The Library for System Solutions End User Interface Reference

Document Number GG24-4107-00

July 1994

International Technical Support Organization
Boca Raton Center

Take Note!

Before using this information and the product it supports, be sure to read the general information under “Special Notices” on page xi.

First Edition (July 1994)

This edition applies to IBM and non-IBM products for End User Interface development.

Order publications through your IBM representative or the IBM branch office serving your locality. Publications are not stocked at the address given below.

An ITSO Technical Bulletin Evaluation Form for reader's feedback appears facing Chapter 1. If the form has been removed, comments may be addressed to:

IBM Corporation, International Technical Support Organization
Dept. 91J Building 235-2 Internal Zip 4423
901 NW 51st Street
Boca Raton, Florida 33431-1328

When you send information to IBM, you grant IBM a non-exclusive right to use or distribute the information in any way it believes appropriate without incurring any obligation to you.

Abstract

This document is part of the Library for Systems Solutions, which is intended for professionals involved in defining solutions in the heterogeneous computing environments. The library consists of three types of documents:

- Computing Technology
- Function Reference
- Technology Reference

This document is the Function Reference book regarding End User Interface.

The book consists of two parts. Part one gives an overview to several aspects of End User Interfaces, Design principles, and explains the key components of Graphical User Interface. Part two of the book briefly describes programs and tools available to develop or utilize Graphical End User Interfaces on multiple software platforms. Some knowledge of operating a Personal Computer is assumed.

(71 pages)

Contents

Figures

Tables

Special Notices

This publication is intended to help IS managers, user group mangers, project leaders, consultants and other professionals involved in product selection for development of enterprise wide user interfaces. The information in this publication is not intended as the specification of any programming interfaces that are provided by products described in this publication. See the PUBLICATIONS section of the IBM Programming Announcement for the products that are mentioned in in this publication for more information about what publications are considered to be product documentation.

References in this publication to IBM products, programs or services do not imply that IBM intends to make these available in all countries in which IBM operates. Any reference to an IBM product, program, or service is not intended to state or imply that only IBM′s product, program, or service may be used. Any functionally equivalent program that does not infringe any of IBM′s intellectual property rights may be used instead of the IBM product, program or service.

Information in this book was developed in conjunction with use of the equipment specified, and is limited in application to those specific hardware and software products and levels.

IBM may have patents or pending patent applications covering subject matter in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to the IBM Director of Licensing, IBM Corporation, 208 Harbor Drive, Stamford, CT 06904 USA.

The information contained in this document has not been submitted to any formal IBM test and is distributed AS IS. The information about non-IBM (VENDOR) products in this manual has been supplied by the vendor and IBM assumes no responsibility for its accuracy or completeness. The use of this information or the implementation of any of these techniques is a customer responsibility and depends on the customer′s ability to evaluate and integrate them into the customer′s operational environment. While each item may have been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environments do so at their own risk.

The following terms, which are denoted by an asterisk (*) in this publication, are trademarks of the International Business Machines Corporation in the United States and/or other countries:

AD/Cycle	AIX
AIX/ESA	AIXwindows
APL2	AS/400
CICS	CICS OS/2
COBOL/2	COBOL/400
Common User Access	CUA
DATABASE 2	DB2/2
DB2/6000	GDDM
IBM	IMS/ESA
IMS/VS	MVS/ESA
MVS/SP	MVS/XA
Operating System/400	OS/2
OS/400	POWERserver

POWERstation
RISC System/6000
SAA
SQL/400
System/390
VisualAge
VM/XA
Xstation Manager

PS/2
RPG/400
SQL/DS
System/370
Systems Application Architecture
VM/ESA
Workplace Shell

The following terms, which are denoted by a double asterisk (**) in this publication, are trademarks of other companies:

Apple Macintosh	Apple Computer,Inc.
Micro Focus Cobol	Micro Focus Limited
Microsoft, Windows	Microsoft Corporation
Microsoft SQL Server	Microsoft Corporation
Motif, OSF/Motif	Open Software Foundation, Inc.
Oracle	Oracle Corporation
Smalltalk/V	Digitalk Inc.
PostScript	Adobe Systems, Inc.
Wabi	Sun Microsystems, Inc.
X Window System	Massachuchetts Insitute of Technology
UNIX	Novell, Inc.
Flashpoint, KnowledgeWare	Mozart, Mozart Systems Corp.
Enfin, Easel Corp.	Ellipse, Cooperative Solutions

Preface

This document is part of The Library for Systems Solutions, which is a set of books that describes and explains information processing requirements and solutions. The library provides solutions for multiple configuration environments ranging from totally host to multilevel distributed environment.

This document is a reference book on End User Interface (EUI) development. User interfaces are now most important aspect of information processing systems. Technological development over the time, has now made it possible to isolate the EUI part of software development from program logic. The architectures and specifications like IBM SAA Common User Access and OSF/Motif are evolving in a way to provide portable EUI across different platforms.

This document defines EUI, discusses principles / guidelines for EUI development, defines major components of workstation based Graphical User Interface (GUI) and finally provide reference information about selected tools and products which IS management can use for developing applications that can provide efficient workstation based user interfaces.

It is not the intention of this document to describe all possibilities and all products that enable development of EUI. The document discusses different aspects of EUI at overview level. References to books and other documents have been provided through out this book for the benefit of those who need more in-depth knowledge on this subject.

The book is intended for IS managers, user group mangers, software development project leaders, system analysts, consultants and other professionals involved in selecting products for EUI development.

How This Document is Organized

The document is organized as follows:

- Chapter 1, “Overview”

  This chapter explains what End User Interface (EUI) is, defines it and provides brief discussion on evolution of EUI from entry level to GUI based Workplace model of EUI. Discussion on different types of user interfaces has been included at overview level.

- Chapter 2, “Design Principles/Guidelines”

  This provides brief overview of EUI design principles and guidelines for product designers. Whenever required the real life examples are included. The design guidelines are mostly based on GUI Workplace model.

- Chapter 3, “Components”

  This chapter defines and provides brief information on key elements used in developing PWS Workplace based Graphic User Interface (GUI).

- Chapter 4, “Solutions”

  This chapter gives short overview of tools which developers can use in building EUI for their applications.

Related Publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this document.

- *IBM Enterprise Solutions for the 1990s*, G320-9929
- *AS/400 Technology Journal*, S325-6020
- *System Application Architecture*, GC26-4784
- *SAA CUA Advance Interface Design Reference*, SC34-4290
- *Object-Oriented Interface Design - IBM CUA GUidelines*, SC34-4399
- *Tkach/Putick, Object Technology in Application Development*, GG24-4290
- *Edmonds, The Separable User Interface*, ISBN 0-12-232150-2
- *Shneiderman, Designing the User Interface*, ISBN 0-201-16505-8
- *OSF/Motif Programmer's Reference Manual*, OSF-0-M-00889-001

International Technical Support Organization Publications

The Library for Systems Solutions includes:

- *Computing Technology Reference*, GG24-4100 *
- *Application Development Reference*, GG24-4101
- *Workload Management Reference*, GG24-4102
- *Data Reference*, GG24-4103
- *Directory, Naming, and Timing Reference*, GG24-4104 *
- *Printing and Viewing Reference*, GG24-4105
- *Security Reference*, GG24-4106
- *End User Interface Reference*, GG24-4107
- *Multimedia Reference*, GG24-4108 *
- *Image Processing Reference*, GG24-4109
- *Open Networking Reference*, GG24-4110
- *LAN Reference*, GG24-4111
- *Office Reference*, GG24-4112
- *System Management Reference*, GG24-4113
- *System Management Reference for Managed System/390*, GG24-4114
- *System Management Reference for Managed RISC/6000 Systems*, GG24-4115
- *System Management Reference for Managed Personal Systems*, GG24-4116
- *System Management Reference for Managed AS/400 Systems*, GG24-4117

Note: Publications marked with asterisk (*) will be published in the future.

A complete list of International Technical Support Organization publications, with a brief description of each, may be found in:

Bibliography of International Technical Support Organization Technical Bulletins, GG24-3070.

To get listings of redbooks online, VNET users may type:

```
TOOLS SENDTO WTSCPOK TOOLS REDBOOKS GET REDBOOKS CATALOG
```

How to Order Redbooks

IBM employees may order redbooks and CD-ROMs using PUBORDER. Customers in the USA may order by calling 1-800-879-2755 or by faxing 1-800-284-4721. Visa and Master Cards are accepted. Outside the USA, customers should contact their IBM branch office.

You may order individual books, CD-ROM collections, or customized sets, called GBOFs, which relate to specific functions of interest to you.

Acknowledgments

The advisor for this project was:

Alexander Gregor
International Technical Support Organization, Boca Raton Center

The authors of this document are:

Sheikh Nisar Ahmed
IBM Pakistan

Ruediger W. Seidl
IBM Germany

This publication is the result of a residency conducted at the International Technical Support Organization, Boca Raton Center.

Thanks to the following people for the advice and guidance provided in the production of this document:

Ian Crane
IBM ITSO Poughkeepsie

Ray N. Shasteen
IBM ITSO Boca Raton

Doris Devensky
IBM ITSO Boca Raton

Chapter 1. Overview

This chapter explains what End User Interface (EUI) is, defines it and provides a brief discussion on the evolution of EUI from entry level to GUI-based Workplace models. Discussions on different types of user interfaces have been included at an overview level.

1.1 What Is User Interface

The End User Interface (EUI) to a computer has always been a subject of prime interest and concern for user management. Even in the early days of batch processing, when users did not directly interface with computers, the designing of output reports and input forms was an important user consideration. Also for programmers′ productivity, a good user-friendly editor was important. Many papers and books on this subject were written. Still the words “User Interface” remained ambiguously defined. In this section we will first establish a widely accepted definition of EUI and in the next section we will try to help you understand this subject in more detail by looking at the gradual evolution of the EUI.

