17

VISTA User Interface Software Study

Vertical Integration of
Science, Technology, and
Applications

G. Chin

April 1990

Prepared for the U.S. Department of Energy
under Contract DE-AC06-76RLO 1830

Pacific Northwest Laboratory
Operated for the U.S. Department of Energy
by Battelle Memorial Institute

DISCLAIMER

This report was prepared as an account of work sponsored by an agency of the United States Government. Neither the United States Government nor any agency thereof, nor Battelle Memorial Institute, nor any of their employees, makes any warranty, expressed or implied, or assumes any legal liability or responsibility for the accuracy, completeness, or usefulness of any information, apparatus, product, or process disclosed, or represents that its use would not infringe privately owned rights. Reference herein to any specific commercial product, process, or service by trade name, trademark, manufacturer, or otherwise, does not necessarily constitute or imply its endorsement, recommendation, or favoring by the United States Government of any agency thereof, or Battelle Memorial Institute. The views and opinions of authors expressed herein do not necessarily state or reflect those of the United States Government or any agency thereof.

PACIFIC NORTHWEST LABORATORY
operated by
BATTELLE MEMORIAL INSTITUTE
for the
UNITED STATES DEPARTMENT OF ENERGY
under Contract DE-AC06-76RLO 1830

Printed in the United States of America

Available to DOE and DOE contractors from the
Office of Scientific and Technical Information, P.O. Box 62, Oak Ridge, TN 37831;
prices available from (615) 576-8401. FTS 626-8401.

Available to the public from the National Technical Information Service,
U.S. Department of Commerce, 5285 Port Royal Rd., Springfield, VA 22161.

NTIS Price Codes, Microfiche A01

Printed Copy

Price Code	Page Range	Price Code	Page Range
A02	1- 10	A15	326-350
A03	11- 50	A16	351-375
A04	51- 75	A17	376-400
A05	76-100	A18	401-425
A06	101-125	A19	426-450
A07	126-150	A20	451-475
A08	151-175	A21	476-500
A09	176-200	A22	501-525
A10	201-225	A23	526-550
A11	226-250	A24	551-575
A12	251-275	A25	576-600
A13	276-300	A99	601-Up
A14	301-325		

**VISTA USER INTERFACE SOFTWARE
STUDY**

G. Chin

April 1990

Prepared for
the U.S. Department of Energy
under Contract DE-AC06-76RLO 1830

Pacific Northwest Laboratory
Richland, Washington 99352

PREFACE

The Vertical Integration of Science, Technology, and Applications (VISTA) project is an initiative to employ modern information and communications technology for rapid and effective application of basic research results by end users. Developed by the Pacific Northwest Laboratory, VISTA's purpose is to develop and deploy information systems (software or software/hardware products) to broad segments of various markets. Inherent in these products would be mechanisms for accessing PNL-resident information about the problem.

As a DOE-supported laboratory, PNL generates and acquires information in certain application areas; for example, environmental modeling. This information has historically been delivered to its client and possibly published in the open literature, but in a sense the information has rarely been reused. VISTA is an effort to render such information useful in solving practical problems for other customers.

The application area selected for development of the first VISTA product is in the environmental regulation and modeling market, because of PNL's expertise and DOE's recognized needs in this area. The first product envisioned is a system that will provide users (consulting firms contracting to DOE for hazardous waste site cleanup) integrated support for performing and documenting hazardous waste site investigations, analyses, and remedial alternative selection to meet evolving environmental regulations.

The objectives of VISTA during its developing stages have been to analyze the market and specify a computer architecture that will be implemented in VISTA products. This document supports the latter objective: it is a summary of the user-interface systems that were examined as potential components of VISTA products. Other documents prepared as part of the background activities summarize the "laboratory notebook" concept of managing information, object-oriented design as a method for deriving conceptual solutions to large information processing problems, and VISTA's goals and background.

VISTA is expected to evolve as the market for VISTA applications changes and as computer systems advance. The goal of this document is to serve as a reference to the thinking that went into the developing VISTA products and to provide an information base for continued development.

SUMMARY

In support of the Vertical Integration of Science, Technology, and Applications (VISTA) project currently being directed and funded at the Pacific Northwest Laboratory (PNL), this study examined the current concepts, trends, and overall state of commercial user interface technology. The study was conducted on behalf of a component of the VISTA system architecture known as the User Interface Management System (UIMS). The VISTA UIMS manages the interaction and communications between the user and the VISTA software system. VISTA hopes to satisfy many of the requirements and functions of the VISTA UIMS by incorporating proven, commercial software products. The intent of this study is to initially comprehend the existing technology and then determine ways to best utilize it within VISTA.

User interface software may be placed within one of three classifications: window systems, user interface toolkits, or user interface management systems (UIMSs). Window systems provide basic graphics support, handle asynchronous events, and maintain window resources. Toolkits furnish prebuilt user interface components such as scroll bars and menus and provide a framework for assembling and maintaining these abstractions. UIMSs support high-level specifications of the user interface through a specification language or a set of interactive development tools. Commercial software systems are available in all three classifications. Proprietary UIMSs, however, are generally limited in selection, functionality, and reliability, owing to the relative immaturity of the UIMS concept. Because the objective of our study is to examine proven technology, the scope of this report is limited to the evaluation of window systems and user interface toolkits.

Today, the workstation window system market is dominated by two systems: the X11 Window SystemTM and the Network-extensible Window System (NeWS)TM. Both systems support a communications model that physically separates an application from the display of its user interface. An application and its display may run on separate machines that are networked together. X11 was the first of the two systems developed, is relatively simple, and has become the de facto industry standard. NeWS, on the other hand, supports sophisticated communications and graphics capabilities and is generally considered technically superior to X11. Selection of a window system should be governed by the requirements of the user interface as it pertains to capabilities, standardization, and development schedule.

Competition at the user interface toolkit level is fierce. A large number of vendors have developed toolkits for X11, and to a lesser degree, NeWS (A list of products examined is given as Table 1). Athena XtTM, probably the most popular toolkit, implements a system known as the X IntrinsicTM, which provides basic communications and framework support. Other toolkits in the market, including Hewlett-Packard (HP) CXITM, Digital Equipment Corporation (DEC) XUITM, Sony WidgetsTM, and American Telephone and Telegraph (AT&T) Xt+TM, consist of the X Intrinsic combined with a proprietary set of user interface components. In addition, Carnegie-

TM X11 Window System is a trademark of the Massachusetts Institute of Technology, Cambridge, Massachusetts.
TM Network-extensible Window System (NeWS) is a trademark of Sun Microsystems Inc., Mountain View, California.
TM Athena Xt is a trademark of the Massachusetts Institute of Technology, Cambridge, Massachusetts.
TM X Intrinsic is a trademark of the Massachusetts Institute of Technology, Cambridge, Massachusetts.
TM CXI is a trademark of Hewlett-Packard Company, Rockville, Maryland.
TM XUI is a trademark of Digital Equipment Corporation, Maynard, Massachusetts.
TM Sony Widgets is a trademark of Sony Corporation, Ichinomiya, Japan.
TM Xt+ is a trademark of American Telephone and Telegraph, Morristown, New Jersey.

