NAT'L INST. OF STAND & TECH R.I.C.
A11103 802998

NISTIR 4876

NIST
PUBLICATIONS

An Introduction to Graphical User Interfaces and Their Use by CITIS

Susan Q. Sherrick

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Gaithersburg, MD 20899

QC
100
.U56
4876
1992
C.2

NIST

 ASSA ABLOY-HID EXHIBIT 1018

QC100
.U56
4876
1992
c.2

NISTIR 4876

An Introduction to Graphical User Interfaces and Their Use by CITIS

Susan Q. Sherrick

U.S. DEPARTMENT OF COMMERCE
Technology Administration
National Institute of Standards
and Technology
Computer Systems Laboratory
Gaithersburg, MD 20899

July 1992

U.S. DEPARTMENT OF COMMERCE
Barbara Hackman Franklin, Secretary

TECHNOLOGY ADMINISTRATION
Robert M. White, Under Secretary for Technology

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
John W. Lyons, Director

AN INTRODUCTION TO GRAPHICAL USER INTERFACES AND THEIR USE BY CITIS

by
Susan Q. Sherrick
National Institute of Standards and Technology
October 15, 1991

I would like to acknowledge John Barkley and Roy Morgan for their contributions and assistance.

ABSTRACT

A Graphical User Interface (GUI) is a powerful tool that is used for simplifying a computing environment. This paper provides a tutorial on the various meanings of the term GUI, describes the usefulness of GUIs, identifies problems with GUIs, and recommends that the X Window System GUI be used within the CITIS specification.

The term "GUI" can have various meanings in different contexts. In order to provide a framework for discussion in this paper, a GUI is characterized as having at least one of the following components:

1. Display Manager Program,
2. Application User Interface,
3. Application Programmer's Interface (API).

One of the major problems with GUIs is the potential lack of portability of an application developed to run using a particular GUI. The X Window System provides a means of overcoming this problem because applications developed using the X Window API may be run using almost all other GUIs. Consequently, the X Window System can be used by CITIS to provide easy to use applications which may be used on almost all GUI platforms.

TABLE OF CONTENTS

I. PURPOSE 1

II. BACKGROUND 1
A. CITIS 1
B. GRAPHICAL USER INTERFACES 2

III. DISCUSSION 3
A. WHAT IS A GUI? 3
1. Display Manager Program 3
2. Application User Interface 4
3. Application Programmer's Interface . . 5
B. WHY USE GUIs? 6
C. PROBLEMS WITH GUIs 7
D. USEFULNESS OF GUIs WITH CITIS 8
E. STATE OF GUI TECHNOLOGY 9

IV. NATIONAL/INTERNATIONAL STANDARDS VS. EXISTING DEFACTO STANDARDS 12

V. RECOMMENDATIONS 13

VI. CONCLUSION 14

VII. REFERENCES 15

VIII. BIBLIOGRAPHY 15

THE USE OF GRAPHICAL USER INTERFACES BY CITIS

I. PURPOSE.

This report is a Computer-aided Acquisition and Logistic Support (CALS) Initiative deliverable in response to task 3.3 (National Institute of Standards and Technology Statement of Work dated February 25, 1991). It contains a discussion on Graphical User Interfaces (GUIs) in general, with specific focus on two areas:

1. Options for GUI use within the Contractor Integrated Technical Information Service (CITIS) specification; and

2. Appropriateness of using national/international standards versus existing defacto standards.

Because of the nature of this report, it is necessary to mention vendors and commercial products. The presence or absence of a particular trade name product does not imply criticism or endorsement by NIST.

II. BACKGROUND.

A. CITIS.

Section 6 of Draft MIL-STD-CITIS states:

> This standard is designed to be incorporated into a contract to define the functional requirements for a computer-based service to provide access to integrated information. CITIS is intended to be an efficient, contractually implementable means for providing the Government with on-line access to contractor generated data and the digital interchange of such data. Ultimately, CITIS will replace most, if not all, contractor delivery of hardcopy information currently required by the Government to manage contractor generated data throughout the life-cycle of the program. The requirements are specified in terms of information services and information functions which must be selected to meet the needs of each application. [MIL-STD-CITIS].