In English the normal use of the word “Interface” names the well-defined relationship between two entities. In computer terms, “Interface” defines the relationship which allows different objects to be connected together. Considering this explanation of interface, we can conclude that a User Interface is a set of techniques and mechanisms consisting of hardware and/or software that a person uses to interact with computer-based systems. It can also be defined as the means by which a user communicates with a computer and vice versa.

It would be timely to mention here that a user (or end user) is not just the user of the application but a person involved in application development (i.e., programmers, analysts etc.). System operations and system programming are also users of the system and the user interface is also equally important to them.

Thus, a user interface in the case of a computer can include a keyboard, a pointing device, items appearing on display screens, the software which supports these devices and techniques (commands, menu options, windows, icons, etc.) and other devices used to provide human interaction with the computer.

The end user interfaces can be divided into three basic categories:

- Command line user interfaces, in which a user remembers commands and types them.
- Menu-Driven user interfaces, in which a user is provided with a hierarchically organized set of choices.
- Graphical User Interfaces (GUIs) in which a user points to and interacts with visible elements of the interface by using a pointing device or a keyboard. This interface was further extended to enhanced models such as the Workplace Model. These models are explained further in this topic.

Depending on the type of display device, the customer selects those models that suit the interface which meet requirements. Table 1 summarizes the availability of EUI in relation to display devices:

Table 1. Display Device and EUI Matrix

EUI Categories Device	Command Line EUI	Menu Driven EUI	GUI Basic	GUI Workplace
NPT (Nongraphic)	√	√		
NPT (Graphic)	√	√	√ *	
PWS	√	√	√	√

Note: * with limited functions √ - denotes availability

The User Interface (at the program level) is that part of the program that presents, displays and accepts input from the person using that program. The rest of the program is called “application semantic,” or briefly, “application.”

A comprehensive User Interface Management System (UIMS) helps developers create and manage all aspects of user interfaces in such a way that they do not have to code many details in the application. Considering the importance of this separability, much debate has taken place about appropriate architectures for user interfaces. Figure 1 represents one of the earliest architectures named after the workshop in which it was developed in 1985. Although basic, it provided a good platform for future development in this area.

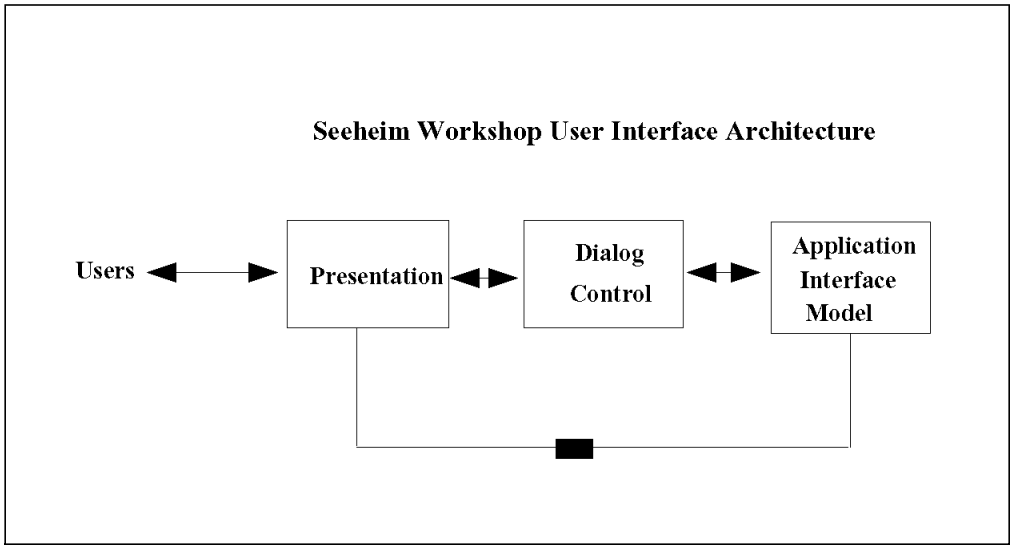


Figure 1. Seeheim User Interface Model

This and other initial EUI models are described in detail in the book, “The Separable User Interfaces” by E. A. Edmonds.

1.2 Evolution of User Interface

1.2.1 Introduction

As mentioned earlier, probably the earliest user interface with a computer was through input forms and output reports. A predesigned punched card used as a fine-ticket by traffic police or by order entry operators to directly punch information written on them is an example of such an interface. This was in the ′60s when almost all computing was done in batch mode. There were no tools available and programmers implemented user interfaces for accepting input from users and gave output to them by incorporating full code in applications.

The facilities such as COBOL Report Writer provided some relief. These made report designing simple and an automated part of report coding. This also separated report designing from application logic.

The need to have comprehensive User Interface Management Systems (UIMS) increased when the user′s requirements moved from batch processing to online processing in the ′70s. The applications were still central host-based with Non-Programmable Terminals (NPTs) attached. Nonavailability of User Interfacing tools made programming too complex with too much focus on technical development and too little on business needs. An average programmer spent more time in developing user interfaces, i.e., menus, screens, printer forms, etc., than on application semantics.

So there was an acute need to simplify or automate the work of developing a user interface to the application. In the subsequent development of map formatters, report formatters, dialog managers, etc., efforts were made to meet this need.

1.2.1.1 Screen Map Formatters

The announcement by IBM* of Basic Mapping Support (BMS) for creating maps (i.e., screens and printer-form layouts) and mapsets for online application development was a major breakthrough. Though appearing difficult to use when compared to the many easy-to-use EUI tools of today, BMS was a great productivity improvement tool of the ′70s. It eliminated tedious coding for screen designing and made a large part of BMS coding reusable. BMS was made available to run IBM mainframe (S/370) operating systems VSE and MVS* supported by its online transaction manager, Customer Information Control System (CICS*). CICS has now become the industry defacto standard as an online transaction manager. It is available on all IBM operating systems and also several non-IBM systems. All CICS environments in some way or other support BMS.

Many high level screen designing tools developed later for IBM mainframe environments (S/370) were based on BMS specifications. In fact, they first generated BMS macros to be converted into assembly code by translators. Screen Designing Facility (SDF) was one of the first. SDF was first announced for VSE, the operating system for IBM mid-range S/370 systems. It is an online development tool for CICS/DOS/VS (VSE version of CICS) application programmers who want to define or edit maps, mapsets, and partition sets for displays and printers. SDF eliminates BMS coding. The online operations and easily used oriented functions of the tool enhanced productivity in application development. SDF was later announced for VM and MVS as SDF-II. This product included functions of SDF and covered other areas of the EUI such as panel definition, and operator control tables. SDF-II provided migration facilities across different operating environments for EUI elements (screens, menus, panels, etc.) on IBM mainframe systems as well as OS/2*. Screen Design Aid

(SDA), an integral part of OS/400*, the operating system for IBM AS/400* system, (a follow-up product of IBM S/36 and S/38) provided a very easy-to-use interface for developing menus and screens.

SDF, SDA and many other screen formatters relieved programmers from tedious user interface coding and helped them to concentrate on the application.

Impacts:

- Screen formatters improved development productivity significantly.
- Limited portability of EUI application elements was possible. SDF-II enabled users to port their screens, menus and panels to almost all IBM environments.
- EUI development was separated from application semantic.
- EUI code reusability increased.

1.2.1.2 Dialog Management

With the rapid increase of online base applications, more capabilities were required than just designing screens, menus and reports. One such requirement is Dialog Management. A dialog is defined as, “The interchange of information between a computer and its user through a sequence of action requests by the user and presentation of responses by the computer.” Panels facilitate this information exchange. The dialog with the user begins when the application displays a panel and asks for user interaction. These panels and their management are provided by the dialog manager. IBM′s Interactive System Productivity Facility (ISPF) was one of the first dialog managers. ISPF provided control and services to support processing of interactive applications in different IBM host environments. ISPF and its associated product ISPF/PDF together are designed to increase productivity in development of online/interactive applications and ease-of-use for the operators. The dialog management facilities were later made available on other platforms in different names and now have become a key element of information processing.

The personal computers and workstations have had significant impact on dialog management development because of better graphic capabilities and built-in intelligence. In the PC environment a dialog is redefined as, “Interaction between a computer and a person through a single primary window, its associated pop-up windows and associated help windows.” All pop-up windows are organized in a single chain of the primary widow. Almost all PC and workstation-based systems offer dialog management facilities to a varying degree, either as a built-in function or as an optional product.

Impacts:

- Dialog managers now offer interfaces that do most of the dialog-related work, thus relieving the application of the need to function. This simplifies development work.
- Makes navigation across different applications possible without any additional code being written.
- Most of the available dialog managers offer Application Programming Interfaces (APIs) which provide greater flexibility to developers.

1.2.2 User Interface with Operating Systems

The user interface with operating systems and subsystems was normally provided by system operator commands or special commands interpreted by these subsystems such as Job Control Language (JCL) commands. In the 1970s IBM′s Interactive Productivity Facility (IPF) provided some relief to the system users who found a command-based interface difficult to learn and work with. IPF reduced the requirement for learning JCL to some extent and in an interactive manner created JCL-based options selected by users. IPF was made available in both VSE and VM, the most popular mid-range mainframe (S/370) operating systems. Later on, the Interactive Interface was provided as a built-in function of VSE/SP, the new version of VSE announced in the mid 1980s, which completely eliminated the requirement for learning commands and all system programming and development tasks could be performed through menu options. This facility was further enhanced in the late ′80s and ′90s.

