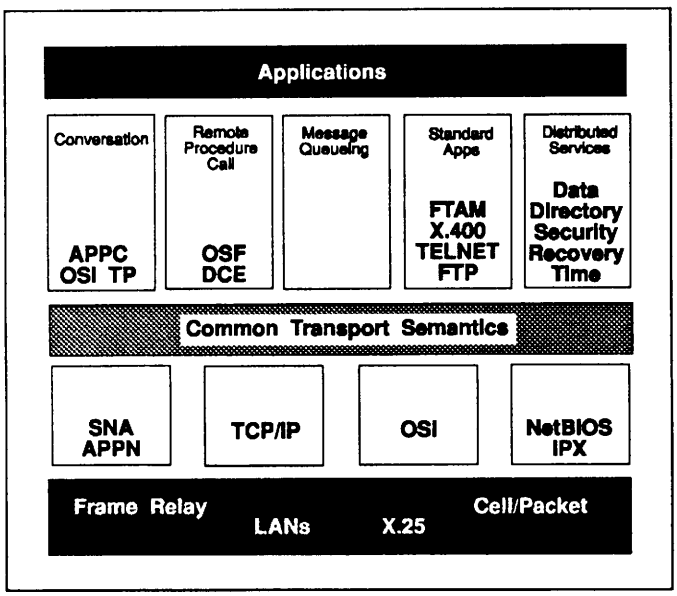
IBM's new blueprint for networking explicitly includes support for a number of de facto and de jure standards. Illustration 1 shows how all of the pieces, beginning at the physical network layer and building up to application profiles, fit together. OSF's DCE is supported as the Remote Procedure Call (RPC) application profile, while OSI TP and IBM APPC are supported for conversational applications. IBM has not identified its approach for messaging applications, although it recognizes this as an important class of applications that it does need to support. Distributed services are another area where DCE support falls within the blueprint.

FLEXIBLE APPLICATION SUPPORT. One of the major objectives of IBM's new networking blueprint is support of multiple application profiles over a consistent networking framework, offering a choice of transports for each profile. These profiles currently include:

- Conversational Applications, applications that are typified by the use of the Common Programming Interface for Communications (CPI-C) and Transaction Processing for OSI. (CPI-C is the standard programming interface for using LU6.2, aka Advanced Peer-to-Peer Communications, or APPC, functionality.) This class of applications had its origins in large computer environments and has been moving downstream to LANs.

- Remote Procedure Call (RPC) Applications, an application type that is characteristically implemented over a LAN and is used in Unix-style distributed function applications.

- Message Queuing Applications, a profile that is important in the kinds of business-critical transaction processing applications often found in industries like banking and securities.



PEER-TO-PEER ARCHITECTURE. Hierarchical SNA networks will evolve in the future to peer SNA networks built on the APPN protocol. Peer networks include Low Entry Network (LEN) nodes, which may use APPN Network Nodes for APPN support; End Nodes, which can manage local links and use a Network Node for directory and route calculation; and Network Nodes, which are fully functioning directory and routing resources. No host is required, although the most recent announcements include host systems as participants in peer-to-peer networking with APPN support in ACF/VTAM Version 4 Release 1.

OPEN LICENSING POLICY. IBM will license its APPN Network Node specification. APPN Network Node lets a variety of devices in an SNA network route SNA traffic. In hierarchical SNA, the routing function is reserved for IBM mainframes and front-end processors. IBM hopes that licensing the protocol will transform the view of APPN Network Node from a proprietary protocol to an open one that can enable other vendors to connect to SNA applications. With other vendors supplying SNA routing in their internetworking products, IBM customers will have more lower-cost options in configuring their corporate networks. This is a shift in position from last fall, when IBM said it would not license APPN Network Node.

A year ago, Apple, Novell, Siemens-Nixdorf, and Systems Strategy Incorporated had already announced plans to support APPN end node functions in their products. In addition, Novell, Incorporated, Network Equipment Technologies (NET), 3Com, and Systems Strategies, Incorporated (SSI) have now announced that they will license the APPN Network Node OEM kit. Other vendors of network hardware can be expected to rapidly follow suit. Along with Common Transport Semantics capabilities, APPN will have the ability to support development of cooperative processing applications that span IBM, Unix, and other systems.

