INTRODUCTION

Graphical user interfaces (GUIs) and windowing systems have received widespread industry attention and are the focus of intense competition between vendor coalitions. As users demand better and increasingly more intuitive interfaces to UNIX operating systems, they also require software applications like the SAS System to take advantage of and support these interfaces. In addition to the requirement for GUI support, interoperability will also be a decisive factor in computing in the 1990's. Organizations are and will continue relying on more than one vendor's architecture. Management concerns about fragmentation of resources and data plus issues of information sharing and resource sharing will affect users' hardware and software procurements. A distributed computing environment supporting industry standard GUIs and software applications that look and feel the same regardless of the hardware platform comprise the model many organizations will strive to achieve. With Release 6.07 under UNIX, the SAS System is strategically positioned to fulfill both mandates: native support for popular GUIs and MultiVendor Architecture" (MVA) to provide support for distributed computing, and interoperability across supported platforms. The objectives of this paper are twofold: to introduce concepts and definitions of GUIs and distributed computing environments and examine the X Window System in the context of computing trends and benefits to users as well as the role windowing systems play in creating a multivendor distributed environment; and to illustrate the flexible structure used to implement the OPEN LOOK™ and OSF/Motif™ interfaces to the SAS System.

THE X WINDOW SYSTEM, GRAPHICAL USER INTERFACES, AND DISTRIBUTED COMPUTING:

Why Is the X Window System a Key Technology?

The X Window System is an operating system independent, hardware independent, network-based windowing protocol that meets several key requirements of a distributed network computing environment. It enables easy access to applications such as the SAS System that can be executed on multivendor systems in different locations on a network. X fosters the development of devices that can transparently access distributed computing resources regardless of the location on a network. Users are no longer restricted to a single vendor for hardware or a single operating system. Many sites currently utilize X technology to achieve interoperable heterogeneous networks.

The X Window System offers a client/server model that separates the input/output portion of an application which allows physically independent application processing and display processing. As a result, not only can an application program operate on one system on the network while the I/O portion runs elsewhere (on an X display terminal, a PC, or a workstation), hardware functioning as an X server can simultaneously execute multiple sessions by providing many windows that operate as separate terminals.

X does not depend inherently on any specific operating system. In addition, it is unbiased towards any specific GUI. Key to the design of X is its open, vendor-neutral standard. As a result of this design, X has gained widespread industry acceptance by both hardware and software vendors and end users. In addition, X enjoys little major competition as a networking window system standard. For vendors and users alike, a standard, vendor-independent windowing protocol is key to the development and implementation of heterogeneous, distributed computing networks. Therefore, the X Window System has the necessary design features and industry and vendor support to be accepted as the standard windowing protocol for distributed network computing.

What Is a Graphical User Interface (GUI)?

A GUI is a computer interface technology that is visually oriented instead of character oriented. GUIs use bit-mapped graphics, windowing, icons, and visually dynamic menus to communicate with users. Users can accomplish tasks much more quickly and easily and can be less concerned with syntax and finding commands. GUIs give users a more productive and natural way to interact with computers. GUIs provide an intuitive vehicle for people to interact with computers, thereby enabling users to be more productive in less time and with less training. As a result, many users in the UNIX market are attracted to a platform that was once considered highly technical. In addition, the dominant trend is the adoption of X and the use of GUIs based on X as the windowing system of choice. X has made it possible for several GUIs to be developed and employed on a wide variety of platforms and networks. At present, no single GUI dominates the UNIX market. However, the two numerically dominant contenders in the UNIX market are OSF/Motif, offered by the Open Software Foundation and favored by Hewlett Packard, IBM Corporation, Digital Equipment Corporation, and Microsoft Corporation; and OPEN LOOK, supported by AT&T and Sun Microsystems and offered by Unix International. With Release 6.07, the SAS System supports both OSF/Motif and OPEN LOOK. Significant GUIs not based on X include the Apple Macintosh environment, Microsoft Windows 3.0 in the DOS environment, and Presentation Manager (PM) under OS/2. With Release 6.06 under OS/2 the SAS System supports Presentation Manager. Current plans call for support of Microsoft Windows.

How Have User Interfaces Evolved with Hardware Advances into Graphical User Interfaces?

User interfaces have existed since the beginning of computing history. When mainframes dominated computing in the 1950s and early 1960s, batch computing was the primary method of processing users' tasks. Intermediate layers of personnel and resource control isolated end-users from computers. Then hardware advances signaled the advent of the minicomputer in the mid 1960s to early 1970s. Many more organizations were able to afford the technology offered by this less expensive class of hardware. Time-sharing lowered the cost of computing for many users; however, a typical user interface was still character-oriented. In the mid 1970s to early 1980s, the microprocessor increased the entry-level cost of computing dipped low enough to be afforded by individuals. Most consumer interfaces were designed around character-based, line-oriented, monochrome video displays.