Mellon University (CMU) Andrew™, Apollo Open Dialogue™, and DEC XUI provide enhanced prototyping and development capabilities by employing strategies that separate the view of user interface objects from the actual data they represent. Standardization efforts at the toolkit level are being directed by Unix International and the Open Software Foundation, two distinct organizations of computer vendors. The scopes of the two independent efforts vary. Although toolkit vendors emphasize the unique capabilities of their systems, most toolkits provide similar capabilities and support many of the same basic functions. Thus, application developers should be more concerned with understanding the technology than selecting the right product.

The complexion of user interface systems has changed drastically within the last decade. Emphasis on open, distributed development environments has led to new innovations in the design and construction of window systems and toolkits. This report serves as an introduction to the latest concepts and developments in user interface technology. The findings of this report indicate that window system technology has matured and stabilized to the point where a viable product selection may be made whereas user interface toolkit technology is still emerging and in a state of transition.

TABLE 1. User Interface Software Products

<u>Product</u>	<u>Manufacturer</u>	<u>Location</u>
X11 Window System	MIT	Cambridge, MA
NeWS	Sun Microsystems Inc.	Mountain View, CA
Athena Xt	MIT	Cambridge, MA
CXI	Hewlett-Packard Co.	Rockville, MD
New Wave™	Hewlett-Packard Co.	Rockville, MD
Andrew	Carnegie-Mellon University	Pittsburgh, PA
InterViews™	Stanford University	Stanford, CA
Open Dialogue	Apollo Computer Inc.	Chelmsford, MA
4Sight™	Silicon Graphics Inc.	Mountain View, CA
Sony Widgets	Sony Corporation	Ichinomiya, Japan
XUI	Digital Equipment Corp.	Maynard, MA
XView™	Sun Microsystems Inc.	Mountain View, CA
Xt+	AT&T	Morristown, NJ
Motif™	Open Software Foundation	Cambridge, MA
Lite™	Sun Microsystems Inc.	Mountain View, CA
NDE™	Sun Microsystems Inc.	Mountain View, CA

™ Andrew is a trademark of Carnegie-Mellon University, Pittsburgh, Pennsylvania.

™ Open Dialogue is a trademark of Apollo Computer Inc., Chelmsford, Massachusetts.

™ New Wave is a trademark of Hewlett-Packard Company, Rockville, Maryland.

™ InterViews is trademark of Stanford University, Stanford, California.

™ 4Sight is a trademark of Silicon Graphics Inc., Mountain View, California.

™ XView is a trademark of Sun Microsystems Inc., Mountain View, California.

™ Motif is a trademark of Open Software Foundation Inc., Cambridge, Massachusetts.

™ Lite is a trademark of Sun Microsystems Inc., Mountain View, California.

™ NDE is a trademark of Sun Microsystems Inc., Mountain View, California.

CONTENTS

<u>PREFACE</u>	iii
<u>SUMMARY</u>	v
<u>1.0 INTRODUCTION</u>	1
<u>2.0 USER INTERFACE PERSPECTIVES</u>	3
<u>3.0 USER INTERFACE COMPONENTS</u>	5
<u>4.0 COMMERCIAL USER INTERFACE PRODUCTS</u>	7
<u>4.1 WINDOW SYSTEMS</u>	7
4.1.1 <u>X11 Window System</u>	8
4.1.2 <u>Network-extensible Window System (NeWS)</u>	9
<u>4.2 USER INTERFACE TOOLKITS</u>	10
4.2.1 <u>Athena Xt Toolkit</u>	11
4.2.2 <u>Hewlett-Packard CXI and New Wave Toolkits</u>	12
4.2.3 <u>Carnegie-Mellon University Andrew Toolkit</u>	12
4.2.4 <u>Stanford InterViews Toolkit</u>	13
4.2.5 <u>Apollo Open Dialogue Toolkit</u>	13
4.2.6 <u>DEC XUI Toolkit and User Interface Language</u>	14
4.2.7 <u>Sun Lite Toolkit</u>	14
4.2.8 <u>Other Toolkits</u>	14
4.2.9 <u>Window System Platforms</u>	14
<u>4.3 TOOLKIT STANDARDIZATION EFFORTS</u>	15
4.3.1 <u>Open Look</u>	15
4.3.2 <u>Open Software Foundation Motif</u>	16
4.3.3 <u>Impact of User Interface Toolkit Standardization</u>	17
<u>5.0 CONCLUSIONS</u>	19
<u>6.0 REFERENCES</u>	21
<u>7.0 ADDITIONAL INFORMATION</u>	23

FIGURES

1. User Interface Reference Model	5
2. Client-Server Model	8
3. Popular Commercial User Interface Toolkits and Window Systems	10

TABLE

1. User Interface Software Products	vi
---	----

1.0 INTRODUCTION

A component of the proposed VISTA information architecture is the user interface management system (UIMS). The VISTA UIMS will act as an intermediary between the modeler and the rest of the VISTA information system. To the modeler, the UIMS represents the interface to the VISTA system and its tools and resources. To the VISTA system, the UIMS is the component that delivers user requests and textually or graphically displays requested information or computed results.

Developing effective graphical user interfaces has historically been a time-consuming and intricate task. User interface construction requires considerations in input techniques, command parsing methods, computer-human interaction, application-user interface dialogue management, and a host of other topics. A tremendous amount of effort and resources has been expended into the study of user interfaces and a variety of user interface development tools has resulted.

A goal of VISTA is to incorporate existing, commercially available user interface technology into the VISTA UIMS. Commercial systems are generally more complete, reliable, and cost-effective than software developed in-house. The objective of this report is to examine the current state of commercial user interface software and discuss the implications of selections thereof.

This report begins by describing the functionality of the user interface as it applies to users and application developers. Next, a reference model is presented defining the various operational software layers of a graphical user interface. The main body follows which examines current user interface technology by sampling a number of commercial systems. Both the window system and user interface toolkit markets are surveyed. A summary of the current technology concludes this report.

2.0 USER INTERFACE PERSPECTIVES

A user interface may be viewed from two different perspectives. In its most usual sense, a user interface defines how an application appears to the user and how the user interacts with the application to accomplish some task or solve some problem. The user interface in this context is commonly referred to as the "look and feel", because it represents the user's view of an application.