In MVS, the operating system for the IBM large mainframe environment and the user interface remained command and JCL-based. This was because customers in this environment wanted a very high degree of performance rather than ease-of-use. Their users were highly skilled, matured and well-trained. Support for a large number of concurrent users and tasks with very fast response time was their priority. However, availability of very high speed processors and the operating system′s ability to support large processors and virtual storage has now made it possible to implement high level, easy-to-use user interfaces even in MVS without compromising performance.

As in the case of mainframe environments, the user interface in smaller systems such as the S/3X was also command-driven in the initial stages. In the late ′80s the Interactive Interface was also implemented in OS/400, the operating system for AS/400 which was the follow-up product of the S/3X systems.

In the latest version of OS/400 (version 2) the system and user interfaces were further simplified and enhanced by including a new interface called Operational Assistant User Interface. This focuses on common system tasks done by users and system operators, such as: working with output, batch jobs, sending and receiving messages, working with device status, backup tasks, changing system options, cleaning the system, enrolling the users, etc. This interface makes users more productive, frees up technical support personnel, and minimizes training needs.

For more information on EUIs provided by AS/400, see IBM publication AS/400 Technology Journal (S325-6020).

Many of the latest versions of the operating systems now also provide GUIs for system management functions. IBM SystemView framework for automation of system management tasks provides specifications for development of GUI-based interfaces.

Impacts:

- System operation and system programming functions became simpler and more manageable.
- Reduced training and learning curve for system personnel.
- IS management dependency on highly skilled system personnel reduced.

1.2.3 User Interfaces In Fourth GL Environment

With significant advancement in fourth GL's and end-user computing in the '80s, query-based user interfaces became important. This was the time when use of Relational Database Management Systems (RDBMS) spread rapidly. The availability of Structured Query Language (SQL) to manipulate data on RDBMS created several opportunities for EUI developers to develop interfaces which were easy to learn and use. These interfaces were:

- Query By Example (QBE) based, where a mask of an entity (table) along with their attributes (columns) is drawn on the screen by tools and the user selects desired attributes for the desired operations and conditions. The tool then builds SQL commands based on the user's input.
- Fill in the Form type, where users are given a form on screen to fill in names of entities, attributes, selection criteria, joining conditions, output format options, etc. The tools then build commands or programs based on this information.
- Conversational Mode, where the user responds to questions asked by these products as if they are in conversation with the system.

IBM Query Management Facility (QMF) and Application System (AS) on the mainframe, Query/36 (followed by Query/400 and QM/400) and OS/2 Query Manager are examples where at least one of the interfaces mentioned above has been implemented. Almost all fourth GL tools provide similar interfaces.

Impacts:

- EUI in the fourth GL environment tremendously reduced the need for developing programs for meeting users' day-to-day requirements.
- Users started using these tools to meet their adhoc requirements.
- Users' dependency on IS was reduced.

1.2.4 Graphic User Interface (GUI)

This was also the time when customers demanded graphic user interface (GUI) with end user computing products. In IBM mainframe environments GUI was provided with the help of Graphic Data Display Manager (GDDM*) and the associated tools. GDDM was available on all three mainframe operating systems, i.e., VSE, VM, and MVS on S/370 platform and users were using it to meet their graphic needs. Many query and fourth GL products built their EUIs on GDDM.

The emergence of personal computers as programmable workstations (PWSs) with their graphical capabilities enhanced GUI requirements. The current GUI specifications provide multiple windows and iconic representations of system capabilities. Users use a pointing device (mouse) or keyboard keys to select an icon. The GUIs are closely related to Object Oriented User Interface (OOUI). Object orientation is now a mandatory requirement for good GUI-based products. Almost all leading Computer Aided Software Engineering (CASE) tools provide a user interface at upper levels, i.e., at analysis and design level based on GUI. With NPTs, users work most of the time in a single session screen mode. Now on PWS a user can open several windows on one screen accessing different applications in each window. This created a need to have a common user interface for all applications or tasks users would run on their PWSs. This means it is now more important that GUIs conform to industry standards.

The main elements from which a GUI is constructed are windows, icons, menus, and pointers. These elements work together to provide users with a consistent and easy-to-use interface to their tasks.

The presentation component produces the interface that the user views and interacts with. It is often, but not always, a graphical user interface. GUIs provide a graphic-oriented presentation front-end to applications, and provide (or simulate) multitasking processing. The major windowing environments are Windows from Microsoft, OS/2 Presentation Manager from IBM, Motif** from Open System Foundation, Openlook from UNIX** System Laboratories (USL) and DECwindows** for Digital′s Ultrix and VMS-based systems.

Macintosh′s** (Apple′s**) contribution in development of GUIs and windowing technology is significant. The ease-of-use initially provided by Apple systems has finally been imported to other environments. The Macintosh interface has been the basis for the new GUI style, such as NeXTStep from NeXT Inc. The NeXTStep′s object-oriented techniques allow a developer to work with visual representation of tasks and their corresponding code and add consistency to the appearance of application screens.

Impacts:

- GUI resulted in significantly improving users and developers productivity. A user could now open several windows on his/her PWS and run different applications in each one of them simultaneously.
- GUI provided consistency across applications and in some cases across different platforms.
- GUI made it possible to look at a screen similar to one′s desk. Now icons can be created to represent each document you are handling.
- GUI made workstation interface possible with new information types such as voice, sound, pictures, images, etc.

During the 1970s, the shift from batch to online processing and integration of telecommunications with computing, took on increased importance as the growth of use of NPTs and interconnection of computing systems via communication lines. In the late 1980s we saw mushrooming growth of personal, work-group, department computing. NPTs were being replaced with Intelligent Work Stations (IWS), or as we have named them in this book, PWSs. Personal and departmental computers were connected together, and also to central hosts. These interconnected systems belonged to different platforms, IBM as well as non-IBM. This created an urgent need to have common frameworks so that customers got consistency in dealing with these systems. This was more important for EUI, as it provided direct interaction with users of the systems. Several architectures were announced to address this problem.

Until the mid ′80s the primary users of computerized systems were professional programmers, operators, data entry clerks and well-trained users. The technological advancement had then made it affordable for individuals and departments to have their own computing facilities. However, they could not afford sophisticated data processing expertise. This meant that they needed an interface with the computer which could mask the complexity of the computer system from them and they could still could do the job. This also increased the necessity of a new design approach and a new interface design. This need demanded tools to provide ease-of-use. However, these tools were very

complex to develop so there was an urgent need for properly and accurately defined architectures.

1.2.5 IBM SAA Common User Access (CUA)

In 1987, IBM made one of the most important announcements of its history: System Application Architecture* (SAA*). SAA defines an architecture design framework that has the objective to provide the base for building consistent applications across IBM's major computing systems. To achieve its objectives of consistency, SAA defines three underlying architectures:

1. Common User Access* (CUA*)
2. Common Communication Support (CCS)
3. Common Programming Interface (CPI)

The last two are beyond the scope of this book. In this section CUA provides detailed specifications for user interface for SAA. Figure 2 briefly explains CUA's position in SAA framework:

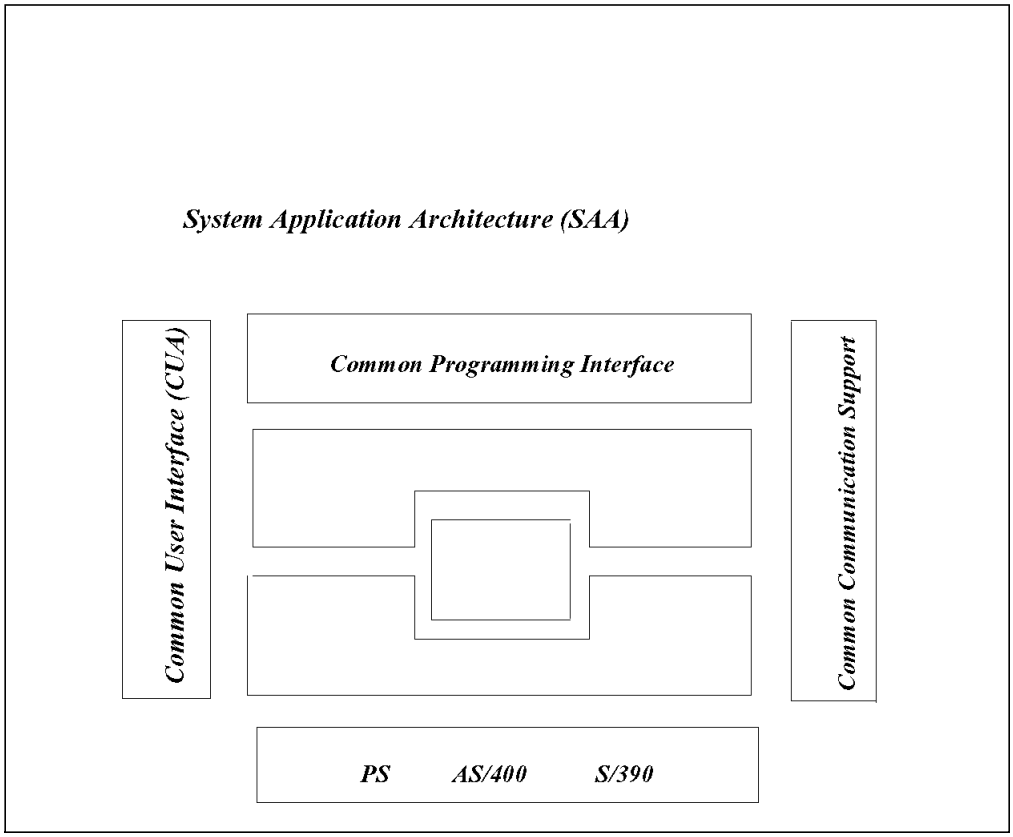


Figure 2. System Application Architecture (SAA) Model

SAA and CUA are major developments in the user interface area. Their guidelines are based on sound user interface design principles and object-oriented relationships. They specify common user interface components, techniques and guidelines for applying them. SAA and CUA provide guidelines for how the user will interface with systems using keyboards, menu selections, mouse, scrolling keys, etc., how the "Help" system should be built into the applications, how the use of colors should be standardize and how users should receive messages from systems and respond to those messages.