The goal is to allow contractor business and technical information to be generated once, reside in one place, and assist DOD

acquisition managers in gaining automated access to this information. As a result, this effort integrates resources, thereby reducing replication of storage and applications, and eliminates increases in information deliverable costs by replacing paper with automated access and delivery services.

B. Graphical User Interfaces (GUIs).

GUIs have become pervasive on both personal computers (PCs) and workstations. GUIs are powerful tools for the end-user and the application program developer. For the end-user, a GUI provides easier access to computing resources. The end-user accesses computing resources by manipulating symbolic representations of the services provided by those computing resources. With a GUI, the end-user is no longer restricted to typing commands one line at a time. For the application developer, a GUI provides the tools for developing such easy to use applications.

Although GUIs are powerful tools for both the end-user and the application developer, portability problems may arise. As a practical consideration, a GUI may be strongly associated with a particular hardware platform. This is not to say GUIs require specific hardware. Although the performance of some GUIs can be improved by the addition of optional hardware, popular GUIs are all software implementations and can, in theory, be implemented on any hardware platform. Nevertheless, this strong association between a hardware platform and a GUI exists. For example, Microsoft Windows is associated with the Intel (IBM compatibles) PC platform. Open Look from Sun/AT&T is associated with the SPARC platform. Motif is associated with platforms that support OSF, such as IBM Risc systems, but is also available on Sun Sparcstations.

Thus, there are several GUIs in common use and each may be implemented on different hardware platforms. Therein lies the portability problem, which affects both end-users and application developers.

The end-user is affected by the different "look and feel" of the different GUIs. Although each GUI makes computer use easier for the end-user, most end-users select a preferred GUI and remain loyal to that one. In addition, the application developer is affected by the different APIs that are usually associated with a GUI. For example, an application developed using the Microsoft Windows API must be modified to use the Open Look API.

There are solutions to the portability problem for both the end-user and application developer. These solutions are discussed in a later section. In order to discuss these solutions, a framework for the meanings of the term GUI is presented. The term GUI can have different meanings depending upon the context of the discussion. For the purposes of this paper, a GUI is characterized as consisting of one or more of the following components:

1. Display Manager Program
2. Application User Interface
3. Application Programmer's Interface (API).

This characterization should aid in explaining the multiple meanings of the term GUI and enable readers to distinguish which meaning is being discussed in other written or spoken material. The characterization described is not intended to follow a characterization used by any particular existing GUI and is presented as a tutorial device to aid in understanding the meanings of the term GUI and as a framework for later discussion.

III. DISCUSSION.

A. What is a GUI?

The term GUI is commonly used to refer to any of the components of a graphical user interface between a user and his computing environment. For this paper, a GUI is characterized as consisting of at least one, and usually more, of the following components:

1. **Display Manager Program** is often referred to as a "desktop" display. This interface handles the interaction between the user and computing services. It provides control of how applications are arranged and re-arranged on the screen, how the user migrates between applications, and how applications communicate with each other. The desktop display is the overall "look and feel" of the system, or the user's view of the computer environment. Users are shielded from hardware and

software specifics about the computer environment and are able to maneuver freely about their system using the desktop display. Pictured below is an example of the Display Manager Program on a Macintosh. All Display Manager Programs provide a similar view of computing resources.