From the standpoint of an application developer, a user interface is a set of routines or components that defines or provides various graphical abstractions, interaction techniques, and communication services. The developer's task is to organize and integrate these components into a coherent interface to his application. The method by which a developer defines and interacts with user interface components is known as the application programming interface (API). Different software products have different APIs. For example, to construct a dialogue box, a given product might require the developer to place a subroutine call with a parameter list containing the desired attributes of the user interface object. Another product employing an object-oriented language might require the creation (instantiation) of a particular object. The two software products represent different methods of user interface construction, and thus, have different APIs.

Because this document focuses on the technical aspects of commercial software products, user interface issues are addressed primarily from the developer's perspective. The user's perspective, however, must be considered in any reasonable discussion of user interface concepts and is addressed where it is appropriate within this report.

3.0 USER INTERFACE COMPONENTS

User interface software may be separated into three different categories based on the level of input interactions, user interface abstractions, and general functionality. The three categories are user interface management systems, user interface toolkits, and window systems. As shown in Figure 1, the higher classifications are the farthest removed from the operating system and realize more abstract and higher-level user interface concepts. User interfaces may be constructed at any level. A developer building a user interface at a particular level may access any user interface component below it.

The window system is the component of user interface software most closely tied to the operating system. Window system functions include window resource allocation and management, drawing and text primitive processing, and input device event handling. Developing user interfaces at this level is a complex and tedious process. Graphical abstractions such as buttons and menus are not predefined and must be built up from basic drawing and low-level input routines. The developer is required to explicitly define an extensive set of attributes to describe each screen object and its inputs and handle asynchronous, multi-tasking events.

The next higher level within the user interface reference model is the toolkit. User interface toolkits implement a set of prebuilt, interactive graphical abstractions such as menus, scroll bars, and push buttons. In addition, toolkits provide what is known as the "framework" for assembling and maintaining these interactive abstractions with application modules. An element of the framework, commonly referred to as the "notifier", collects events from input devices and dispatches them to the appropriate user interface or application components (Probst 1988). A second function of the framework is to provide tools to construct new user interface components

FIGURE 1. User Interface Reference Model: Layers Represent Levels of Abstraction. Lower Layers Require a More Granular Specification of the User Interface.

from existing ones. All in all, the user interface toolkit relieves the developer from handling low-level details such as polling input devices, dispatching events, and defining the individual pieces of familiar graphical abstractions.

The UIMS is identified as the top layer of the user interface reference model. UIMSs are characterized by high-level specification of the user interface and strict separation between user interface and application (Linton et. al. 1989). These two objectives are often accomplished by means of a user interface description language which allows developers to define user interface components at an abstract level. The user interface description is then compiled and linked with a run-time library. Other UIMSs support interactive tools for defining the elements of the user interface. For example, a UIMS tool might be a visual layout editor that allows the user interface developer to move user interface objects around with the mouse, or an icon editor that maintains a square bitmap whose elements may be toggled to form a particular symbol. An objective of the UIMS is to provide the application developer with tools to rapidly construct a user interface without requiring the developer to have specific programming skills.

The UIMS is a relatively modern concept and is still a topic of current research. Most UIMSs today are either confined to academic circles or struggling for public acceptance. Such systems include NASA's Transportable Applications Environment (TAE)TM, Carnegie-Mellon University's SerpentTM, Brigham Young University's Menu Iteration Kontrol Environment (MIKE)TM, Boeing Computer Service's AxxyzTM, and Team Engineering's TIGERTM. This report maintains that current UIMS technology is too premature, unrefined, and inaccessible to meet the user interface needs of sophisticated applications and chooses not to examine existing UIMSs in detail. The separation of user interface from application, which is central to the UIMS concept, however, is still an important idea and have been realized through various strategies and to differing degrees by a number of user interface toolkits.

Before diving into the technical details of window systems and toolkits, some brief definitions of computer graphics terms may be helpful. A pixel is the smallest element of a computer screen and displays a particular color. A pixel map is the internal representation in computer memory of a rectangular area of pixels. A bitmapped or raster system maintains and organizes its display as a series of pixels. An imaging model is the method used to describe an image or display. A bitmapped imaging model assumes the display is simply a plane of pixels. Modifications to the display are achieved by applying raster operations to pixel maps. A Stencil/paint imaging model utilizes the metaphor of brushing paint over a stencil. The characteristics of the paint and stencil, such as color, pattern, and shape, define the final image (Farrell and Lavalee 1989).

The notion of window in window system carries a much broader connotation than is normally thought. A window is not just a rectilinear pane with a border and a background, but may also be one of a number of user interface components such as scroll bars, buttons, and menus. In general, windows occupy some area, receive input events, perform prescribed operations or methods, and have some visual representation. Windows or user interface components are sometimes referred to as "widgets."

TM Transportable Applications Environment (TAE) is a trademark of National Aeronautics and Space Administration, Greenbelt, Maryland.

TM Serpent is a trademark of Carnegie-Mellon University, Pittsburgh, Pennsylvania.

TM Menu Iteration Kontrol Environment (MIKE) is a trademark of Brigham Young University, Provo, Utah.

TM Axxyz is a trademark of Boeing Computer Services, Seattle, Washington.

TM TIGER is a trademark of Team Engineering Inc., Santa Cruz, California.

4.0 COMMERCIAL USER INTERFACE PRODUCTS

Since the introduction of the interactive desktop, computer systems have moved increasingly away from command-oriented user interfaces. The graphical user interface had a major impact on the workstation market as hardware vendors developed and distributed proprietary desktop environments with their computer systems. As the need for distributed, portable, and open computer systems evolved, so did the need for distributed, portable, and open user interfaces. The following survey of commercial systems examines the current gamut of user interface products and their strategies for supporting application developers in constructing graphical user interfaces.

4.1 WINDOW SYSTEMS

In first-generation user interface software, window system and toolkit functionality were frequently combined in a single windowing package. Such windowing packages are kernel-based in that both the application and the graphics display run on the same machine. In most cases, first generation windowing packages were proprietary and designed to run on particular hardware and operating system platforms. Existing first-generation window packages include Microsoft Windows™ and Presentation Manager™, Sun Microsystems SunView™, and Apple Macintosh™ Toolbox.

Second-generation window systems evolved from the demand for portable and compatible application software plus distributed windowing system capabilities. The designers of second generation window systems focused on several design goals. First, applications written on a window system should be portable to other machines and operating systems. Second, window systems should be network-based and support the client-server model. As shown in Figure 2, this model physically separates the application from its display. The client is the machine where the application is running, while the server is where the display is running. The client-server model is intended to support multiple access to applications with graphical user interfaces. Note that the relationship between the client and server is the reverse of network file systems where the server acts as the central repository and the clients are the exterior nodes. Third, window systems should be policy-free. In other words, window systems should place no policies or restrictions on the application developer as to how his user interface should look and operate. Fourth, window systems should be extensible. When additional features are defined, they may be easily incorporated into the current version of the window system without compromising compatibility with existing applications.