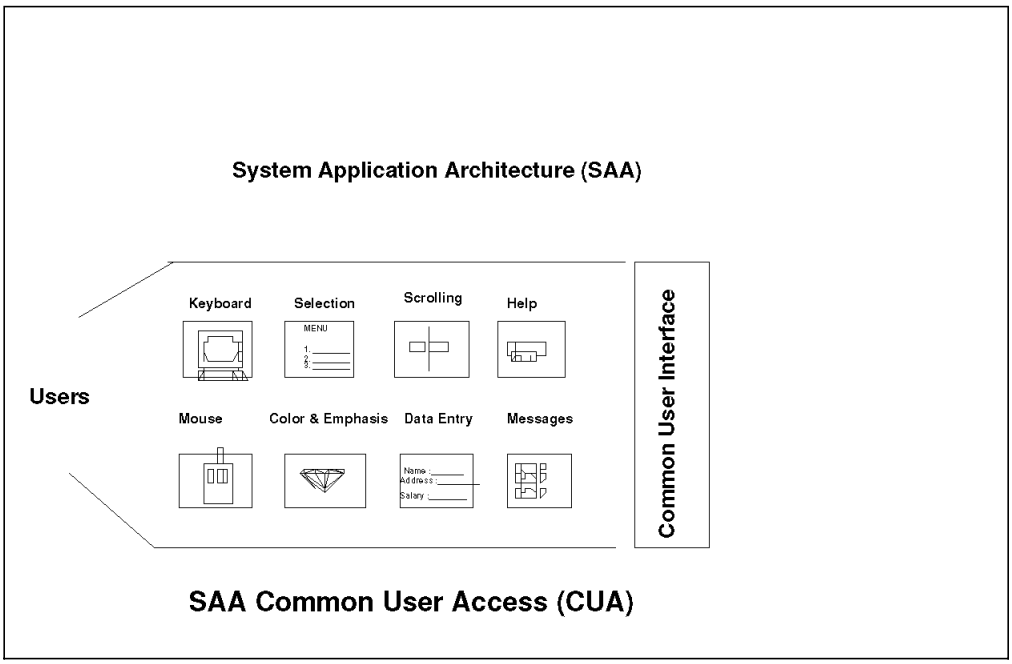


Figure 3. SAA Common User Access (CUA)

Like other major frameworks and architectures, CUA specifications have also changed in line with technological development and users′ needs and expectations. In its first announcement in 1987 (CUA87), it was assumed that an environment of personal computers would be intermixed with host-attached nonprogrammable terminals. It had, at that time, a goal of consistency and the transfer of a user′s knowledge between these environments. But personal computer technology and capabilities advanced rapidly. The gap between how a user could use a terminal and a personal computer widened significantly.

The CUA interface specifications as modified in 1989 (CUA89), focused more on the needs of personal computer users. CUA89 was still based on the main principles of CUA87 and still provided a rich set of user interface mechanisms, such as windows which are sizable and movable, standard menus, user interface controls and dialogs. The first product to be announced based on CUA89 was OS/2. OS/2 Presentation Manager provided end user interface as promised by CUA87 and CUA89. The multitasking capability of OS/2 and its Presentation Manager offered a graphical view of programs, data, results and the ability to run multiple applications concurrently.

CUA was further redefined in 1991 (CUA91). This time specifications to move user interfaces closer to the way users accomplish work in the real world were included. The OS/2 Workplace Shell* which was based on CUA91, redefined data and applications to create a set of familiar user objects and provide the ability for users to utilize these objects in ways that support a variety of user tasks.

While recognizing the importance of PWSs, one must not forget that large numbers of terminals (NPTs) are still in use and will continue to be used by computer users. To balance the need to support both NPTs and PWSs, CUA defines two categories of user interface models: Entry/Text Models and Graphic Models.

Graphic Model is best suited for PWSs running with OS/2 and makes extensive use of its windowing capabilities. Graphic Model was further extended as mentioned above to Workplace Model in CUA91. This supports integration of applications into an electronic version of a working environment. This model fully exploits GUI specifications.

Entry/Text Model is best suited for NPT environment and for applications that have panels with menu prompts. It is also intended for data entry intensive applications on NPT and may also be appropriate for such applications on PWSs.

CUA's evolution since 1987 is now maturing. More and more IBM and third-party applications are now available which have based their end user interfaces on CUA. CUA has influenced industry standards as well as customers. The user interfaces defined by SAA and CUA can be summarized as follows:

1. Entry model for NPTs. This is a single window based full screen with menu option selections.
2. Text model for NPTs and PWSs. This includes action bars and multiple windows with pull-down and pop-up facilities.
3. Basic Graphic Model for PWSs and Graphic NPTs.
4. Workplace Model for PWSs. This is iconic and object-based.

For more information on SAA and CUA refer to the following documentation:

- System Application Architecture - *IBM SAA Strategy (GC26-4784)*
- Object-Oriented Interface Design - *IBM CUA Guidelines (SC34-4399)*

Impacts:

- Provides consistency across key IBM operating environments to application users as well as developers. This will result in increasing their productivity and investment protection.
- Provides a sound base for EUI tool development. We have already seen products conforming to CUA and many more will be available in the near future. This means customers will have wider choices.
- The migration from one platform to another can more easily be kept transparent to users.
- Many CUA specifications have been accepted as industry standards at the same time CUA have whenever possible accepted open system standards based on OSF/Motif**. This has increased portability and coexistence of SAA and open systems in an enterprise.

1.2.6 EUI and Object Oriented Technology

The user interfaces based on object orientation are called Object Oriented User Interfaces or OOUI. When using OOUI, the user's focus is on objects. The user sees and uses representations of their data and applications in the form of objects, and each different kind of object supports actions appropriate for the object. The objects are composed of and contain other objects, which can be used individually or collectively. The hierarchy of these objects and their classes and subclasses is closer to the real world. An OOUI allows a user to focus on objects and work with them directly. This more closely reflects a user's real world way of doing things rather than having to go through an application to get to these objects. The user performs actions on objects using various

techniques, including point-and-select, menu option, and direct manipulation. These objects are often represented on a user′s screen as icons (small graphic images that help a user to identify an object). Thus, icons are used to provide a concise, easy to manipulate representation of an object regardless of how much additional information the object may contain. The user can open an icon to see this additional information in a window if he/she wants. However, iconic representation is not a must for OOUI. On nongraphic, nonprogramable terminals enough details are given for users to perform desired operations on the objects. The user can this by going through hierarchically organized menus or even by commands. AS/400 object orientation is an excellent example of such an interface.

The Workplace Model of CUA91 is based on object orientation. OOUI offers many benefits, to operators, programmers or application users:

- Direct access to and focus on objects:

  By giving a user direct access to objects, an OOUI lessens the need for a user to be aware of programming that is providing functions that the user needs. Instead, the user can concentrate on objects and actions that the user wants to perform.

- Removal of user interaction requirements and obstacles:

  An OOUI removes user interaction requirements and obstacles that some graphic user interfaces still impose. For example, by removing the necessity for starting and running programs, you can simplify the learning process for each user. The learning process is simplified because the user has only one process to deal with, opening an object as opposed to starting an application and then finding and opening or creating a file.

- The internal structure of the object is hidden: For example, a data base file is made up of four elements, i.e., a space, a cursor, data and data index. In object orientation all this detail is masked from the user and he/she only knows one element, i.e., the file. Also, the user does the same operations on objects whether it is a file, document, letter, folder or even a network definition.

An implicit benefit of an OOUI is that the designers have to think more precisely about distinctions between object classes that are useful to users. This ultimately results in better quality applications.

The announcement of AS/400 by IBM in the late ′80s was a significant progress towards providing object-based user interface. The machine and its operating system, OS/400, were designed based on object orientation. Each element, whether a file, a program, a library, a screen, a menu or even a network definition was treated as an object and was treated in the same way. Similar operations which could be command or menu driven could be done on all elements. By selecting the proper menu options, authorized users could do whatever they wrere authorized to do, without having to give commands. On later models of AS/400 a new graphical user interface expanded the usability of the AS/400 system significantly. End users and system operators were able to take advantage of direct manipulation capabilities of Programmable Work Stations (PWS), i.e., point and click or drag and drop, rather than having to type commands or give menu options to do the same things.

The AS/400 object oriented user interface is designed to allow each user to comfortably begin work and grow in productivity using menus, layered entry

panels, list panels and command lines. It is a very consistent interface across all AS/400 functions and introduces consistency between programmable workstations and nonprogrammable terminals, without compromising the potential and strengths of either.

The other products currently using object based GUI's are: Flashpoint**, Mozart**, ENFIN** and Ellipse**.

Additional Reference Information:

- *Object Technology in Application Development*
- *Object-Oriented Interface Design IBM CUA Guidelines (SC34-4399)*.

Impacts:

- Object Oriented User Interface (OOUI) makes object oriented programming easy. Without these interfaces OOP would have been extremely difficult.
- Faster development through standard objects and their codes re-use rapid prototyping and modeling.
- OOUI have made it possible for a business analyst to actively participate in application development or even to replace an IS analyst.
- Applications implementing OOUI principles are readily accepted by users.