IS THIS AN OPEN SYSTEM? Does the licensing of APPN Network Node make SNA an open system? Not in the formal sense, but it could be considered so in the same sense as NFS is considered an open system standard because Sun licenses it. The real test will be to see how many additional vendors decide to support APPN Network Node and End Node and whether it is possible to build an APPN network without any IBM product. If that occurs, then from the perspective of many customers, APPN will be an open system.

MULTIPROTOCOL ROUTING. This new networking scheme will allow SNA networks to participate in multiprotocol-router, single physical networks. Alternatively, using Common Transport Semantics, it enables simultaneous routing of OSI, TCP/IP, and SNA over APPN or TCP/IP, or other network protocols in a single physical network. By using products from third parties, APPN will allow routing of IPX, XNS, AppleTalk, and DECnet protocols as well. This capability will allow for such things as support for TCP/IP socket applications over an SNA backbone and for applications written to the CPI-C interface to run over the internet protocol (IP).

Appn Offers Advantages To Users

APPN lets various devices, such as routers, and microcomputers, minicomputers, and mainframe computers, act as peers in an SNA network. It offers dynamic configuration that can benefit existing mainframe-based networks, and APPN network-node support for multiprotocol routers should reduce the use of front-end processors at remote sites. It eliminates the need for a host by determining the best routing through the network to a resource. APPN networks can be managed using either IBM NetView or OSI network management protocols.

As a part of its recent announcements, IBM has added native LU2 routing over APPN. This means that 3270 devices can communicate over APPN protocols, offering additional flexibility. APPN support will also be added to AIX on the RS/6000.

Migrating from hierarchical SNA to APPN networks should help IBM customers upgrade work and cut down on a lot of routine weekly and monthly configuration maintenance. Older SNA networks do not provide dynamic configuration and automatic updating, and require elaborate update work for systems programmers. In addition, network users will find APPN's directory services extremely useful.

Netview Management Of APPN

IBM indicated that it plans to provide both systems and network management in both distributed and centralized environments. Each will be comprehensive, including problem, configuration, change, and performance management.

Obviously, since NetView requires a mainframe host for managing an SNA network, management of an APPN network without a host is not currently possible with NetView. IBM intends to address this within the framework of NetView Version 2 enhancements. Enhancements will include problem, topology, and accounting management that will support growth of peer-to-peer networks without affecting SNA network management.

The management system will focus on X.700 and include SNA management services, Simple Network Management Protocol (SNMP), and Common Management Information Protocol (CMIP). This implementation will be provided within IBM's SystemView framework.

Unanswered questions remain about how APPN will affect SNA's ability to provide time-sensitive responses in transaction-processing environments.

Timing

The planned rollout for the capabilities outlined in IBM's blueprint will begin in 1992, and, we believe it will occur over the next 12 to 15 months. It will be packaged in new products as well as new releases of existing products. The ability to fully implement this scheme with products in general availability will probably occur in late 1993. The blueprint itself is intended by IBM to provide a framework for products for the remainder of the decade.

Conclusion

We agree that this is the most important IBM networking announcement in nearly 20 years. Whether or not it is good depends on whether or not you were hoping that IBM would throw its whole weight behind OSI. The thrust of the blueprint is behind multiprotocol support, or in current parlance, freedom of choice. While the blueprint and product announcements do not orphan OSI, they recognize that OSI by itself is not a complete answer today and that customers want workable solutions, not solutions that are just "politically correct."