Most machines available today are sold with their own proprietary, kernel-based window packages, but the move toward network-based window systems is prominent. Second-generation window systems may be acquired for most UNIX®-based workstations and have been ported to a number of other operating systems such as MS-DOS™, VMS™, and MacOS™ (Macintosh Operating System).

™ Windows is a trademark of Microsoft Corporation, Redmond, Washington.

™ Presentation Manager is a trademark of Microsoft Corporation, Redmond, Washington.

™ SunView is a trademark of Sun Microsystems Inc., Mountain View, California.

™ Macintosh is a trademark of Apple Computer Inc., Cupertino, California.

® UNIX is a trademark of American Telephone and Telegraph Bell Laboratories, Murray Hill, New Jersey.

™ MS-DOS is a trademark of Microsoft Corporation, Redmond, Washington.

™ VMS is a trademark of Digital Equipment Corporation, Maynard, Massachusetts.

™ MacOS is a trademark of Apple Computer Inc., Cupertino, California.

FIGURE 2. Client-Server Model: Applications Reside on Client Machines While Displays Reside on Servers. A Display may Access More than a Single Application. Conversely, an Application may Utilize More than One Display.

Today, two network-based window systems dominate the workstation market: X11 window system and Network-extensible Window System (NeWS). Although X11 and NeWS have markedly different philosophies and capabilities, both are striving to become "open standards." These systems hope to be "open" in the sense that they are available from more than one vendor and "standard" in the sense that they are widely accepted.

4.1.1 X11 Window System

Developed at the Massachusetts' Institute of Technology (MIT) within a project named Athena, the X11 window system is the first of the two popular, network-based window systems to gain wide public acceptance. X is based on earlier research initiated at Stanford University on systems known as the V and W window systems. While at MIT, X has gone through several revisions to its current eleventh version, popularly known as X11. Throughout the development process, MIT has maintained close contact with software and hardware vendors to ensure that X11 is synchronized with the needs of industry.

Inherent in the design of X11 is the premise that a window system should provide the mechanism, not the policy, for constructing user interfaces. The original X11 designers wished to develop a window system that would support basic graphics and windowing operations, but

would not dictate how those operations were used. A second premise was to confine the capabilities of the window system to those that were understood and well-accepted. Simplicity was the key. As new features became necessary, they would be added to X11 in the form of releases. Thus, extensibility was a critical requirement of the X11 system (Myers et. al. 1988).

Adhering to the philosophy of keeping things initially simple, X11 currently supports two-dimensional graphics and a bitmapped imaging model. Some developers view X11 as functionally too primitive and restrictive to handle sophisticated graphics and real-time applications. Extensions in progress, however, include three-dimensional graphics, Display PostScript™ support, and image-processing capabilities.

MIT distributes a UNIX-based X11 implementation that runs on a number of operating systems including Unix 4.3 BSD+tahoe, Ultrix™, SunOS™, HP-UX™, Apollo Domain/IX™, IBM AOS™, and A/UX™ (Fulton 1988). Other UNIX implementations are available from a wide variety of commercial vendors. In addition, X11 has been ported to a number of other operating systems including MS-DOS, VMS, MacOS, and Lisp environments. X11 accessibility is further complemented by the introduction of inexpensive, X11 display terminals that download the server code from a computer or store the server code in ROM (Read-Only Memory) chips. Such systems attach to a host computer via serial connections or Ethernet. The price of X11 display terminals ranges from \$1400 to \$3000.

Because of X11's accessibility, portability, basic capabilities, and close relationship with industry, X11 has become the de facto standard in the UNIX community. In official circles, the American National Standards Institute (ANSI) and the International Standards Organization (ISO) are in the process of defining X11 protocol standards within formal committees (Myers et. al. 1988).

4.1.2 Network-extensible Window System (NeWS)

In 1986, Sun Microsystems introduced NeWS to challenge X11's rise toward the industry standard. Sun's initial bid to corner a significant portion of the window system market was largely unsuccessful, owing to X11's popularity and momentum and technical problems with the NeWS product arising from its hurried release into the market. The advanced capabilities of NeWS, however, could not be overlooked. Given time, marketing, and further technical refinements, NeWS has finally emerged with a significant following of its own.

Like X11, NeWS is a network-based window system that supports basic graphics and windowing capabilities. Again, the philosophy is to provide the mechanisms for constructing user interfaces, but not the policy. The strength of NeWS, however, lies in its use of PostScript™ as both an extension language and a stencil/paint imaging model. In X11, the client communicates an image to the server in terms of predefined commands or operations. The protocol is static.

™ Display PostScript is a trademark of Adobe Systems Inc., Palo Alto, California.

™ Ultrix is a trademark of Digital Equipment Corporation, Maynard, Massachusetts.

™ SunOS is a trademark of Sun Microsystems Inc., Mountain View, California.

™ HP-UX is a trademark of Hewlett-Packard Company, Rockville, Maryland.

™ Domain/IX is a trademark of Apollo Computer Inc., Chelmsford, Massachusetts.

™ AOS is a trademark of International Business Machines Corporation, Armonk, New York.

™ A/UX is a trademark of Apple Computer Inc., Cupertino, California.

™ PostScript is a trademark of Adobe Systems Inc., Palo Alto, California.

NeWS, on the other hand, utilizes the power of a page description language to perform this same function.

The advantages derived from the PostScript protocol are numerous. First of all, it supports extensibility. Extensions are easily incorporated into NeWS by writing PostScript codes. Secondly, applications may interactively download PostScript routines onto the server. For example, if an application is required to repetitively draw a complex figure, the application may pass a routine that draws the image to the server. The figure may be regenerated by invoking the downloaded subroutine rather than respecifying the individual elements of the complex figure. In addition, application-specific data formats may be defined for communications between client and server. Finally, system performance may be enhanced by localizing network-intensive operations within the server (Leffler 1988). For example, a potentially expensive operation is rubber-banding which connects the cursor to a fixed point by drawing a line between them. As the cursor moves, the line or rubber-band stretches or compresses to maintain the connection. A rubber-banding operation in X11 would require continuous polling and feedback between the application and user interface to ensure that the cursor position and the rubber-band depiction transformed accordingly. In NeWS, a routine may be written that handles the rubber-band/cursor interaction and placed entirely within the server. This feature provides the application programmer flexibility to minimize the amount of network traffic and optimize performance.