1.2.7 EUI in Client/Server Computing

The Client/Server (C/S) Computing has been defined in a number of different ways. However, in simple terms, it is a multicomputer environment where processing is done and is most suited to do so. The other terms used for this are cooperative processing, distributed processing and network processing. We will not go into details about describing what C/S computing is. From a User Interface point of view we will visualize the most common Client/Server environment and then discuss the requirements and progress being made in this area. The most common C/S environment is where a number of PWSs are connected to a computer which has more computer resources (however, this is not necessarily the case). This computer provides some services to PWSs connected to it. It is called the server, and since PWSs request some services they are called clients. Depending on application requirements this role may interchange. When we talk about EUI we mean interface with the client, which most of the time is PWS.

The application user interface in Client/Server computing is basically the same as all other environments we have seen above. It is built around GUIs and further extended to OOUI and workplace models. The significant impact of EUI in C/S computing is in application development. The new development tools have been made available and existing products are evolving to take full advantage of the GUI and C/S capabilities of PWSs. These products distribute development work to client and server to improve productivity and system performance. The modeling, designing and code development part is done on client or PWS where facilities such as windowing, iconic objects, cut/paste, etc., significantly increase development and productivity.

A client/server application can be developed in several ways. However, their user interface can be built in the following ways:

- Distributed Presentation - The GUI presentation part may be split up across both client and server. This interface could be used when a large part of the

users′ needs could be met by NPTs attached to the host, but some specific requirements could only be met by PWS using GUI. The users of legacy applications requiring the power of personal computers in selected areas can go for this option.

- Remote (client) Presentation - The GUI presentation may reside entirely on the client (PWS) with the rest of the application residing on the server or spread over both. This would be good for applications that require the power of PWSs and the flexibility of GUI. With the gap between the costs of personal computers and NPTs narrowing and also the availability of tools improving, this is becoming the preferred option.

IBM′s Cooperative Development Environment/370 (CODE/370), CODE/400, and HighPoint are some of the development tools that take full advantage of GUI and PWS and develop applications for host environments.

Impacts:

- Processing is done where it is needed and most suitable.
- Development load is distributed across clients and servers so host availability for production jobs increases.
- Use of CASE and OOP techniques increases productivity in development work.
- Client Server applications are more complex to develop. However, GUI-based tools are now helping this development.

1.2.8 EUI in Multimedia Environment

With the advancement in personal computer technology supported by advancement in GUI technology, we now see large numbers of graphic and image applications emerging which uses high performance graphics. This has not limited itself to computer games only but many business applications have emerged. With this the use of animation, sound, voice recognition, still video, and motion video is increasing. This has created a need for information processing technology called “Multimedia.”

Multimedia is comprised of support for elements such as animation, touch screen, still video, motion video, sound, images, etc. It uses computers to integrate and control these elements. The EUI in multimedia is based on GUI workplace models with some more elements such as touch sensitive displays included in it. Figure 5 indicates the evolution of applications using EUIs from entry/text based to multimedia based over time. Multimedia information can enrich and improve communication between people and between users and their computers. However, designing a product with a multimedia interface is more challenging than designing a product that uses text and graphics. A designer or a team of designers must be well versed with graphics as well as audio and video production aspects.

Impacts:

- Development of multimedia applications is very complex.
- Increased demand for multimedia specific skills.
- Demand for systems with high speed processing and large databases increased.
- Opens up computer power to new applications and new users.

1.2.9 Open Systems User Interfaces

The definition of Open Systems has changed over time. UNIX does not have to be called “Open.” Different international standard organizations have defined open systems. The most accepted definition comes from IEEE:

“A comprehensive and consistent set of international information technology standards and functional standard profiles that specify interfaces, services, and supporting formats to accomplish interoperability and portability of applications, data and people′s skills.”

In open systems, in the area of user interfaces, attention has been focused on the emergence of GUIs, based on windows and icons, that utilize input devices such as buttons and mouse devices. While there are still many competing definitions of how a visual interface would be implemented, most are sufficiently intuitive to serve users even if there are differences in implementation. Examples of GUIs in open systems include Motif, Open Look, Windows, HP NewWave**, and the Macintosh interface.

IBM is committed to open systems and promises to make AIX*, its UNIX operating system which is now available on Personal Systems, RS/6000, S/390s* and the follow-up systems of S/370s, a true open operating system. User interface in AIX is offered by AIX presentation services. These services are provided through the X-Window system and AIX Windows. AIX Windows is a user interface management application/toolkit that is based on the OSF/Motif.

OSF/Motif is a GUI based user interface combining a toolkit, presentation description language, window manager and style guide. The toolkit is a collection of software modules called widgets and gadgets for building OSF/Motif compliant applications. Motif widgets and gadgets are based on an earlier widget library developed by Hewlett-Packard**. The presentation description language can be put into simple text files and describes the visual properties of the initial states of interface components. The window manager provides the functions to set up and handle window operations including such things as sizing, iconizing and arranging windows. The style guide provides information on how new widgets and gadgets should look and react to developing applications that conform to OSF/Motif specifications. The OSF/Motif programmer′s reference manual (OSF-0-M-00889-001) provides the details of how to build new widgets and gadgets.

Motif is based on the X-Window System developed by X-Window Consortium at the Massachusetts Institute of Technology. IBM intends to provide AIX Windows and X-Window systems** for its SAA platform, i.e., MVS, VM, OS/400 and OS/2.

IBM also promises to provide an consistency between SAA, CUA and OSF/Motif so that customers can implement multivendor systems in such a way that hardware and software platforms remain transparent to users.

To supplement the evolving Client/Server environment, IBM has announced that it will divide AIX into client and server versions that will share as much of the AIX infrastructure as possible. A/UX is the UNIX operating system for Macintosh (Apple) with powerful windowing features. PowerOpen is the joint IBM and Apple operating system which is based on IBM′s AIX and Apple′s A/UX. The aim of PowerOpen is to make Macintosh productivity applications and AIX technical applications accessible to clients through common user interface. The

"PINK" system, another joint venture of IBM and Macintosh, promises to provide an object-oriented environment to PowerOpen.

Many software houses have started developing tools which make it possible to develop EUIs that are portable across different platforms. One such example is Neuron Data Open Interface. This set of tools builds GUIs which are portable across five major windowing environments: Windows, OS/2 Presentation Manager, OSF/Motif and AT&T Open Look.

Impacts:

- Mushroom growth of window based applications.
- OSF/Motif-based applications run on many platforms from different vendors so it provides wider choices to customers for meeting their information processing needs.

1.2.10 Concluding Remarks

As advancement is being made in an exponential manner in Information Technology, more and more functions are being automated. A system programmer in the '60s and '70s never imagined that a multiuser/multitasking system could be installed and made operational within hours without going through the tedious tasks of system generation. This is possible because the system management part is being automated continuously and even now a system programmer is not required to know what is happening inside the machine. He/she communicates with the machine and operating system through user interfaces which mask the system complexity from him/her. Similarly, so is the case with application development. A large part of the application can now be built without the writing code. The fourth GL tools, the application generators and CASE tools have reduced coding to a large extent with the help of their easy-to-use user interfaces. Advancement in this area is also continuing. The story is the same in data management, network management and other aspects of information processing. This means that outside software laboratories, large numbers of computer users, whether programmers, system programmers, IS analysts, business analysts, application users or operators, will be interacting with computers through EUIs (And this number is continuously increasing). Thus, End User Interface (EUI) is becoming the most important element in information processing.

As we are moving towards the end of this century, the technology and functional enhancements to EUI components will be optimized to PWSs and progress on EUI for NPTs will be capped but support will continue. The new level of EUIs will be designed to support Voice, Images, Video information type (i.e., EUI for Multimedia) based on the GUI workplace model. The user interface processing load of an application and that part will run on PWSs. Figure 4 gives you some idea about trends in EUI technology based on customers needs.

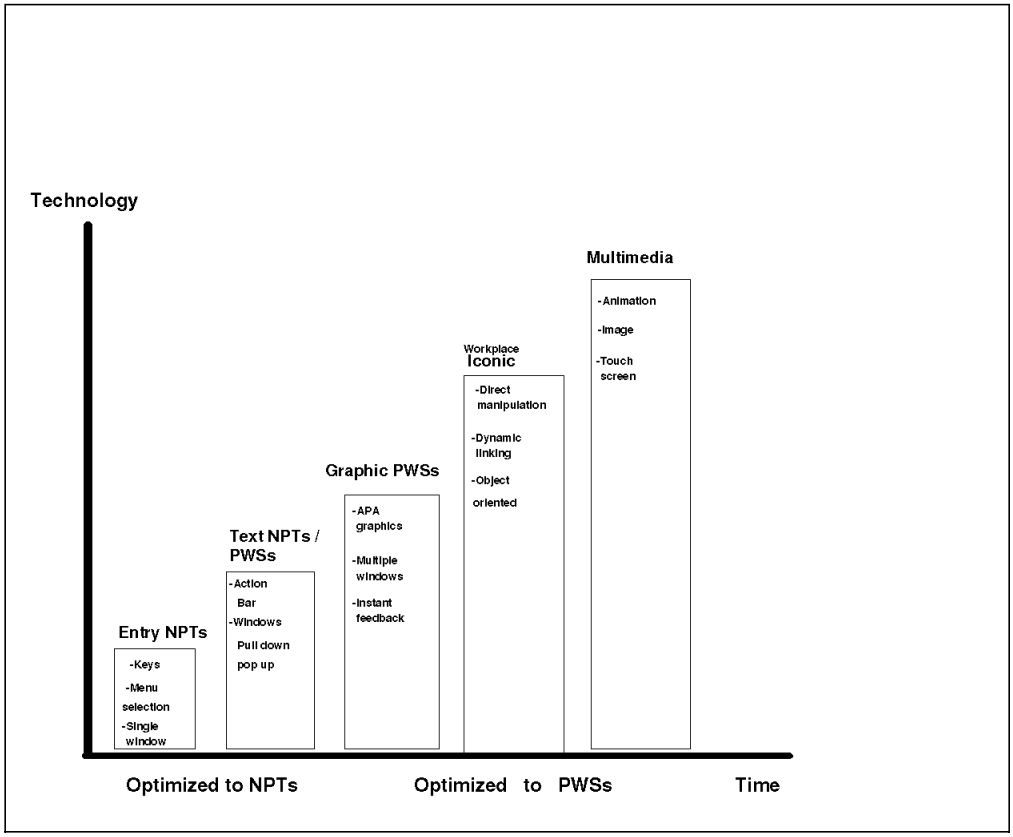


Figure 4. EUI Extensions

In this important area IBM strategy remains based on dynamically changing customers' requirements. IBM will continue to support SAA and CUA evolution to meet its customers on the proprietary systems. At the same time IBM will very aggressively participate in the open system area and whenever industry standards are available will meet those standards. In the absence of those standards IBM will develop its own. IBM will also incorporate open system standards in CUA so that CUA-based interfaces become part of a heterogeneous environment. The overall goal of IBMs strategy is to meet its customers' need to build and use applications that run on multiple platforms. The CUA Graphic User Interface's (GUI) workplace model will be IBM's preferred model. However, support and required enhancements in the entry model and basic GUI model will continue.