The 1980s brought less expensive memory prices and more powerful microprocessors. Workstations emerged as a new hardware platform designed to perform certain classes of technical applica-
tions, such as computer-aided design. Workstations were also networked to share data and resources. Typical early workstation displays had fairly high resolution with some gray scale or color where necessary. With windowing capabilities and pointing devices such as a mouse, these systems supported GUIs that provided users with more sophisticated means of interaction than previous systems. Advances in semiconductor technology led to increased computing capabilities by the end of the 1980s. Among the significant developments were the following:

- faster computing technology, such as reduced-instruction-set-computing (RISC) microprocessors;
- advances in complex-instruction-set computing (CISC) processor technology, such as combining CPU, memory management, and arithmetic processing on one chip;
- dedicated graphics processors;
- larger and faster memory chips and video-specialized memory;
- higher-resolution displays.

As history illustrates, decreasing costs in entry-level use of new technology creates a canvas for the emergence of a wide range of applications and new markets. Through technological breakthroughs, not only are GUIs and applications now more affordable, they are advantageous when compared to character-based interfaces.

Distributed Computing and the X Window System

Much of the success of the X Window System can be attributed to its inherent support of networking. The market for X will continue to grow due to the ability to put together functionally homogeneous distributed systems composed of heterogeneous elements. A network in its most basic form is a collection of computing resources tied together by communications channels in a way that allows easy transfer of information between resources. Distributed computing networks allow access in a transparent manner to resources on a network regardless of the location, the vendor, and the operating system of the resources.

For example, consider a scenario that may be a typical distributed computing model. The average user has a workstation. The machine is configured with no disk storage and minimum memory. Also present at the site is a mainframe that stores a number of large data sets as well as a DB2® database. Also available is a departmental UNIX server that has a larger, faster CPU than the local workstation and ample disk storage available. All three computers are on a network. The SAS System is present on the UNIX server and the mainframe.

The user of the local workstation has the capability to access the departmental UNIX server through the X Window protocol and the network. Therefore, it is possible for the user of the workstation to use the processing and storage capabilities of the UNIX server. Because the UNIX server is also connected to the mainframe, the user can use the "SAS/CONNECT®" software to download an extract data set from DB2. The downloaded DB2 extract can then be processed transparently by the SAS System on the departmental server by the user of the local workstation.

This environment in place, the user of the diskless workstation is running an application that provides a means of display called an X server or server. The diskless node acts as a graphics display device with network support. The server sends different types of messages that collectively are called the X protocol. In this example, the X protocol messages are sent across the network and received by the X client, which is the X Window System running on the UNIX server.

A client is an application program running on a host anywhere on a network. Because the local node has the capabilities to act as an X server, the user can access X client applications (for example SAS software applications) that are executed on a remote-networked system, in this case the UNIX server. Because the X Window System is network based and uses the client/server model, users on a network can invoke and execute applications such as the SAS System, from hardware that is different and separate from the machine hosting the SAS System. For application software like the SAS System to perform as X clients in a distributed computing environment, Release 6.07 of the SAS System (6.07) is written to comply with the operating schema and I/O conventions of the X Window System.

Release 6.07 users of the SAS System on a UNIX platform can use a distributed network to access other platforms running Version 6 of the SAS System such as MVS, VM, VMS, or OS/2. Therefore, distributed computing allows for more efficient use of resources at a site, less reliance on any one vendor, greater range of available resources, easier expansion, and corresponding economic benefits. The MVA® design of Version 6 of the SAS System provides SAS users with the necessary software ingredients and capabilities for a distributed computing environment. All the fundamental technology elements for distributed network computing and high-performance X Window displays are now readily available. Adoption of distributed computing is no longer limited by economic considerations or restricted by limits in technology. Rather, the critical factors will be standards deployment and acceptance, increased management awareness of the advantages of distributed computing and GUIs, effective network management software, and availability of applications such as the SAS System.

THE IMPLEMENTATION OF SAS 6.07 UNDER X

The second part of this paper illustrates the design used for the user interface portion of the SAS System, Release 6.07, and the benefits of designing the user interface this way.

Problems in developing a User Interface

The user interface part of application is a key component of an application. This is where first impressions are made, and a user's attitude about the user interface can affect the evaluation of the rest of the product. A good interface is an asset; a weak user interface can hamper an otherwise excellent product.

In addition, user interfaces are expensive to develop and maintain. Different operating systems provide different user interfaces, and user interface technology has changed rapidly in recent years. An application must stay current with new technology if it is to be competitive.