Sun has conceded to X11's dominance as the industry standard and is now providing both X11 and NeWS support for its workstations. This support comes in the form of a merged X11/NeWS window system that is capable of simultaneously displaying windows from both window systems. The merged window system is currently available through Sun as part of a product known as OpenWindows™. In addition, American Telephone and Telegraph (AT&T) has pledged to provide both X11 and NeWS support in UNIX System V.4.

4.2 USER INTERFACE TOOLKITS

In contrast to the limited competition within the window system market, there is a proliferation of commercial user interface toolkits. The more popular commercial toolkits and the window systems on which they are built are illustrated in Figure 3. Figure 3 also illustrates a great disparity between the number of X11 and NeWS toolkits. The imbalance emphasizes the extent to which X11 has become the de facto industry standard.

MIT Athena Xt	HP CXI	HP New Wave	CMU Andrew	Stanford Inter- views	Apollo Open Dialogue	DEC XUI	Sony Widgets	SGI 4Sight	Sun XView	AT&T Xt+	OSF Motif	Sun Lite	Sun NDE
X11 Window System												NeWS	

FIGURE 3. Popular Commercial User Interface Toolkits and Window Systems. Toolkits Delimited by Dashed Lines are Targets of Standardization Efforts.

™ OpenWindows is a trademark of Sun Microsystems Inc., Mountain View, California.

Toolkits inherit many features from the underlying window system. For instance, a toolkit is only as portable and extensible as its window system allows. All toolkits perform the two basic functions of providing a framework and prebuilt user interface components. The distinction between competing toolkits comes in two forms. First of all, a toolkit may provide additional high-level programming tools to aid program development. Often, these tools are similar to those provided by UIMSs. For example, a common user interface toolkit utility is a user interface description language that allows developers to specify the user interface outside of the application. Another form of distinction is within the set of prebuilt user interface components itself. Different toolkits support different types of user interface abstractions.

One possible explanation for the amount of user interface toolkit competition relates to the subjective nature of these abstractions. The toolkit is the level of the user interface reference model where the look and feel of an application is actually realized. Since look and feel is an inexact, subjective science, it is reasonable to expect different vendors to have different definitions of a good user interface. Toolkits represent a means of conveying, packaging, and promoting proprietary user interface preferences and concepts.

4.2.1 Athena Xt Toolkit

The developers of X11 foresaw the need for a higher-level programming interface and set of tools than that provided by the X11 window system. As a result, the task of developing the Xt toolkit was incorporated into MIT's Project Athena. One component of the Xt toolkit, known as the X Intrinsics, supports a well-defined framework and various communication services. Specific functions of the X Intrinsics include the provision of mechanisms for creating and registering user interface components, translation of input events, binding of event sequences to application modules, and limited geometry management of windows. Furthermore, the Xt toolkit provides a set of prebuilt user interface components such as buttons, scroll bars, labels, and dialogue boxes. The Xt toolkit is part of MIT's X11 distribution package.

X11's philosophy of separating mechanism from policy trickled up to the Xt toolkit. The appearance and behavior of user interface components are defined outside of the X Intrinsics and are encapsulated within the Xt widget set. This philosophy and design has allowed the X Intrinsics to evolve into a standard of its own. Many commercial toolkits simply supply a set of proprietary, prebuilt user interface components to realize a particular look and feel, and rely on the X Intrinsics to provide framework and communications support.

The X Intrinsics allow "vendors to develop their own user interfaces on a standard (and therefore portable) platform" (O'Reilly, 1989). The separation of widgets from the X Intrinsics allows toolkit developers to focus more closely on the details of the user interface and less on underlying programming and communication mechanisms. In theory, a developer may intermix widgets from different Intrinsics-based toolkits. As the number of Intrinsics-based toolkits grows, so does the variety of user interface abstractions made available to application developers. Some toolkits, however, are not built on the X Intrinsics and are, by and large, less portable, standardized, and extensible than their Intrinsics-based counterparts.

4.2.2 Hewlett-Packard CXI and New Wave Toolkits

Hewlett-Packard was one of the first companies to introduce a commercial X toolkit. A toolkit named XRay™ was developed for the X10 window system™. More recently, HP's CXI toolkit was developed to run on the X11 window system. CXI is built on the X Intrinsics and contains an extensive set of widgets that support features such as pull-down menus and a unique 3-dimensional window appearance. A less powerful version of CXI is available through MIT's X11 distribution package while the complete CXI toolkit is commercially available through HP.

New Wave™ represents HP's next generation of X11 toolkits. HP's close working relationship with Microsoft is quite apparent in the design of the New Wave product. The look and feel is similar to that of Microsoft's Presentation Manager™, which is a user interface toolkit/window system combination developed for the IBM OS/2™ operating system (Hewlett-Packard 1989). As of late, New Wave and Presentation Manager have been the focus of tremendous controversy. Apple computer has filed a copyright infringement suit against Microsoft and HP claiming that both Presentation Manager and New Wave misappropriately incorporate copyrighted user interface work. Until the suit is resolved, the legality and future of New Wave and Presentation Manager are uncertain.

4.2.3 Carnegie-Mellon University Andrew Toolkit

The Andrew toolkit exists as a component of the larger Andrew application environment currently used to network machines at Carnegie-Mellon University. Originally built on the Andrew window system, the Andrew toolkit has been ported to X11 in conformance to the industry standard. The Andrew toolkit supports its own framework and communication services, and is not based on the X Intrinsics. The Andrew toolkit is available through MIT's X11 distribution package.

The most notable aspect of the Andrew toolkit is its support of the "Data-View" architecture. This architecture is based largely on the Modeler-View-Controller model, which is the backbone of Xerox's Smalltalk™ object-oriented programming environment. With the Andrew toolkit, multiple views may be generated for a single data object. Multiple views within the same user interface provide alternative visual representations of the same data. Operations on one representation will manipulate the model and automatically be reflected in the remaining views (Probst 1988).

The "Data-View" architecture is supported by a logical separation of user interface and data objects. Multiple views are achieved by forming connections between a single data object and several view objects. This architecture also supports multiple user interfaces to a single application. Multiple user interface implementations may address the needs of different levels of users or provide a diverse sampling of look and feels from which user's preferences may be evaluated. Prototyping capabilities are enhanced, because modifications to the user interface have minimal effect on the actual data or application.

™ XRay is a trademark of Hewlett-Packard Company, Chelmsford, Massachusetts.

™ X10 Window System is a trademark of the Massachusetts Institute of Technology, Cambridge, Massachusetts.

™ New Wave is a trademark of Hewlett-Packard Company, Chelmsford, Massachusetts.

™ Presentation Manager is a trademark of Microsoft Corporation, Redmond, Washington.