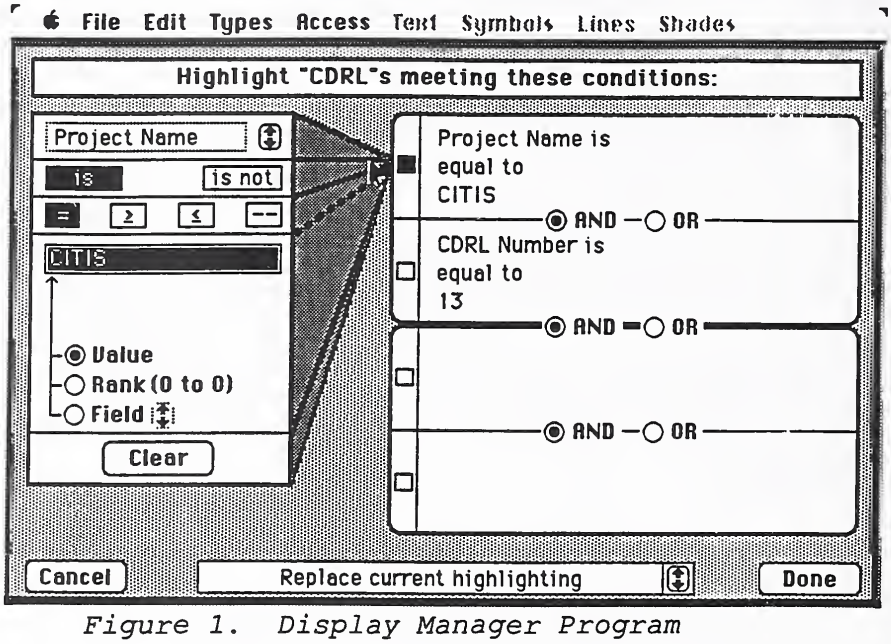

Figure 1. Display Manager Program

2. **Application User Interface** defines the interaction between the user and an application program. It determines the overall visual appearance specific to an application. The application user interface may differ from one application program to another within the same system environment, depending upon the software developer. This may cause confusion and force the user to learn a new interface for each application program. Consistency between application interfaces allows a user to explore and comfortably predict how a new application will behave. There are "style guides" available for application programmers which provide guidelines for developing consistent application user interfaces. In addition to style guides, user interface management systems (UIMS) are becoming available. These software systems are capable of generating source code from a non-

procedural representation of a user interface [FISH92]. Even with the introduction of this new technology, a consistent application user interface among application programs is lacking in many GUI environments. Pictured below is an example of an application user interface for a database application program on a Macintosh.

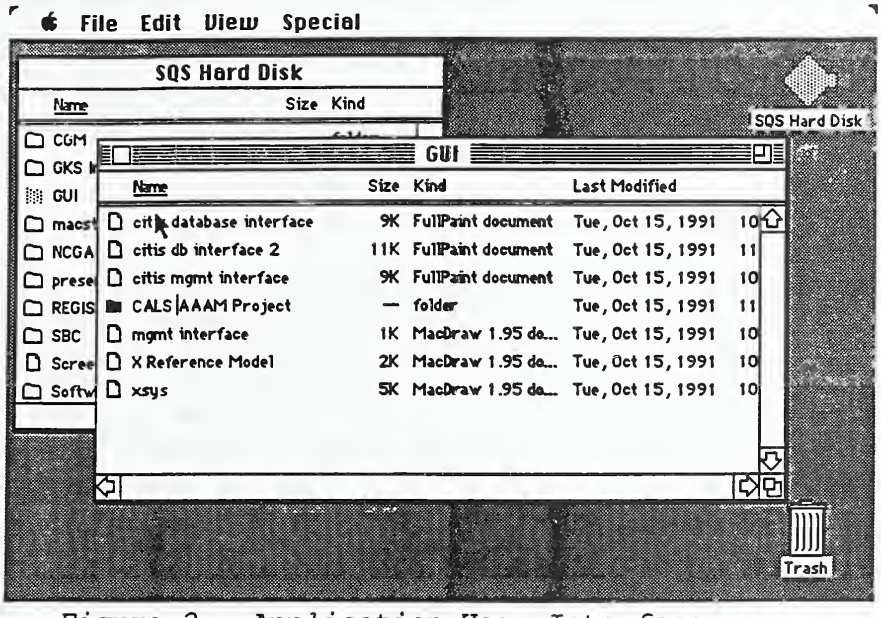

Figure 2. Application User Interface

3. **Application Programmer's Interface (API)** is a library of procedures and data objects for interacting with the GUI. The API is used to program screens and process the dialogue between a user and an application, i.e., how user entries are communicated to an application and how application responses are communicated back to a user [FISH91]. The API is used to implement the display manager program and the application programs. In this paper, for simplicity, an API is characterized as consisting of at least two parts: low-level procedures which draw simple geometric shapes, and toolkits which consist of procedures to draw composite objects, such as windows with scroll bars. Such toolkits invoke the low-level procedures and encourage the production of applications with a consistent user interface.

The three components described above may characterize a GUI either by themselves or in some combination, depending upon the context. Many GUIs support all three elements. The Display Manager Program and Application User Interface work together to provide end users with the ability to interact with their systems and application programs in a graphical and consistent manner. In addition, the API provides application programmers with tools needed to develop user-friendly interfaces.