This announcement will stir a lively debate about IBM's commitment to OSI. On the one hand, IBM has always had a strong commitment to OSI in Europe, where its demand is high. From the U.S. perspective, there are too many unresolved pieces in OSI that IBM has no control over. What it does control is SNA, and that is where the company decided to place its focus in laying the groundwork for enterprise networks for the rest of the decade. If IBM customers migrate to peer-to-peer applications over SNA, it may actually help OSI by validating peer-to-peer architectures. By reinventing SNA, IBM is making a dramatic move to prevent erosion of its base of 50,000 SNA networks by Novell's IPX and TCP/IP. Whether or not OSI will suffer as result is in the hands of IBM's customers. —*M. Goulde*

VENDOR FOCUS: HEWLETT-PACKARD

Nine Vendors Join HP to PROmote PA RISC

PRO Joins East and West

Hewlett-Packard and nine industry partners announced the formation of the Precision RISC Organization (PRO) in an effort to advance and proliferate HP's Precision Architecture RISC (PA RISC) throughout the computer and electronics industries.

HP has molded a strategy that is unique when compared to that of other vendors that have formed organizations or strategic alliances to advance their processor architectures. The company has traveled around the globe selecting partners from among companies that were both interested in licensing PA RISC and would complement HP's product directions as well as those of the other PRO members. The result of this effort is a group that will use PA RISC in applications ranging from massively parallel supercomputers to automotive electronics.

HP RETAINS CONTROL While HP retains ownership of its implementation of the PA RISC architecture, the members of PRO will establish standard hardware and software interfaces to support application portability across different PA RISC products. These standards will in-

clude an Application Programming Interface (API) and an Application Binary Interface (ABI). Members will also develop procedures to contribute to the evolution of the PA RISC architecture and its related standards. Additionally, PRO members will have the ability to cross-license technology from other members and use certification and consulting services provided by the organization.

MEMBERSHIP BALANCES EAST AND WEST. There are four levels of membership in PRO with corresponding membership fees: Sponsor ($100,000), Senior ($10,000), General ($1,000), and Associate ($100). PRO's founding and sponsor members, in addition to Hewlett-Packard, are:

- Convex Computer Corporation. Convex Computer will use PA RISC in a new, massively parallel system. This manufacturer of supercomputers also has expertise in software supporting parallelism and array processing.
- Hitachi, Limited. A past licensee of PA RISC, Hitachi has been developing a range of products, from laptops to high-performance multiprocessor systems. Hitachi is also developing a version of PA RISC for embedded control applications.
- Hughes Aircraft Company. Hughes Aircraft will consider PA RISC for applications including defense electronics, satellite communications, air traffic control, and automotive products.
- Mitsubishi Electric Corporation (MELCO). The opportunity to develop software applications and to resell PA RISC workstations attracted MELCO to PRO.
- Oki Electric Industry Company. Applications that interest Oki include an embedded controller for products like telecommunications systems, automotive electronics, and factory automation.
- Prime Computer Incorporated. Now a reseller of HP 9000 systems, Prime will support PA RISC as it seeks to expand markets for its software products.
- Sequoia Systems, Incorporated. HP and Sequoia are developing a new fault-tolerant computer based on PA RISC. This will replace the existing systems that Sequoia sells to HP.
- Yokogawa Electric Corporation. Yokogawa Electric, a manufacturer of industrial automation products, will embed PA RISC in industrial control products.

WHAT HAPPENED TO SAMSUNG? One mystery surrounding the PRO announcement was, What happened to Samsung between a consultants' non-disclosure briefing on March 12 and the PRO press announcement on March 24? This $50B Korean electronics giant was on the list of PRO members that was shared with consultants two weeks before the announcement, but it was not listed among the participants at the PRO announcement. HP has indicated that Samsung was unexpectedly slower to agree to terms than the other participants. Our understanding is that Samsung is hesitant about the cross-licensing of technologies, a key part of PRO. Samsung is one of the leading semiconductor companies in the world and is apparently shy about licensing technologies in that highly competitive business. Perhaps the Koreans have less confidence in the value of this kind of business arrangement than do the participating American and Japanese companies.

PRO Differs From Other Consortia

HP's model for PRO differs from SPARC International, the ACE Initiative, and 88Open in that it is an exclusive club—membership by invitation, as opposed to the open invitation model of the others. This is because HP's objectives are different than those of the founders of the other consortia.