™ OS/2 is a trademark of International Business Machines Corporation, Armonk, New York.

™ Smalltalk is a trademark of Xerox Corporation, Palo Alto, California.

4.2.4 Stanford InterViews Toolkit

InterViews is another X11 toolkit that bypasses the X Intrinsics layer and connects directly to the window system. Developed at Stanford University, InterViews is primarily a research system confined to an academic environment. The toolkit, however, is available on MIT's X11 distribution package and has some unique features that deserve attention.

The InterViews toolkit was developed in C++ and adheres to object-oriented methodology. Basic user interface and application components are defined as interactive and abstract objects respectively. Specialized or complex user interface components may be defined by combining existing interactive objects by composition. The notion of interactive and abstract objects effectively separates the context of user interface and application components. From the perspective that abstract objects encapsulate application data and interactive objects may represent views to that data, InterViews supports functionality similar to that of Andrew's "Data-View" model. Linking multiple, interactive viewing objects to an abstract object is analogous to defining multiple views to encapsulated data. Again, multiple views allow application developers to construct different user interfaces to applications and different visual references to data objects (Linton et. al. 1989).

4.2.5 Apollo Open Dialogue Toolkit

Open Dialogue is built on the X11 window system but is not based on the X Intrinsics. Apollo is promoting Open Dialogue as a user interface management system (UIMS) because the product employs a description language that allows specification of the user interface outside of the application (Apollo 1989). The user interface specification is compiled and then linked with a runtime library. In more traditional toolkits, user interface components are accessed via procedural calls within the actual application modules. Although the intermediate description language is a powerful mechanism, Open Dialogue does not come with a large assortment of user interface development tools which are characteristic of UIMSs. Several other commercial toolkits provide description languages similar to Open Dialogue's. Although Apollo markets Open Dialogue as a UIMS, the product's functionality is comparable to a number of existing commercial toolkits.

Like Andrew, Open Dialogue enforces the separation of the user interface from the application. The user interface description language facilitates this feature by localizing all user interface components and their connections to application modules in a separate file. The logical separation provides functionality as was mentioned with the Andrew toolkit. Unlike Andrew, however, the separation is also a physical one and provides Open Dialogue with additional features not realized by Andrew.

User interface prototyping in Open Dialogue is simple and quick. A new user interface is defined by describing user interface components and their connections in a file. No modifications are required to the actual application. In fact, the user interface may be constructed before the application is even developed. This convenient packaging of the user interface also facilitates easy transfer of user interface designs from one application to another. Furthermore, a set of user interface specification files may be defined to serve as generic templates for constructing user interfaces to similar applications or enforcing a particular look and feel among all applications.

4.2.6 DEC XUI Toolkit and User Interface Language

Running primarily on DEC workstations, DECwindows™ is a distributed computing environment that supports basic computing functions such as networking, graphics and application development. XUI is an X11-based toolkit embedded within DECwindows and built on the X Intrinsics. A feature of the DECwindows environment which augments the functionality of the toolkit is the User Interface Language (UIL). Similar to Open Dialogue's user interface description language, UIL provides separation of user interface from application, enhances prototyping capabilities, allows multiple user interfaces, and simplifies transferability of user interfaces between applications.

Most toolkits make liberal use of windows when defining user interface components such as buttons, menus, and scroll bars. This practice is viable as long as windows are considered inexpensive resources. The developers of XUI, however, realized that a great number of windows may actually degrade the performance of a window system. This revelation led to the conception of "gadgets", which allow developers to realize user interface components by painting them instead of creating additional windows. Purportedly, the performance of gadgets exceeds that of more traditional approaches (O'Reilly 1989).

4.2.7 Sun Lite Toolkit

The Lite toolkit was developed by Sun Microsystems to run on NeWS. Making use of PostScript's flexibility, user interface components in Lite are written in PostScript and then downloaded onto the NeWS server. Unlike some X11 toolkits, Lite does not shield the developer from the capabilities of the underlying window system. Window server programmability remains fully intact.

In developing Lite, Sun wished to deliver a toolkit that could support both experienced and casual toolkit programmers. Lite employs an inheritance-based classing system and provides a basic set of abstract superclasses. Programmers wishing to construct complex user interface objects may do so by means of subclassing. Casual toolkit programmers may use Lite's default mechanisms to utilize predefined user interface components. Furthermore, Lite provides access to many of the low-level facilities of NeWS such as lightweight processes, programmable events, and window resources (Myers et. al. 1988). Lite attempts to give application programmers access to the full functionality of the window system plus the features of more traditional toolkits.

4.2.8 Other Toolkits

Two products yet to be mentioned are 4Sight and the Sony Widgets. 4Sight was developed by Silicon Graphics to run on NeWS. The toolkit was developed to specifically support Silicon Graphics's high-end, Iris class, graphics workstations. Sony has developed an extensive set of widgets which are built on the X Intrinsics. The Sony Widgets are available on MIT's distribution tape.

4.2.9 Window System Platforms

The toolkits so far described are among the most widely accepted and commercialized. Incidentally, most of them are built on X11. As for NeWS, Sun and a number of other toolkit

™ DECwindows is a trademark of Digital Equipment Corporation, Maynard, Massachusetts.

developers have had little success in promoting NeWS toolkits outside of their organizations. One apparent reason for this failure is NeWS' lack of an elaborate distribution network. For the cost of the media, MIT provides a distribution tape containing the X11 window system and Athena toolkit, plus contributed software from commercial vendors and academia. Contributed software consists of additional X11 accessories such as toolkits, widgets, window managers, and text fonts. A group, known as the X Consortium, controls the contents of the distribution tape, circulates patches to detected X11 bugs, and answers questions on X11. Sun distributes NeWS also at a minimal cost, but the distribution tape does not contain a plethora of additional software.

4.3 TOOLKIT STANDARDIZATION EFFORTS

Today, user interface standardization is a controversial issue within the computer industry. Standardization efforts are being directed by two different organizations of computer vendors who both proclaim their charters to be the development and promotion of standards-based, open application environments for computer systems. The two organizations, Unix International and Open Software Foundation (OSF), have essentially divided the computer world in half, as both sport an impressive list of corporate supporters. OSF garners support from the likes of IBM, DEC, and HP, while Unix International members include AT&T, Sun Microsystems, and Intel. Although both organizations have received tremendous verbal support from its members, it is unclear, at this time, whether their members will abide by the standards these bodies set forth.

Unix International and OSF have defined their initial user interface systems and are actively promoting their user interface standards. Unix International is endorsing a user interface known as Open Look while OSF is promoting a user interface package known as Motif. The philosophies and scopes of these two competing systems differ significantly.