See appendix A for an overview of types of customer and their EUI needs.

Chapter 2. Design Principles/Guidelines

In this chapter we will discuss the design principle and considerations for developing EUI products and applications. Our discussion will also be oriented towards the EUIs currently in use and where technological advancements are being made, in other words, GUI based workplace models of EUI.

2.1 Introduction

As defined in the first chapter EUI is for enabling computer users to interact or communicate with computers. We have also seen how technology in this area has progressed over time from the EUI entry model to the GUI workplace model and beyond.

Considering the importance of EUI in information processing, many frameworks have been announced and tools are being developed based on them. IBM SAA Common User Access (CUA) stands out in terms of clarity of specifications and details about implementations. In the real world OSF/Motif provides tools and APIs for developing user interfaces. In this chapter we will summarize the principles and guidelines from these frameworks and available specifications with a large part coming from IBM, SAA, and CUA documentation.

These guidelines will not only be helpful to designers but also to IS executives involved in EUI product selection. For more information on EUI principles, guidelines and techniques see the following manuals:

- *SAA and CUA Guide to User Interface* (SC34-4289).
- *SAA and CUA Advance Interface Design Reference* (SC34-4290).
- *OSF/Motif Programmers' Reference Manual* (OSF-0-M-00889-001).

The major objectives of EUI design can be summarized as follows:

- Increases users' productivity. (Users meaning besides application users, developers, system programmers, and operators).
- Increases users' satisfaction with the system they are using.
- Reduces users' error rate. This can also be termed as increased productivity.
- Helps users to transfer knowledge across products. This means that if a user learns to use one product he/she should be able to use most of the other products as part of his/her overall system environment.

Basically, the user can be divided into three categories: novice users, knowledgeable intermittent users and experts (frequent users). For a successful product, it is very important that it addresses all these categories. A novice user of today may become an expert tomorrow.

The principles a designer must adhere to while developing EUI tools to achieve the above objectives can be grouped according to these categories:

- Placing the user in control of the user interface
- Reducing the user's (personal) memory load
- Making the user interface consistent

- Simplicity and clarity
- Balanced performance with functions and features

If a designer combines these design principles with their knowledge of the requirements of the user of the product, the resulting product will meet the objectives. Now let us discuss these categories of principles and guidelines to implement them.

2.2 Place a User in Control of the User Interface

A user should always be able to communicate with the computer and should never feel that the computer is in control. Whenever possible, a designer should avoid program-driven sequences that prompt a user through fixed steps and directive messages. Program-driven interaction is like travelling in a train, for example, a user must go where the program takes him/her and should reach the destination after following a fixed route. A designer should aim for user-driven interface which is like driving a car, giving enough flexibility to users to reach their destination through the route of their choice, according to a schedule suitable to them. The product should allow a user an alternative course of action and should not limit a user by imposing the “correct” sequence for accomplishing a task. In general, users should never feel that they are doing something wrong and the product should always be able to respond to any user action.

This principle is difficult to implement but not impossible. Following are some guidelines to implement this.

2.2.1 Usage of Modes

A mode is a state of a product in which only certain actions are available. That is, modes restrict a user′s options. For most of the users the confusion starts because of their lack of understanding of the mode in which they are working. Many times we hear complaints such as, ″Isn′t it funny, this command works OK sometimes and sometimes it doesn′t!″ In fact, the command works only in certain modes and the user is unable to differentiate from them.

However, modes in many situations are useful. For example, they can extend the capabilities of input devices by allowing several actions to be accomplished with the same technique, key, button, etc. Modes can also help an expert user perform a series of actions very quickly.

Modes can be useful in directing a user′s interaction with the product. However, designers have historically overused modes without regards for how modes affect users. Users can feel powerless when a mode restricts their actions. For a long time we have been seeing products which will not allow users to give a command which is not valid for the mode they are in. For example, if you are working with an editor you could only give commands valid in edit mode. If you want to give other commands you have to come out of the edit mode and enter another mode. The IBM VM system was the first IBM system to give some relief to users in this area. In VM users could also give commands for the modes higher in hierarchy than the one they were working in.

While working in different modes, users tend to forget which mode they are in unless the product clearly indicates the current mode. A user can be puzzled when an action leads to an unintended result. For example, pressing a right

arrow key in a typing-mode typically moves the cursor to the right. In a line-drawing mode, the same key extends a horizontal line by some increment. If a user presses the right arrow key intending to edit some text, and instead draws a longer line, the user can be confused, surprised or even angry.

To avoid some of the problems associated with modes, a designer should try to use only those modes that require an ongoing user action, such as pressing a mouse button or a keyboard key, to maintain the mode. In addition, the designer should specify some kind of mode indicator. For example, a pointer or cursor could have one visual representation when the user is in text-editing mode (for instance, I-beam pointer) and another when the user is in a line-drawing mode (for instance, cross-hair pointer).

If a designer finds it necessary to use a mode for a particular part of a product, the designer should keep the scope of the mode narrow and should allow a user to continue to interact with other parts of the product while the mode is in effect. For example, if a product requires more information from the user and displays a message window to elicit the information, the user should still be able to scroll the underline window and interact with other parts of the window not affected by the lack of that information. The user should also be able to interact with other objects.

In general, a designer should use modes with caution and should make them obvious, easy to get out of and get into.

2.2.2 Displaying Helpful Messages

It is possible that the user will interact with a product in a way that the designer did not anticipate. In this situation the product should indicate to the user that it cannot interpret the user′s action. Typically, a product should display some kind of helpful message. In simple situations an audible cue or a graphical cue would be sufficient. Different levels of help messages can be provided. For example, level-1 can be for experienced users and level-n can be for novices.

Because a situation in which a message is displayed is often a situation in which a user needs the most support, messages should be clear and provide a mechanism for the user′s interaction. A message should describe the situation objectively, without placing blame and should help the user correct the situation.

For example, one of the most common examples is if a user inserts a diskette that has not been formatted, and then tries to save an object to the diskette, the product should display a message that tells the user that the diskette is not formatted. The message or message window should include mechanisms that allows the user to complete his/her task, that is, to format the diskette without leaving the window or to take some other course of action.

2.2.3 Providing Immediate Feedback

The results of the user′s actions should be obvious immediately. If the results are not as expected, the user can choose an alternate action right away. For example, when a user selects a choice to change the font of some selected text, the appearance of the selected text should change immediately. The user can then decide if the resulting effect is desirable or can select another choice. The other example could be spreadsheet products where change in one figure should immediately change the resulting subtotals and totals. The product should also allow users to “undo” the impact of a user′s action. If the user

wants to, he/she should be able to “re-do” the work he/she has undone. In other words, no user action should be irreversible. In most cases, particularly in the case of irreversible actions, the product should display a message that gives an outcome of the action taken. Also it should indicate alternate actions which could be taken.

If the results of a user′s actions cannot be made obvious immediately, for example, if a network delay interferes, the product should still provide some kind of feedback. For example, it should be indicated to the user that the action is being processed. However, the designer should make sure that a product′s feedback does not interrupt a user′s work.

2.2.4 Consider Users with Different Skill Levels

Much of the user interface design is focused on novice or casual users. However, products should also offer features for more expert users. When designing a user interface, a designer should provide a way for a user to proceed at a comfortable pace. The design should also provide a way for a user to go beyond the basic level of knowledge required for frequently used features. As the user′s expertise increases, the user should be able to discover more advanced features.

For a novice user, a designer should rely heavily on visual cues and should avoid making a user remember details or key in a lot of information. To accommodate an expert or experienced user the design should provide a mechanism, such as shortcut keys, commands or condensed sequences of steps. The expert should also be able to turn off the display of some of the information that is not then required.

In general, an interface should be flexible enough to accommodate a full range of users, but the designer should make sure that the interface must serve the needs of the primary users.

2.2.5 Transparent User Interface

A user interface provides tools that help a user accomplish a task. Therefore, an interface should focus a user′s attention on the task or end product. A user is only incidentally interested in the EUI tools that are menus, pointers, keyboards, fastpaths, icons, windows, etc. A user is normally interested in calculating tax returns, preparing payrolls, replenishing inventory, preparing a project plan or a machine design. The interface designer should make sure that the tools provided by a user interface do not get into the user′s way. A good user interface helps the user concentrate on the business task they are performing and relieves the user from the technicalities of using interface tools.

2.2.6 Customizable User Interface

No two users are alike. They have varying interests, motivations, and backgrounds. To accommodate individual differences, a designer should create flexible interfaces that each user can customize according to personal preferences. He/she should be able to customize menu choices, sequences of steps in a process, colors, sizes of windows/characters and any other aspect of the user interface. A designer should never underestimate the user′s creativity or desire for imprinting a personal style on a computer.