The result of combining the three elements of a GUI is a complete set of tools that can be used to implement or use any interface required.

B. Why Use GUIs?

GUIs encourage users to explore and exploit applications without having to read long manuals and memorize commands. They offer easy access to functions on a computer. Because people tend to respond better to the visual than to the written, a GUI is often preferred to the character-mode interface. It allows a user to input commands into a system through the use of menus or by pointing to icons that picture commands rather than typing command sequences on the keyboard. Because GUIs typically offer multiple windows, a user is able to see and process more than one activity at a time in a multi-tasking environment. GUIs make computing more accessible to more users and allow users to perform a broader range of complex tasks more easily.

A GUI allows for a consistent way of moving across a range of applications using visual clues that are self-explanatory; this is often referred to as "direct manipulation," (e.g. moving a picture of a document to a picture of a trash can to delete the document). It also allows for the movement of data between applications easily. A GUI serves as the common glue between applications in a computing environment. This allows easy learning as users are able to predict the behavior of a new application program based on their experience with other applications running under the same GUI. Not only does a GUI make a user feel more at home, it involves him more by improving response time due to the quick switching between application programs.

GUIs impact day to day activity by increasing productivity and providing consistency between different applications and perhaps even disparate computing environments. They are usually easy to

learn, fun to use, and give users more confidence in the use of their computing environments.

C. PROBLEMS WITH GUIs.

As with any new technology, GUIs are not without their problems. The main problem is a standard GUI is not available with all of the elements mentioned above. This results in vendors developing unique and incompatible GUIs, sometimes depending upon the platform for which they are built.

Systems today have multiple platforms from different vendors, each running its own flavor of GUI. Because platforms have different capabilities, the interface for a particular application program may differ between platforms. The result is that applications do not appear or behave in the same manner on disparate platforms. Another possible problem is incompatibility between applications due to varying interfaces for different application programs on the same platform, ultimately causing confusion for users. The choice of hardware, and even application programs, determines the flavor of a GUI [FISH91]. The end result may be the inability to share data and resources among diverse applications and computing environments in an organization or enterprise.

Another problem results from the use of different technologies to build GUIs. Some GUIs use the client/server technology (e.g., Motif, OpenLook) while others use single-user access technology (eg., MicroSoft Windows, MacIntosh). In single-user access mode, a user is only able to access applications that are resident on his system and is not able to share resources. In a client/server environment users are able to share resources and interact with several application programs running on different platforms. For example, in the diagram below, workstation A has two windows on its display. One window is controlled by client application A1 running on Host H1. The other window is controlled by client application A2 running on Host H2. Similarly, workstation B displays a window controlled by client application A1 on host H2 and another window controlled by client application A2 running on host H1. For both workstations A and B input to the appropriate application can be controlled by the location of the mouse cursor on the display.

The workstation is the server and the application programs residing on the hosts are clients. Since the applications controlling the interactive devices of the workstations are running on remote

hosts, the hosts are remote execution servers for the client workstations. Conceptually, what constitutes a client and what constitutes a server depends on the nature of the service relationship and not on whether a node is normally thought of as a workstation or a host [BARK89].

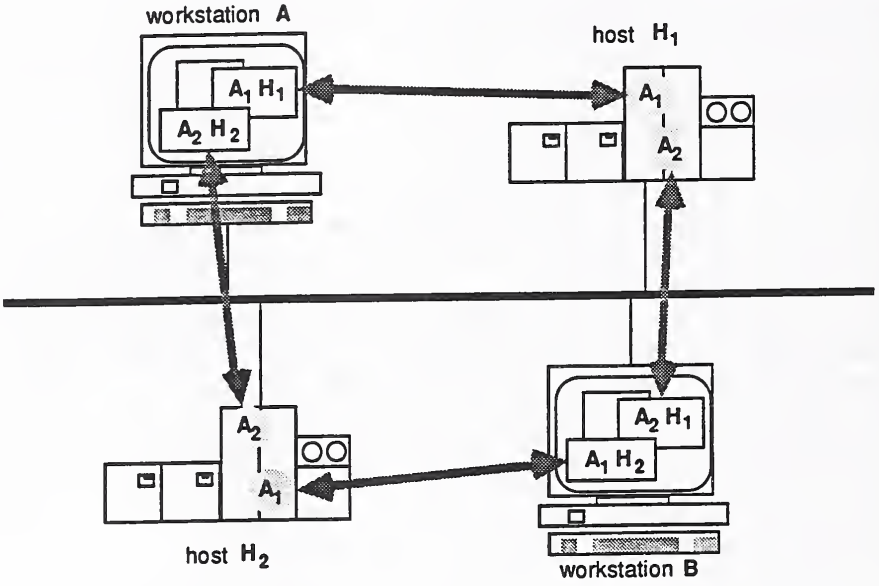

Figure 3. Client/Server Model [BARK89]