4.3.1 Open Look

Open Look is a look and feel specification developed by Sun Microsystems for AT&T. Open Look was developed from scratch with several key design goals in mind. The most obvious objective was to define a look and feel which incorporated good visual design. Sun employed graphic artists within the design team to ensure that aesthetic issues were addressed. A second design goal was to achieve some balance between "simplicity, consistency, and efficiency" (Hoerber 1988). The user interface should be simple to use, consistent across applications, and efficient in minimizing required user interaction. Often, these goals will conflict. In addition, Sun did not wish Open Look to deviate significantly from existing look and feels. Interoperability with existing systems would promote a smoother transition towards the Open Look standard. The final objective was to develop a user interface independent of particular hardware characteristics. For example, Open Look visuals must work consistently on all displays regardless of dimensions, resolution, and color support. Furthermore, Open Look must allow for a variable number of mouse buttons and keyboard function keys (Hoerber 1988).

An interesting aspect of Open Look is that it is not really a tangible product. It is merely a specification or set of guidelines for defining what an application user interface should look like and how the user should interact with one. Open Look presumes nothing about the implementation details of how the user interface is actually realized. The task of developing toolkits for different computing platforms is left to computer companies and third-party software vendors. The desired scenario is one of vendors developing a number of different Open Look toolkits with the identical look and feel.

The first three Open Look toolkits have been defined by Sun and AT&T. Sun is developing the XView and NDE toolkits which will be built on X11 and NeWS respectively. Recall that Open Look makes no assumptions about the implementation details of the toolkit, and thus, makes no assumptions about the underlying window system. AT&T is developing Xt+ which is built on X11. With XView, Sun is attempting to lure existing programmers and applications towards Open Look by defining an API similar to that of SunView. SunView is a kernel-based window system/toolkit which supports the current line of Sun workstations. The same enticement holds true for Xt+ which has an API based on that of MIT's Athena toolkit (Probst 1988). Both XView and Xt+ will be packaged as part of MIT's X11 distribution tape. Originally, the two toolkits were also intended to be distributed as part of AT&T's UNIX System V.4 release. Recent rumors, however, suggest that AT&T may hedge on that commitment (Cortese 1989). The NDE toolkit will be distributed as part of Sun's NeWS Development Environment package.

4.3.2 Open Software Foundation Motif

In reaction to the Open Look standardization effort, OSF set forth to define a user interface standard of its own. In November of 1988, OSF issued a formal request for user interface technology. Among the 39 entries it received, OSF selected a hybrid of three submissions to promote as an industry standard. The hybrid was named Motif and consisted of DEC XUT's application programming interface and user interface language, HP New Wave's icons and 3D decorations, and Microsoft Presentation Manager's look and feel. Unlike Open Look, Motif is based primarily on existing systems. The represented companies in the hybrid, incidently, are the three largest of OSF members (Rauch-Hindin 1989).

OSF's selection of the Presentation Manager's look and feel is designed to attract the support of the PC community and to include these often-neglected users and developers in user interface standardization efforts. Presentation Manager serves the function of both a window system and a user interface toolkit for the IBM OS/2 operating system. In addition, the selection of XUT's API and user interface language provides Motif with a large, immediate base of DECwindows programmers and applications. DECwindows is Digital's own proprietary desktop environment.

The rigidity of the Open Look and Motif specifications are significantly different. Open Look promotes the strict specification of a particular look and feel across applications. Each user interface component takes on a specific form and function (Hoerber 1989). Motif, however, focuses less on the first and more on the latter half of the look and feel equation. The Motif specification does not concentrate on what user interface components look like but rather how those components operate. Three basic principles apply. First, similar user interface controls should have similar operations and usages. For example, pull-down, pop-up, and cascading menus are similar controls which should perform similar functions. Second, the same user action should always incur the same result. For instance, clicking the select mouse button in a particular area of the scroll bar should always produce the same effect. Last, the mouse cursor should be manipulated by the actions of the user and not positioned arbitrarily by the application (Open Software Foundation 1989b). These principles enforce how the user interacts with a user interface but not how he sees the user interface.

Unlike Open Look, Motif is not simply a look and feel specification but a concrete, tangible system. This represents a markedly bolder approach than Open Look's. Where AT&T and Sun deliberately confined standardization efforts to look and feel, OSF opted to provide a toolkit, window manager, and presentation description language as part of its standards specification

(Open Software Foundation 1989b). Although some of these utilities are also available with the Open Look interface, they are not part of the Open Look specification. Motif, on the other hand, bundles these utilities into a single package and distributes it as a complete user environment. The user environment defines not only the look and feel interface but also the application programming interface. Motif is part of a larger standards specification known as AES (Application Environment Specification), which addresses other application interface areas including operating systems, network services, graphics services, database management services, and programming languages (Open Software Foundation 1989a). With AES, OSF has the unenviable task of promoting an all-encompassing standard in a field which cannot even agree on the number of buttons to put on a mouse.

4.3.3 Impact of User Interface Toolkit Standardization

The greatest advantage of user interface standardization is the notion of user portability. A consistent user interface across different platforms will allow users to transfer their experiences from application to application and machine to machine. Apple Macintosh software demonstrates this capability well. Apple requires Macintosh software developers to follow a particular user interface style. As a result, when a Macintosh user moves from one application to another, he will find similar user interface abstractions and modes of operation. The user needs not relearn how to interact with the computer.

Commercial standardization of window systems was easy. Window system functionality was well understood and agreed upon. The question was not what to implement but how. The standardization of user interface toolkits, however, has a totally different complexion. Re-emphasizing the subjectivity of user interface design, there are guidelines but no clear-cut rules for the specification of user interface components. Furthermore, competitive market forces act against standardization. Fierce competition encourages the development of products which are easily distinguishable from others. Toolkit designers respond with distinctive look and feels.

5.0 CONCLUSIONS

To support user interfaces in today's world of distributed computing, two window systems have emerged with the necessary technology and public acceptance. The window systems, X11 and NeWS, address different user interface development needs. Application developers requiring high performance and sophisticated capabilities are best suited by NeWS. Developers desiring formal standardization and a large, existing, window system application base should move toward X11. Recall, however, that extensions such as 3-dimensional graphics and PostScript support are currently under implementation for X11. NeWS may also be extended to support additional capabilities, and features will certainly be added to it as well. Thus, functionality issues boil down to the question of not which features window systems provide but when those features will be provided. Criteria governing the selection of window systems should include standardization concerns, current window system capabilities, and the timeliness of future extensions.

In today's competitive toolkit market, no single or set of toolkits stands out as the de facto industry standard, nor is any likely to in the near future. From the end user's standpoint, this lack of a standard implies that users will continue to see inconsistent look and feels across applications and machines. From the developer's standpoint, this implies that developers must continue placing blind faith in the toolkits on which they build. There is no assurance that any particular toolkit can survive the dynamic and evolving nature of the business.