Allowing users to completely customize EUI can lead to higher productivity and higher user satisfaction. However, a designer must provide defaults that are satisfactory to most of the users and that a user can revert to.

2.3 Reduce User′s (Personal) Memory Load

A user should never have to rely on his/her memory. A product should be able to “remember” for him/her. The designer must know that people are better at “recognition” than at “recall.” A product should present alternatives and let the user choose from among them. For example, a product could provide a list of items, such as choices in a menu. A user can recognize choices in a menu without having to recall commands or their syntax.

A product should also provide reminders to help a user keep track of the task at hand. For example, a product could provide visual cues, (such as highlighting), progress indicators or textual cues, (such as status messages). Highlighting can remind a user that an object is selected and a progress indicator can remind a user that a process is underway.

Another way to avoid overloading a user′s memory is to provide default settings and save previously selected settings. For example, a user might want to change the colors that appear on the screen. After experimenting with various colors the user might settle on a particular combination and save those settings. At a later date, however, the user might decide to revert to the original colors. By providing for a default setting for screen colors, the product relieves the user of the responsibility of remembering the original colors.

2.3.1 Meaningful and Concise Object Classes

When designing objects for object-oriented user interfaces, a designer should consider the tasks a user will want to accomplish and then should ensure that the characteristics of the object support the user′s tasks.

A designer should clearly define the properties of each object and should establish a hierarchy of object classes based on these properties.

The objects should be designed so that a user can easily recognize members of an object class and can understand what distinguishes one class of objects from another. The distinction among classes should be meaningful to a user and should not be based on underlying programming requirements.

An object hierarchy should be concise. So should the visual representation of the objects in the hierarchy. Typically, each object in an object class is represented by the same icon. The individual objects are identified by different icon labels which indicate the name of each object. While creating an object class hierarchy the designer should be aware of the fact that too many object types with too many iconic representations can make the user uncomfortable. To keep users focused on the correct objects, the user should keep the number of object classes to a minimum without loosing an object′s specific characteristics.

2.3.2 Concrete and Recognizable Objects

When a computer object resembles a real-world object in appearance, behavior or both, the user can transfer knowledge about the real-world object to the computer environment easily. By including familiar objects in a product, a designer can help users learn to use the product more quickly. The visible representation of computer objects should be easily recognizable and a computer object should resemble its real world computer counterpart.

2.4 Consistent User Interface

Consistency helps a user transfer knowledge from one product to another and helps a user predict how something new will work. To create a consistent user interface, a designer should develop paradigms that provide for identical implementation of common functions throughout the product. For example, IBM SAA, CUA or OSF/Motif guidelines specify that a user should be able to use the same technique for editing text, regardless of where text appears in their respective environments.

For any single design decision, a designer must consider whether being consistent with respect to one component of an interface can affect the consistency of other components. Some components might be consistent in shape, location and color. Others might be consistent in interaction techniques. A designer should make sure that components are consistent in a way that users would expect.

Sometimes it is impractical or impossible to be completely consistent. In that case, a designer must make consistency compromises based on the knowledge of the user′s conceptual model and should be consistent in whichever way seems more natural to a user.

2.4.1 Sustaining the Context of a User′s Task

A product should maintain a useful point of reference while a user works on a task. For example, when a user adds objects to a folder, the appearance of the folder′s window should remain the same while the appearance of the window contents changes.

Also, a user should be able to complete a step or a series of related steps without having to switch between input devices. For example, a user should not have to use a pointing device to scroll through text while editing that text from a keyboard. The text should scroll automatically when the cursor reaches the boundary of the area (or window) the user is working in. The product should also use the keyboard′s scrolling mechanism in this situation.

2.4.2 Continuity Within and Among Products

A designer should not discount a user′s experience with other user interfaces, such as those provided in prior versions of a product, or those generally accepted as industry standards. Instead, a new product or a new version of the existing product should be built on a user′s knowledge and experience. Therefore, the designer should be cautious in changing the behavior of an object from one version of the product to the next. A designer must test a new behavior to make sure that its benefits outweigh the drawbacks of forcing a user to relearn the object′s behavior. One way to accommodate both new users and

experienced users is to provide both old and new behavior for an object and let users choose which to use.

2.4.3 Aesthetic Appeal

The appearance of a product's interface significantly affects a user's attitude towards that product. Inconsistent design and haphazard placement of the objects can confuse a user and can contribute to a user's dissatisfaction with the product.

When designing the appearance of a user interface, a designer should adhere to generally accepted practices for information presentation. By skillfully using background space, colors, proximity, size and shape differences and other components of visual communication, a designer can make an interface more efficient and effective and can increase the users' satisfaction with the product.

2.5 Simplicity and Clarity

Part of this aspect of user interface has already been discussed. But its importance demands that we discuss it further under a separate heading. Simplicity is the most important aspect of user interface. The first impression a user gets about an application is from how it looks. A good visual presentation contributes to simplicity. Interface should look good and distinctive but not too flashy. For example, an architect wants a new building to have a presence of its own but does not want it to look out of place in the surrounding environment. Users are more likely to be afraid to try an application that is visually overwhelming. If it looks complicated or too different, users will treat it with caution.

Clearly labelled controls contribute to simplicity. At one time or another, each of us has been a beginning user. The rate at which we become more experienced is based on several factors, including our technical interest, need, and how frequently we use the product. In fact, some products are used so infrequently that users cannot be expected to remember how they operate. For example, a tax return preparation program might be used once a year only. In that program the command for calculating and printing a tax return could be labelled as "Calculate and Prepare Tax Return." If the same program is designed to be used by accountants who understand the process of tax return preparation and are using this program frequently, the same command can be labelled as "Tax Return."

Both novices and experts appreciate the clarity. No one will appreciate ambiguous labels and commands.

2.6 Balanced Performance with Function and Features

No matter how good a user interface is, unless it is supplemented by good system performance it will not be acceptable to users. For example, to be more helpful to users you design an interface where you display a part description with a part number of all alternate parts in a situation where a part in demand is out of stock. This information is coming from the database server or central host. This is an excellent help to application users, but make sure that the user does not have to wait unusually long for this information. A designer needs to evaluate each function and feature offered to users not only from a user point of view but also from a system performance point of view. Balance the users' need

and system performance needs. The designer should be aware that good performance is also one key user requirement.

Chapter 3. Components

A brief discussion on different key elements of EUI on PWS is included in this chapter. The discussion is restricted to the GUI workplace model. However, these elements can also be used for other models.

3.1 Introduction

The elements described here are used in different kind of interfaces and many IBM and non-IBM products are already using it. Please note that all elements are not included here. The objective is to provide some more details to IS senior personnel who wish to know more on this subject. However, we have taken care not to leave out any important element. In the user interface area the most accepted specifications are IBM SAA CUA and OSF/Motif. Whenever possible a brief comment on similarities and differences in these two specifications are included. While doing, so we are restricted to EUI components only.

For more information on this subject please see :

- *SAA CUA Advance Interface Design Reference* (SC34-4290).
- *Object Oriented Interface Design - IBM CUA Guidelines* (SC34-4399).
- *OSF/Motif Programmer Reference Manual* (OSF-0-M-00889-0001).

Products like OS/2 Presentation Manager implement support for these elements and provide application programming interfaces (APIs) so that users can build applications based on these elements. Where support for these element is not provided the application must provide them to meet the requirements of the GUI workplace model.

3.2 Key Components and Their Description

The following sections describe the components of GUI workplace model.

3.2.1 Keyboard and Mouse

Both keyboard and mouse are the primary means of user interaction in the workplace style of graphical user interface. Both CUA and Motif provides specific guidelines for using these devices. Applications should provide equal support for both these devices and both should virtually be interchangeable. CUA and Motif both support this principle. However, CUA puts greater emphasis on the mouse in the graphical model.

Mouse functions are basically the same between CUA and Motif. Motif assumes a three-button mouse and CUA assumes a mouse with two buttons and a third optional button which is application defined. Motif also defines functions of the third button to be application specific.

3.2.2 Workplace

The workplace is a container on your screen that holds all objects you deal with while using your work station. It usually fills the whole screen and serves as a electronic desk-top for a user's work. Objects in the workplace for different user's tasks are represented by icons. The users are allowed to include or delete any object from the workplace and also to arrange them the in any way they want.

The workplace is based on the GUI Workplace Model specified by SAA CUA and IBM OS/2. The operating system for IBM Personal Systems is the first product to implement this. Many other vendors now promise to implement the workplace model in their PWS products.

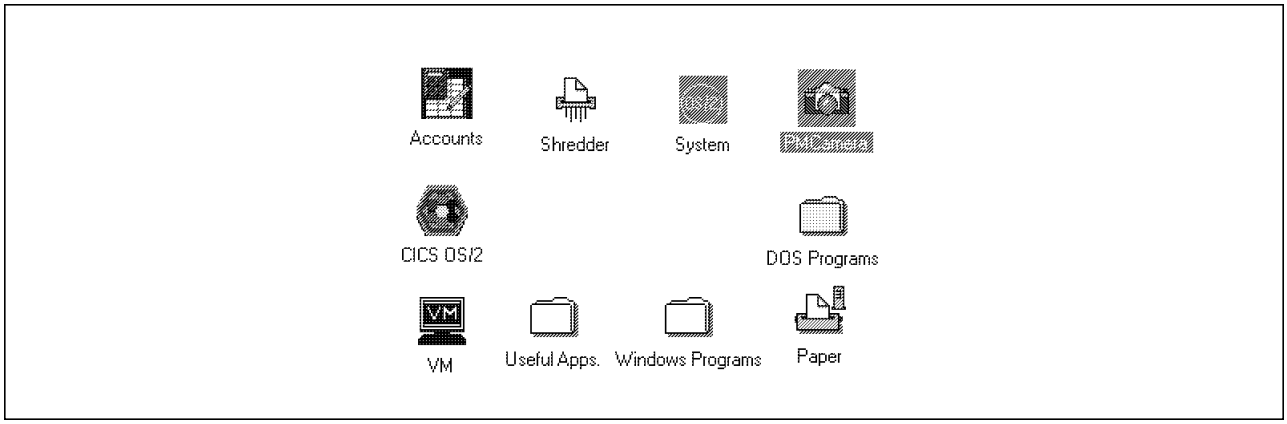


Figure 5. Workplace, the Container which Holds Objects

3.2.3 Cursors and Pointers

Cursors and Pointers are visual cues that indicate where a user's next interaction with the user interface will take place. They provide a way for a user to select and interact with things that appear on the workplace.