COMPARED TO SPARC INTERNATIONAL. By freely licensing the SPARC architecture and encouraging cloning of SparcStations, Sun hoped to proliferate its architecture and contest the Intel PC domination of the desktop market. However, none of the multitude of companies that joined SPARC International has succeeded in gaining much market share, either when measured as a share of the SPARC market or when measured against the workstation market. While there are a variety of reasons for this, one of the chief reasons was Sun's retaining control over the critical system software necessary to have a true clone market. More recently, Sun's growth has slowed dramatically, shrinking the size of the pie that SPARC cloners can try to share.

HP hopes to avoid the win-lose situation within SPARC by bringing together companies whose product strategies are essentially complementary, rather than competitive. By joining PRO, each member buys into what HP calls a win-win strategy by recognizing the strengths of the other members and drawing on those strengths rather than challenging them. All of this is informal, of course, since formally agreeing to such an arrangement would probably constitute restraint of trade.

AN ARCHITECTURE NOT A SPEC. The ACE Initiative is another open-to-all-comers consortium. It is trying to drive adoption of the MIPS architecture by systematiz-

ing the technical model of the IBM personal computer clone market by specifying a hardware standard, Advanced RISC Computer (ARC), in addition to a processor standard (MIPS). ACE hopes that the creation of such a standard, combined with system software from independent software suppliers (SCO and Microsoft), will avoid the problem that SPARC International ran into and will guarantee healthy competition, stimulate a new market, and yield a share of a new pie for everyone.

To avoid the commoditization that crept into the PC mode, ACE included in the ARC specification a Hardware Abstraction Layer (HAL), software that provides a consistent interface between the hardware and operating system. HAL allows vendors to be innovative with their hardware while ensuring operating system portability.

IBM's POWER STRATEGY. In some ways, IBM's agreements with Apple, Motorola, and Bull for licensing the Power Architecture are more similar to the HP approach than the consortium approach. IBM's agreements are thought to be much narrower in scope than PRO's charter. In IBM's case, there is no independent organization to manage the relationship, and IBM retains control of the architecture. It is not clear to what extent IBM will seek a consensus on the needs of Apple, Motorola, and Bull in evolving the Power Architecture.

PRO STANDARDIZES AT A DIFFERENT LEVEL. The goal of PRO, on the other hand, is not to create standardized hardware, although it could if members chose. Neither is its goal to create a clone market. Each member will use PA RISC in ways that uniquely meet its business objectives. Some will build laptops, others will build supercomputers. There is nothing to prevent one of the PRO members from building a workstation that complies with all of the nonarchitecture components of the ACE ARC specification.

The objective in developing an ABI is to create the opportunity for a shrinkwrapped software market for PA RISC. PRO has released an ABI based on HP/UX and will evolve that specification toward a future OSF/1 release. The API for PRO will focus on created source code portability through higher-level interface standards, such as those included in OSF's Application Environment Specification (AES), to ensure interoperability. Its concern is not operating system portability. This is in keeping with PRO's focus on a much broader range of applications than just computer systems.

LOOK TO JAPAN FOR THE PRO BUSINESS MODEL. PRO's focus is to have companies that complement one another in the marketplace cooperate in order to broadly proliferate PA RISC. The goal is to achieve a win-win situation among the members, with each achieving a strong market presence in its segment. If this strategy sounds very Japanese, it should. In a sense, PRO will act somewhat like an ad hoc MITI (the Japanese Ministry of Industry and Trade), working to ensure the success of members through cooperation and collaboration.

REVENGE OF THE NERDS. HP has been working toward building industry support for PA RISC for some time. It suffered setbacks by not being selected when Compaq and Digital were RISC shopping, primarily because HP didn't have the broad industry support that Compaq believed was required to create the volume ISVs would want. It also wasn't selected by Apple or Bull, primarily because HP wasn't the kind of partner those companies were seeking. After those defeats, one would think that HP would give up.

Think again. In a scene reminiscent of the movie "Revenge of the Nerds," HP has gone and collected a cadre of deep-pocketed partners to challenge the incrowd at their own game. PRO enables HP to claim that it has the most scalable architecture and an organizational structure that will generate benefits for all who are involved. At the same time, by keeping membership in PRO limited to complementary partners, it avoids the dilution of focus that can occur in an all-comers consortium.