A Solution through Modular Design

One of the benefits of a network is the ability to upgrade individual components with little or no impact on the rest of the system. There is a corresponding idea in software design. As a program is designed, groups of functions are isolated into modules, with well-defined interfaces between modules. With this design, individual components can be enhanced or even replaced with minimal impact on the rest of the program.

Release 6.07 of the SAS System uses such a design; a structure was chosen that simplifies the implementation of multiple user interfaces. This is similar to the MultiVendor Architecture used in Version 6. Since the focus of this paper is on the X-based interfaces under Release 6.07, a brief overview of the X Window System is given below. This is followed by a brief review of the MVA design philoso-
The most basic outlined. The client and mouse that is used to communicate software developers do not have to worry about the protocol directly. Instead, they use functions from one or more libraries that provide them with a higher-level interface.

The Xlib library provides low level functions. These include primitives for connecting to the server, creating windows, drawing lines, and rendering text and graphics.

Most X clients also use other libraries, or toolkits to simplify program design and development. As applications have become more powerful, they have become more complex, and are rarely viewed simply as lines on a screen. Instead, the design constructs used are the ones that people interact with, such as menus, push buttons, and scroll bars. These are collectively referred to as widgets.

A toolkit provides functions needed to create and manage widgets. It is the widget set (and, therefore, the toolkit) that gives an application a particular style. For example, under OSF/Motif, menus are pulled down from a single action bar at the top of a window. Under OPEN LOOK, the menu buttons provide the same function in a different way.

Both of the X interfaces in Release 6.07 use a toolkit based on the Xt toolkit intrinsics, or Xt library. The intrinsics provide an abstract view of the functions needed to use widgets. The Xt library, like Xlib, provides functions that are independent of any particular style. These functions provide a bridge between Xlib and the widget set, and manage things such as widget creation and destruction, keyboard translations, X resources, and inter-client communication.

The functions in the widget library allow a programmer to create widgets, such as a scrolled text window, without worrying about drawing each of the lines in the scroll bars, or drawing each character as it is displayed.

The three X libraries make up a layered design. The Xlib library is used with any X-based program. The Xt intrinsics library defines another layer, and the widget set used specifies a particular visual style and behavior.

THE MULTIVENDOR ARCHITECTURE OF VERSION 6

One of the goals of distributed computing is to provide systems and solutions that are independent of any particular vendor. The Multi-Vendor Architecture allows Version 6 of the SAS System to meet this goal - Version 6 is the same whether you use it under DOS on a PC, MVS on a mainframe, or on a UNIX workstation.

The layering idea is key in the MVA design. The current MVA design defines three distinct layers. The top layer contains the applications code; procedure writers use an API that frees them from many of the details of file management and the user interface. The second layer is usually thought of as a portable supervisor. It provides services for the applications layer, but does so in a way that is independent of any particular hardware or operating system. The first two layers are completely portable, and account for a major portion of the code used in Version 6.

The bottom layer connects the portable supervisor to the host operating system. The specifics of memory management, input/output, and numeric calculations are done here. This allows for simplified design within the various components. Where possible, the code can be optimized for a particular host.

Extending MVA

Version 6 of the SAS System is implemented not only on a wide variety of hosts but also under a variety of user interfaces. The SAS Display Manager provides its own windowing system on terminals such as the IBM 3270 and the DEC VT100. There are also implementations under native windowing systems, including Microsoft Windows, Presentation Manager on the IBM PC, DECwindows under VMS, and OPEN LOOK and OSF/Motif under UNIX and other platforms.

Since it seemed to work so well for X11 and for the main part of the SAS System, a layered design was used for the user interface portion of Release 6.07. Layers were identified that would allow for the sharing of code across different GUIs, without sacrificing efficiency.

The three layers used in Release 6.07 for UNIX are as follows:

1. A common UI layer contains functions that are portable to any user interface, whether text-based or graphics-based. Functions here include things such as basic initialization and reading in PMENUs from catalogs.
2. The X common layer is used with any X-based user interface. The graphics portion of the Release 6.07 X interface uses only Xlib functions, and is therefore portable to any GUI that uses Xlib.
3. The Xt intrinsics library contains functions to keyboard translations, resource management, and other widget-related utility functions. The third layer in the Release 6.07 X interface is the code that uses these functions, but does not reference any GUI-specific widgets.
4. In the topmost the code focuses on using a specific widget set. Thus, there is one layer for the OSF/Motif interface, and another for the OPEN LOOK interface. Since the OSF/Motif interface is based on the Xt intrinsics, we chose the OPEN LOOK intrinsics toolkit (OLIT) to provide an X interface with the OPEN LOOK style.

This layered, modular design provides an improved user interface for Release 6.07. Implementing additional user interfaces, or updating existing ones, is much easier because much of the code can be reused. This will allow the SAS System for UNIX to adapt with the industry, as user interface technology continues to evolve.