There are three possible general solutions to the problems associated with GUIs. The first is to mandate the use of one particular platform running one particular display manager and application user interface using one particular application programmer's interface throughout an organization or project enterprise. However, this approach would be costly as new hardware and software would be required. The second solution is to allow users to purchase any GUI that suits their needs, which would probably result in a diverse, possibly incompatible enterprise. The third solution is to create or specify a GUI that is able to run on an assortment of disparate machines and provide the functionality of the three components of a GUI. This will create uniformity and consistency among application programs and disparate computing environments.

D. <u>The Usefulness of GUIs with CITIS.</u>

There are many suppliers and contractors in the CALS environment,

and each supplier will be required to set up a CITIS service for government users. Each supplier has a different interface into his system, and even a single supplier may have different interfaces into the many applications that he is developing for CALS. This causes a problem in the area of remembering keystrokes and function keys to process information. Training is expensive and there is much to remember. A standardized GUI would eliminate the need to memorize many different commands required for accessing information from the different suppliers and applications. This would mean using a GUI that has a standardized display manager program, application user interfaces, and API interfaces for existing platforms.

The use of a GUI would create a graphical user-friendly atmosphere in the CITIS environment and allow users to access authorized information in a consistent manner. It would not be necessary to memorize commands or functions, or to learn a new interface each time the CITIS service is used. A GUI would enable users to remotely execute programs in a windowing environment.

The GUI should be operating system independent so as to avoid the cost of purchasing new equipment by running on existing platforms. The GUI will be required to provide well-integrated interaction within a window as well as between windows.

The GUI will need to be adaptable in order to accommodate a wide range of users - from the novice user to the well-experienced user. This will allow a user to customize the GUI to his preference while still providing compatibility between applications and platforms.

E. State of GUI Technology.

There are numerous GUIs available, each with a different "look and feel." The platform may determine the windowing environment that is used. Most workstations and PCs today offer a windowing environment which contains at least two of the three characteristics of a GUI (presented in Section III.A.). The X Window System (also known as X) is the most widely supported GUI architecture available today. X was developed at the Massachusetts Institute of Technology (MIT) as a network-based, bit-mapped graphic user interface system.

X is a powerful network-wide, windowing system that delivers device-independent, operating system-independent, and network-

independent operations, uniting disparate systems and allowing resources to be shared among any computer platform. X is commonly associated with Unix, but is not operating system specific [TREA88]. It taps the power of a stand-alone workstation and the resources of corporate enterprise, regardless of the location, operating system, and processing capabilities.

The X Window System assumes a client/server model of distributed computing, and user interface applications based upon bit-mapped graphic displays. There are seven layers to a GUI, according to the NIST User Interface Model [FIPS158]. These are mapped to the layers of the X Window System in the diagram below.

MODEL LAYER	X WINDOW SYSTEM COMPONENT
6: Application	X Application Program
5: Dialogue	User Interface Management System
4: Presentation	User Interface Definition Language
3: Toolkit	Toolkit
2: Subroutine Foundation	Xt Intrinsics
1: Data Stream Interface	Xlib
0: Data Stream Encoding	X Protocol

Figure 4. NIST User Interface Reference Model [FIPS158]

Layer 0: Data stream encoding, or the X Protocol. It defines the format and sequencing of byte streams passed between the client and server.

Layer 1: Data Stream Interface, or the Xlib. It specifies a function call interface to build the messages defined in Layer 0.

Layer 2: Subroutine Foundation, or Xt Intrinsics. It uses features of the Data Stream Interface to provide the means to build components of window interfaces such as scroll bars; it can be thought of as a toolkit with which to build toolkits.