WILL PRO SUCCEED? HP points to the growth in workstation shipments in 1991 as evidence for the strength of PA RISC. According to the company, its shipments grew 49 percent versus IBM's 30 percent, Digital's 8 percent and Sun's 3 percent growth for the year. If such growth continues, HP should be in the leadership position within a couple of years. HP also claims to have the leadership in commercial Unix shipments with its HP9000 Series 800 machines, with almost equally healthy growth rates. Even HP's PA RISC-based HP3000 minicomputers running the proprietary MPE/iX operating system are showing better than 5 percent growth in a stagnant market. The PRO partners obviously hope that some of HP's magic will rub off on them.

The challenges that both PRO and HP face are mostly in execution of product, marketing, and sales and support strategies. It is here that the future success of PRO and PA RISC lies. —*M. Goulde*

LETTERS TO THE EDITOR

I was amused to read your speculations as to whom MKS is licensing POSIX conforming technology in the February issue of *Unix in the Office*. Unfortunately, due to confidentiality agreements, readers will have to be content with speculation.

In that same article you made a commendable attempt to dispel the notion that POSIX is limited to Unix operating systems. Until the VMS POSIX, MPE/iX, and CTOS demonstrations at UniForum '92 brought non-UNIX implementations of POSIX to the forefront of open systems technology, the viability of such implementations *was* greeted with skepticism.

This skepticism results from an unclear view of how open systems standards are reshaping our industry. As is the case when creating most new standards, the scope of applicability for POSIX has diversified once the work began. Taking only VMS POSIX, MPE/iX, and CTOS into consideration, POSIX outside of UNIX already represents a larger installed base than the entire Unix marketplace. The trend is just beginning.

As the promise of POSIX to broaden the open systems marketplace beyond Unix continues, applications developers and end users must also alter their perceptions of what an open system is. The ability of interface specifications like POSIX and X/Open to have many underlying implementations gives it a large advantage over a description of a product like SVID. If open systems standards represent the base-level requirement to enter the market, differentiation of implementations adds competition to that market. By leveling the playing field, these standards allow the advantages of applications portability and interoperability in new markets in which Unix has not been particularly strong, such as embedded systems and fault tolerant OLTP systems.

More education is needed to help users and applications developers understand that a POSIX system is much more than Unix. This education must start from within the standards process, much of which has been dominated by system vendors who are representing only Unix interests. The interests of users, the vast majority of whom must work with the installed base of non-Unix systems, must also be represented in the standards process. Far from weakening the standard, the broadening of its applicability ensures that POSIX has a wide acceptance.

The fact that POSIX Conformance Test Suites, created by the National Institute of Standards in Technology, originally disqualified the POSIX-compliant CTOS system simply because it was not Unix, underscores the urgent need for this education.

Non-Unix systems must receive greater acknowledgment in the open systems standards community if they are to successfully compete in a market which is moving to open systems. One way of doing this is for vendors, application developers, and end-users of non-Unix systems, whether or not they have made the transition to POSIX, to become involved in the standards process.

Anyone can be a member of a POSIX working and balloting group. For information on the IEEE working and balloting groups, your readers can contact:
Secretary, IEEE Standards Board
445 Hoes Lane, P.O. Box 1331
Piscataway, NJ 08855-1331

In addition, the X/Open Company is working to develop a broader range of specifications, based on POSIX and other open systems standards, that addresses the needs for real applications and systems. Readers can contact X/Open at:
X/Open
1010 El Camino Real, Suite 380
Menlo Park, CA 94025

We have just seen the tip of the iceberg in this new trend to marry market-proven system features with the benefits of open systems standards. Whether you are a user, application developer, or system vendor keeping informed is strategically important for your business.

Randall J. Howard, President
Mortice Kern Systems Inc.