Despite these words of discouragement, users and developers alike will cope as they always have in a field where technology grows faster than it can be absorbed. In fact, many users and developers feel that the lack of toolkit standardization is not a critical dilemma. One software engineer attending the Special Interest Group for Computer-Human Interaction (SIGCHI) Conference in Austin commented that the current spectrum of toolkits in the market represents a twenty percent variation in functionality. The intent of the comment was to point out that all toolkits provide a basic set of familiar and intuitive user interface components, and that the metaphors represented by the components are clear enough to dispel any minor differences in look and feel. As for application programming interfaces, all toolkits adhere to and have structures reflecting basic object-oriented design principles. Application developers familiar with object-oriented or toolkit programming should have limited difficulty in porting source code from one toolkit to another.

Solving modern problems requires modern tools. Knowledge and understanding of the current concepts and technology is essential in finding effective, contemporary solutions. Many programs today do not keep pace with the evolving industry. As a result, many computer systems are obsolete from their inception. They incorporate outdated concepts, duplicate existing technology, and are frequently less reliable. VISTA hopes to avoid this dilemma. The intent of this study is to acquire the necessary background to make informed decisions on user interface issues and discover ways to utilize that technology to our benefit. In the simplest terms, we must comprehend the technology before we can effectively employ it.

6.0 REFERENCES

- Apollo Computer Inc. 1989. *Apollo Open Dialogue*. Chelmsford, Massachusetts.
- Cortese, A. February 5, 1989. "AT&T's Open Look Support Vanishing." *Computerworld*, p. 6.
- Farrell, J., and D. A. Lavalee. 1989. "User Interfaces in Window Systems: Architecture and Implementation." In *SIGCHI '89 Course Proceedings*. SIGCHI Conference, April 30-May 4, 1989, Austin, Texas.
- Fulton, J. 1989. *X Window System, Version 11 Release 3: Release Notes*. X Consortium, MIT Laboratory of Computer Science, Boston, Massachusetts.
- Hewlett-Packard Company. 1989. *Leadership in User Interface Technology*. Rockville, Maryland.
- Hoeber, T. Autumn 1989. "Open Look Design Goals." *SunTechnology*, pp. 63-75.
- Leffler, S. J. June 1988. "A Window of the Future?" *Unix Review*, pp. 62-69.
- Linton, M. A., J. M. Vlissides, and P. R. Calder. February 1989. "Composing User Interfaces with InterViews." *Computer*, pp. 8-22.
- Myers, B. A., A. Schulert, S. Wallace, O. Densmore, and D. Goldsmith. 1988. "User Interface Toolkits: Present and Future." In *SIGGRAPH '88 Panel Proceedings*. SIGGRAPH Conference, August 1-5, 1988, Atlanta, Georgia.
- O'Reilly, T. February 1989. "The Toolkits (and Politics) of X Windows." *Unix World*, pp. 66-73.
- Open Software Foundation. 1989a. *Application Environment Specification (AES). User Environment Volume*. Cambridge, Massachusetts.
- Open Software Foundation. 1989b. *OSF/Motif Style Guide*. Cambridge, Massachusetts.
- Probst, R. Autumn 1988. "Open Look Toolkits." *SunTechnology*, pp. 76-86.
- Rauch-Hindin, W. February 20, 1989. "Is Motif the Best OSF could have Done?" *Unix Today*, pp. 23-24.
- Rosenthal, D. S. June 1988. "Going for Baroque." *Unix Review*, pp. 71-79.

7.0 ADDITIONAL INFORMATION

- Asente, P. September 1988. "Simplicity and Productivity." *Unix Review*, pp.57-63.
- Ballou, M. December 5, 1988. "New Windows Open Up." *Digital Review*, pp. 77-80.
- Binding, C. *The Architecture of a User Interface Toolkit*. Olivetti Research Center, Menlo Park, California.
- Bunshaft, A. J., R. Puk, G. D. Laib, and S. Abi-Ezzi. 1988. "Extending Graphics Standards to Meet Industry Requirements." In *SIGGRAPH '88 Panel Proceedings*. SIGGRAPH Conference, August 1-5, 1988, Atlanta, Georgia.
- Buxton, W., R. Baecker, and B. Myers. 1987. "Human-Computer Interaction: Selected Topics." In *SIGGRAPH '87 Course Proceedings*. SIGGRAPH Conference, July 27-31, 1987, Anaheim, California.
- Chandler, D. June 1988. "Going to Market." *Unix Review*, pp. 32-45.
- Grimes, J., S. Bancroft, W. Mackay, A. Marcus, W. Teitelman, and J. Veeder. 1988. "User Interface Considerations of Windowing Systems." In *SIGGRAPH '88 Course Proceedings*. SIGGRAPH Conference, August 1-5, 1988, Atlanta, Georgia.
- Hanner, M. A. January 1989. "Gambling on Window Systems." *Unix Review*, pp. 51-59.
- Kannegaard, J. Autumn 1988. "Open Look: Industry Outlook/Overview." *SunTechnology*, pp. 58-62.
- Lee, E. June 1988. "Window of Opportunity." *Unix Review*, pp. 47-61.
- McCormack, J., and P. Asente. 1988. *An Overview of the X Toolkit*. Digital Equipment Corporation, Western Research Laboratory.
- Olsen Jr., D. R. 1988. "User Interface Management Systems." In *SIGGRAPH '88 Course Proceedings*. SIGGRAPH Conference, August 1-5, 1988, Atlanta, Georgia.
- Sussman, A. January 23, 1989. "OSF UEC Wins Early Thumbs Up." *Unix Today*, p. 5.
- Swick, R. R. and T. Weissman. 1988. *X Toolkit Athena Widgets - C Language Interface*. Digital Equipment Corporation, Western Software Laboratory.

DISTRIBUTION

No. of
Copies

OFFSITE

2 DOE/Office of Scientific and Technical
Information

ONSITE

2 DOE Richland Operations Office

D.K. Jones
D. E. Trader

43 Pacific Northwest Laboratory

K.J. Adams
G.D. Black
S.F. Bobrowski
J.C. Bower
G. Chin (10)
A.L. Corrigan
P.J. Cowley
C.R. Cole
C.E. Cowan
V.L. Crow
B.A. Garrett
C.A. Geffen
L.H. Gerhardstein
M.J. Graham (5)
B.L. Hoopes
E.W. Kleckner
J.R. Lewis
W.D. Meitzler
M. Olander
J.O. Schroeder
M.R. Siegel
J T. Thomas
M.B. Walter
M.A. Whiting
Publishing Coordination
Technical Report Files (5)