Layer 3 Toolkit layer which contains components such as menus, pushbuttons, scroll bars, etc. These components are used

to build application interfaces. Toolkits vary with vendors, but typically contain most of the common user interface elements.

Layer 4 Presentation layer, which determines the appearance of the user interfaces such as size, style, and color. It specifies how components in the Toolkit are composed to create windows and is usually accompanied by a window manager, which controls the size and location of windows.

Layer 5 Dialogue layer which coordinates the interaction between the application program and the user.

Layer 6 Application layer or X application which implements the functions required by the user [FIPS158].

Layers 0, 1 and 2 are standardized in Federal Information Processing Standard Publication (FIPS PUB) 158, and have the support of most major computer vendors.

X furnishes the means to produce graphical output by an X application. The window manager is the display manager and controls all X application windows. Vendors can develop applications that incorporate application interfaces without being concerned about the underlying display hardware on which the application runs.

X masks command-line functions of high-performance, multi-user, multi-tasking systems. The user can display several applications within windows running simultaneously on the display, although the applications may be on different machines with different operating systems.

Most GUIs today support X, meaning that X applications may run in a proprietary GUI. Within the Unix environment there are two commonly used GUIs. These are Open Look, which is supported by the Unix International consortium, and Motif, which is supported by the Open Software Foundation. Open Look supports the API interface as well as the management interface. It is supported by major computer vendors such as Sun and AT&T. Motif supports the API, management and application interfaces. It is supported by major computer vendors such as Digital Equipment Corporation, Hewlett-Packard and IBM. Both Open Look and Motif are built on the X Window System, but they have different features and different APIs.

X servers are available for the PC environment, and can be run with or without a proprietary GUI. For example, X servers are available for Microsoft Windows as well as other PC windowing environments.

The result of what is available is a variety of windowing environments from numerous vendors for virtually any platform that is compatible with X.

IV. NATIONAL/INTERNATIONAL STANDARDS VS. EXISTING DEFACTO STANDARDS.

There is no formal standard available for every layer of a graphical user interface at the national or international level. This is because there are so many GUIs available from different vendors, and no two are consistent at every interface layer. There is a Federal Information Processing Standards Publication (FIPS PUB) 158, which standardizes the lowest 3 layers of the X Window System, Version 11, Release 3. The X Window System is also included in the Application Portability Profile, a suite of open systems specifications needed to support the requirements of a specific domain of applications for an open systems environment developed for the U.S. Government.

While FIPS 158 does not standardize all elements of a GUI, it is intended to lay a foundation for standards that will assist in developing and acquiring network-based, bit-mapped graphic user interface applications.

It is conventional to develop applications using toolkit interfaces that are at a higher level than the interfaces covered by FIPS 158. The interfaces specified in FIPS 158 are intended to provide a foundation on which other layers will be added as consensus emerges within industry. The interfaces specified in FIPS 158 represent the consensus of the industry for lower-level X Window System interfaces [FIPS158]. A national standards group, Institute of Electrical and Electronics Engineers (IEEE) Portable Operating System Interface (POSIX) 1201.1, is currently developing a specification for a toolkit interface. The base document for this work is XVT. A second national standards group, IEEE POSIX P1201.2, is developing a recommended practice (i.e., Application User Interface) for those elements of user interfaces that must be consistent to allow users to transfer easily from one "look and feel" to another.

V. RECOMMENDATIONS.

The following table summarizes NIST recommendations for the use of a graphical user interface by CITIS:

GUI Characteristic		Recommendation	Benefits
Application Programmer's Interface (API)	Low Level	FIPS 158 (X Toolkit Intrinsics, Xlib, X Protocol)	Provides application portability in a network based client/server environment among almost all GUI platforms.
	Toolkit	Choose or develop one that supports the CITIS Application User Interface, or consider a user interface management system (UIMS).	
Application User Interface		Choose or develop "style guide" that defines the "look and feel" of the application. In the longer term, identify a suitable user interface management system (UIMS).	Provides consistent user interface to all CITIS applications.
Display Manager Program		Further study to determine if a display manager program is required.	It is not clear that investment in a CITIS display manager is warranted.