Editor's note: An often heard complaint from users is that vendors are not addressing their real business requirements as they develop standards. Vendors, on the other hand, complain that users are not sufficiently involved in the standards process. I am not sure that it is realistic for users to become involved in the technical specification work of the standards bodies. However, these groups need to have consumer interests in mind as they go about their work, and perhaps the establishment of consumer councils to oversee and advise standards bodies might be a suitable vehicle for their input. It is difficult if you are a financial services company to justify the cost of sending full time representatives to the various IEEE subcommittee meetings, but having representation in an advisory capacity designed to help ensure that standards will meet strategic IT objectives would seem to be a more reasonable expectation.

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
☐ *Paradigm Shift—Patricia Seybold's Guide to the Information Revolution*	6 issues & tapes per year	$395	$407	$419
☐ *Paradigm Shift—Patricia Seybold's Guide to the Information Revolution*	6 issues per year	$295	$307	$319

Please send me a sample of: ☐ *Network Monitor* ☐ *Office Computing Report* ☐ *Unix in the Office* ☐ *Paradigm Shift—Patricia Seybold's Guide to the Information Revolution*

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: _____

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'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.

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

IU-492

Topics covered in Patricia Seybold's Computer Industry Reports in 1991 & 1992:

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

Office Computing Report

1991—Volume 14

#	Date	Title
5	May	The Battle for LAN-Based E-Mail—Lotus, Microsoft, and WordPerfect Go Head to Head
6	June	IBM OS/2 2.0—The Quest for the Desk
7	July	End-User Information Systems—An EIS for the Rest of Us
8	Aug.	The Windows Office—Evaluating Microsoft Windows as the De Facto Desktop Office Environment
9	Sept.	Unraveling the NewWave Confusion—Differentiating the NewWave Environment from the NewWave Office from Microsoft Windows
10	Oct.	Positioning Windows Word Processors—Looking Beyond a Set of Features
11	Nov.	Keyfile—Bringing Imaging and Workflow to the Desktop
12	Dec.	IBM/Lotus Relationship—Building a Platform for Communicating Applications

1992—Volume 15

1	Jan.	The Groupware Phenomenon—Does It Focus on the Proper Issues?
2	Feb.	Digital's TeamLinks—A Renewed Focus on the Client Desktop
3	Mar.	Requirements for Workflow—What Should We Expect from the Vendors?

UNIX in the Office

1991—Volume 6

#	Date	Title
5	May	Clarity's Rapport—The Designing of an Integrating Application
6	June	Uniface—Developing Database-Independent Applications
7	July	Can Digital Become an Open Software Company?
8	Aug.	Interbase Software—Extending the Relational Model to Handle Complex Data
9	Sept.	Uniplex's New Vision—A Pragmatic Approach to the Open Office
10	Oct.	OSF's ANDF—The Key to Shrinkwrapped Software?
11	Nov.	The SQL Standard—Can It Take Us Where We Want to Go?
12	Dec.	Positioning Desktop Options—How Does Unix Fit in the Client Environment?

1992—Volume 7

1	Jan.	Downsizing with Open Systems—Can Unix Symmetric Multiprocessing Systems Meet MIS Requirements?
2	Feb.	System V.4 and OSF/1—Matching up in the Marketplace
3	Mar.	Europe's Harness Project—Integrated Technology for an Open, Object-Oriented, Distributed Applications Platform

Network Monitor

1991—Volume 6

#	Date	Title
5	May	PeerLogic PIPES Platform—Building Distributed Applications
6	June	Ellipse—LAN-Based OLTP Platform
7	July	IBM/Distributed Systems—Big Blue's Emerging Client/Server Architecture
8	Aug.	Name Services—Converging on a Two-Tier Model
9	Sept.	Common Object Request Broker—OMG's New Standard for Distributed Object Management
10	Oct.	OSF DME: The Final Selections—OSF Chooses an Object-Oriented Management Platform
11	Nov.	ANSA—A Model for Distributed Computing
12	Dec.	PowerBuilder—Graphical, Client/Server Database Applications Tool

1992—Volume 7

1	Jan.	Securing the Distributed Environment—A Question of Trust
2	Feb.	HyperDesk DOMS—A Dynamic Distributed Object Management and Applications Development System
3	Mar.	Smart Hubs—Establishing a Manageable Internet Foundation for Distributed Computing