Table 1. NIST Recommendations

1. X is the only viable heterogenous, networking, computing environment offering graphical capability that allows resources to be shared among computer platforms. Thus, NIST advises CALS to use FIPS 158, the X Window System, as the API to be used in the development of CITIS applications. Using X in the CITIS environment will allow CITIS applications and resources to be shared among heterogeneous computing environments by enabling government users to access and transparently display remote applications running on any platform at contractor sites. While the response time for an X application running over a Wide Area Network (WAN) may currently be unacceptable in some cases, plans are underway to increase the capacity of WANs to gigabyte range. These WANs would be available to CITIS.

2. CALS should use one application user interface that will

define the appearance and interaction for all CITIS applications. The emerging specification from the IEEE POSIX working group (P1201.2) should be used as a basis for defining the "look and feel" of a CITIS application. A suitable, existing toolkit such as XVT [NIST91] to support the CITIS style guide should be chosen. Alternatively, a User Information Management System which can support the style selected for CITIS application without sacrificing source code portability could be identified. Finally, a toolkit could be developed that will support the CITIS style guide that is developed. Any of these suggestions will provide a consistent graphical interface into contractor services and allow easy, electronic access to the CITIS databases.

3. By using X as the basis for the CITIS environment, a CITIS display manager program can be developed, an existing display manager program supporting X may be chosen, or there may be no display manager program. That choice should be decided by the users.

The above recommendations allow for the integration of government and contractor resources. It will also provide consistency of CITIS applications among diverse systems and an open systems environment where the sharing of resources and data can take place freely in a graphical environment.

VI. CONCLUSION.

A GUI must provide consistent style and functionality throughout application domains within an organization or enterprise environment. A GUI must be able to provide needed functionality in various hardware environments [FISH91].

X is the only viable network-transparent, device-independent vendor-neutral windowing environment that allows resources to be shared among any computer platform. The use of the X Window System as the basic CITIS environment would eliminate the requirement for new equipment, retain a competitive market, and allow for integration of government and contractor resources.

The functional requirements of FIPS 158 are supported by most major computer vendors, and most GUIs implemented by vendors today support X applications.

VII. REFERENCES.

[BARK89] Barkley, John and Olsen, Karen, "Introduction to Heterogeneous Computing Environments," NIST Special Publication 500-176, November 1989.

[FIPS158] Federal Information Processing Standards Publication 158, The User Interface Component of the Applications Portability Profile, 29 May 1990.

[FISH91] Fisher, Gary, Draft "Open System Environment Graphical User Interface (GUI) Transition Strategies", August 1991.

[MIL-STD-CITIS] Draft Military Standard on Contractor Integrated Technical Information Services (CITIS), 18 May 1992.

[NIST91] NIST Special Publication 500-187 APPLICATION PORTABILITY PROFILE (APP) The U.S. Government's Open System Environment Profile OSE/1 Version 1.0, April 1991.

[TREA88] Treadway, Richard, "The View from DEC: 'The X Window System is an Exceptional Standard," Digital News Technical Topics, July 1988.

VIII. BIBLIOGRAPHY.

[AZZA91] Azzara, Mike, "How The X Puzzle Fits Together," Unix Today!, March 1991.

[BROW90] Brown, Dr. C. Marlin, "Be Choosy About GUI," Software Magazine, 10 November 1990.

[DALY91] Daly, James, "Graphical Face on Supercomputing," Computer World, 27 May 1991.

[DANC91] Dance, Richard A., "Windows and Other GUIs," Federal Computer Week, 4 March 1991, p. 41-42.

[GETT86] Gettys, Jim and Schiefler, Robert W., "The X Window System," ACM Transactions on Graphics, Vol. 5, No. 2, April 1986, p. 79-109.

[HEILER] Heiler, Sandra, Siegel, Michael and Zdonik, Stanley, "Heterogeneous Information Systems: Understanding Integration"

[HOLM91] Holmes, Edith, "Federal Use of Desktop Products Outstrips Industry Expectations," Federal Computer Week, 12 August 1991.

[LEVI90] Levine, Harvey A., "Graphical Interfaces Give Power to Users," Software Magazine, November 1990, p. 87-93.

[LINE90] Linell, Dennis, "Graphical Interfaces Give Users Tools to Explore," Government Computer News, 3 September 1990.

[MARC91] Marcus, Aaron and van Dam, Andries, "User-Interface Developments for the Nineties," Computer, September 1991, p. 49-57.

[MARS90] Marson, Carolyn Duffy, "GUIs: Many Federal Users Are Choosing GUIs Over Command-driven Systems," Federal Computer Week, 5 March 1990, p. 14-27

[PADO91] Padovano, Michael, "Motif and Open Look: Two Views on Managing Windows," Systems Integration, June 1991.

[SEYM91] Seymour, Jim, "Graphical User Interfaces Give PC Users a Winning Hand," Today's Office, June 1991 p. 17-20.

[SYST91] "Vendors Use NCGA To Roll Out New Gear," Systems Integration, June 1991, p. 13.

[VIOL91] Violino, Robert, "Are GUIs Worth the Hassle?," Information Week, 11 March 1991, P. 30-36.

[WAGN91] Wagner, Mitch, "POSIX Pick Could Slow GUI Wars," Unix Today, 5 August 1991.

[WAGN91] Wagner, Mitch, "Motif Vs. Open Look: Window Managers Still at Loggerheads," Unix Today!, March 1991.

[WAGN91] Wagner, Mitch, "Sorting Out X Server Options," Unix Today!, March 1991.

[WOHL90] Wohl, Amy D., "Graphical User Interfaces Make Computing Child's Play," Today's Office, May 1990, p. 31-34.

NIST-114A
(REV. 3-90)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY

BIBLIOGRAPHIC DATA SHEET

1. PUBLICATION OR REPORT NUMBER
NISTIR 4876

2. PERFORMING ORGANIZATION REPORT NUMBER

3. PUBLICATION DATE
JULY 1992

4. TITLE AND SUBTITLE

An Introduction to Graphical User Interfaces and Their Use by CITIS

5. AUTHOR(S)

Susan Q. Sherrick

6. PERFORMING ORGANIZATION (IF JOINT OR OTHER THAN NIST, SEE INSTRUCTIONS)

U.S. DEPARTMENT OF COMMERCE
NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY
GAITHERSBURG, MD 20899

7. CONTRACT/GRANT NUMBER

8. TYPE OF REPORT AND PERIOD COVERED

9. SPONSORING ORGANIZATION NAME AND COMPLETE ADDRESS (STREET, CITY, STATE, ZIP)

10. SUPPLEMENTARY NOTES

11. ABSTRACT (A 200-WORD OR LESS FACTUAL SUMMARY OF MOST SIGNIFICANT INFORMATION. IF DOCUMENT INCLUDES A SIGNIFICANT BIBLIOGRAPHY OR LITERATURE SURVEY, MENTION IT HERE.)

A Graphical User Interface (GUI) is a powerful tool that is used for simplifying a computing environment. This paper provides a tutorial on the various meanings of the term GUI, describes the usefulness of GUIs, identifies problems with GUIs, and recommends that the X Window System GUI be used within the CITIS specification.

The term "GUI" can have various meanings in different contexts. In order to provide a framework for discussion in this paper, a GUI is characterized as having at least one of the following components:

1. Display Manager Program,
2. Application User Interface,
3. Application Programmer's Interface (API).

One of the major problems with GUIs is the potential lack of portability of an application developed to run using a particular GUI. The X Window System provides a means of overcoming this problem because applications developed using the X Window API may be run using almost all other GUIs. Consequently, the X Window System can be used by CITIS to provide easy to use applications which may be used on almost all GUI platforms.

12. KEY WORDS (6 TO 12 ENTRIES; ALPHABETICAL ORDER; CAPITALIZE ONLY PROPER NAMES; AND SEPARATE KEY WORDS BY SEMICOLONS)

Graphical User Interface (GUI); X Window; Computer-aided Acquisition and Logistic Support (CALS).

13. AVAILABILITY

XX	UNLIMITED
	FOR OFFICIAL DISTRIBUTION. DO NOT RELEASE TO NATIONAL TECHNICAL INFORMATION SERVICE (NTIS).
	ORDER FROM SUPERINTENDENT OF DOCUMENTS, U.S. GOVERNMENT PRINTING OFFICE, WASHINGTON, DC 20402.
XX	ORDER FROM NATIONAL TECHNICAL INFORMATION SERVICE (NTIS), SPRINGFIELD, VA 22161.

14. NUMBER OF PRINTED PAGES
24

15. PRICE
A02

ELECTRONIC FORM