The Pointer: Typically only one pointer appears on the workplace at a time. It is associated with the user's pointing device, such as mouse, trackball or joystick. When a user moves the pointing device, the pointer moves correspondingly, and when the user presses a button on the pointing device, the object that the pointer is on is affected or selected.

The pointer is usually shaped like an arrow. However, the shape of the pointer can change to indicate the type of action being taken. Products can provide product specific pointers. For example, many word processing products provide an I-Beam pointer when it is over the text to be edited. The graphic products provide special Cross-Hair pointers to help the user draw lines accurately.

The Cursor: According to CUA specifications, there is only one cursor on the workplace at a time, and it is associated with the user's keyboard. There are two types of cursors, a Selection cursor and a Text cursor. The selection cursor indicates which item user can interact from the keyboard. For example, a selection cursor can indicate which item can be selected or which item can display a pop-up menu.

The text cursor is used to enter data through the keyboard. The text cursor is further specified as the insertion cursor and replace cursor.

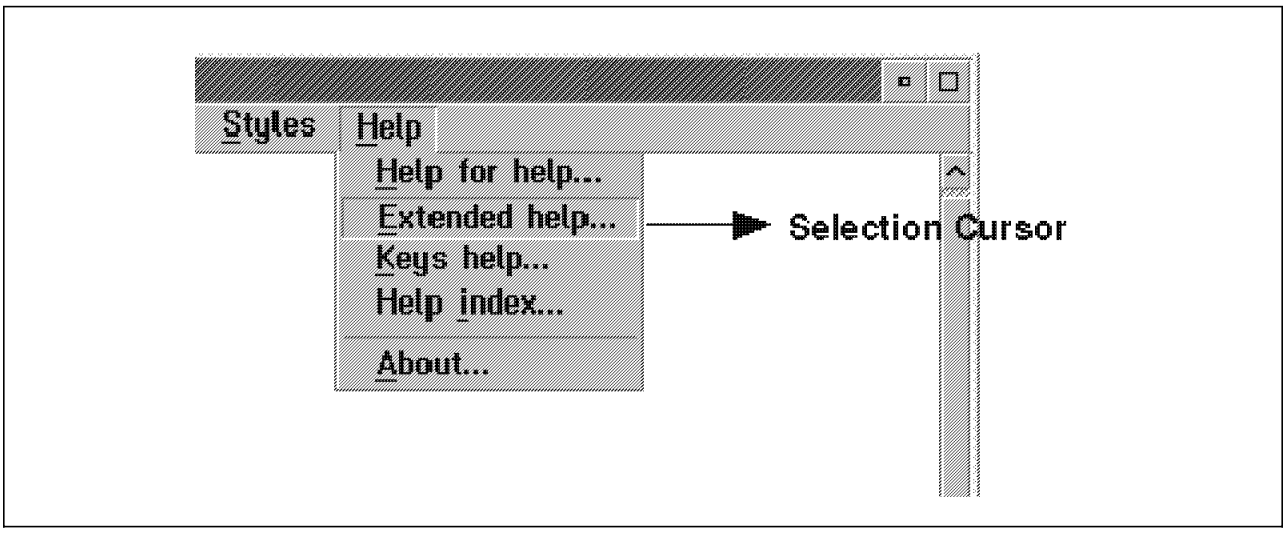


Figure 6. Selection Cursor

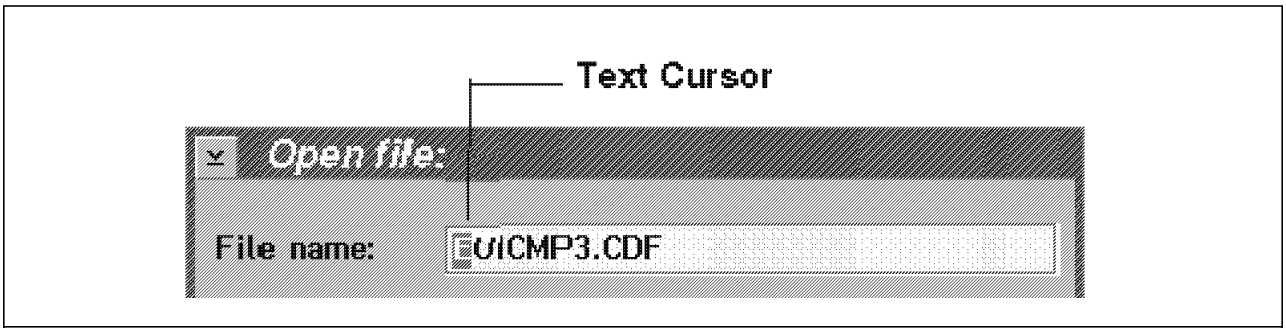


Figure 7. Text Cursor in Replace Mode

A user can move the cursor by using cursor movement keys given on the keyboard,for example, Arrow keys, Page-up, Page-down, Home and End keys. A user can also move the cursor by pressing and releasing a button on the mouse or the other pointing devices. For example, when a user presses the selection button on the mouse, the cursor moves to the position of the pointer.

Motif also defines two types of cursors, a location cursor and a insertion cursor. The location cursor reflects movement of the cursor around the screen using navigation keys and indicates current keyboard input focus. The insertion cursor is used to indicate where text may be entered. The specifications about further divisions of the insertion cursor as we have in CUA is missing in Motif.

3.2.4 Shortcut Keys

A shortcut key is a key or a combination of keys on the keyboard that a user can press to select a choice from a menu. The menu need not be displayed. Shortcut keys are provided for choices that are used frequently. These keys provide quicker methods of interaction, particularly for experienced users who are likely to prefer remembering the combinations for displaying and navigating through a menu for each desired choice.

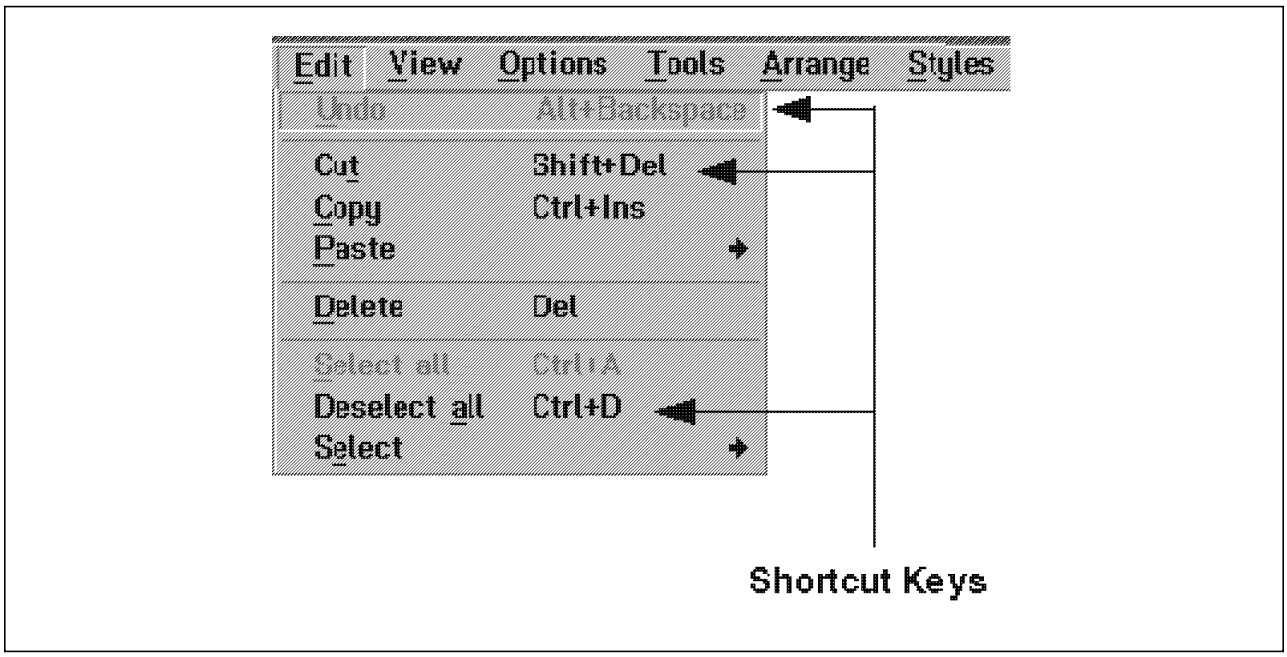


Figure 8. Shortcut Keys

3.2.5 Windows

A window is a part of the workplace through which a user can view an object. A window is bounded by a window boarder, which separates the window from other windows. Within the windows there are mechanisms that allow a user to manipulate the window and its contents. SAA CUA and OSF/Motif interface specifications provides two types of windows: a Primary window and a Secondary window. A primary window appears when a user opens an object; it is where the main interaction between user and object takes place.

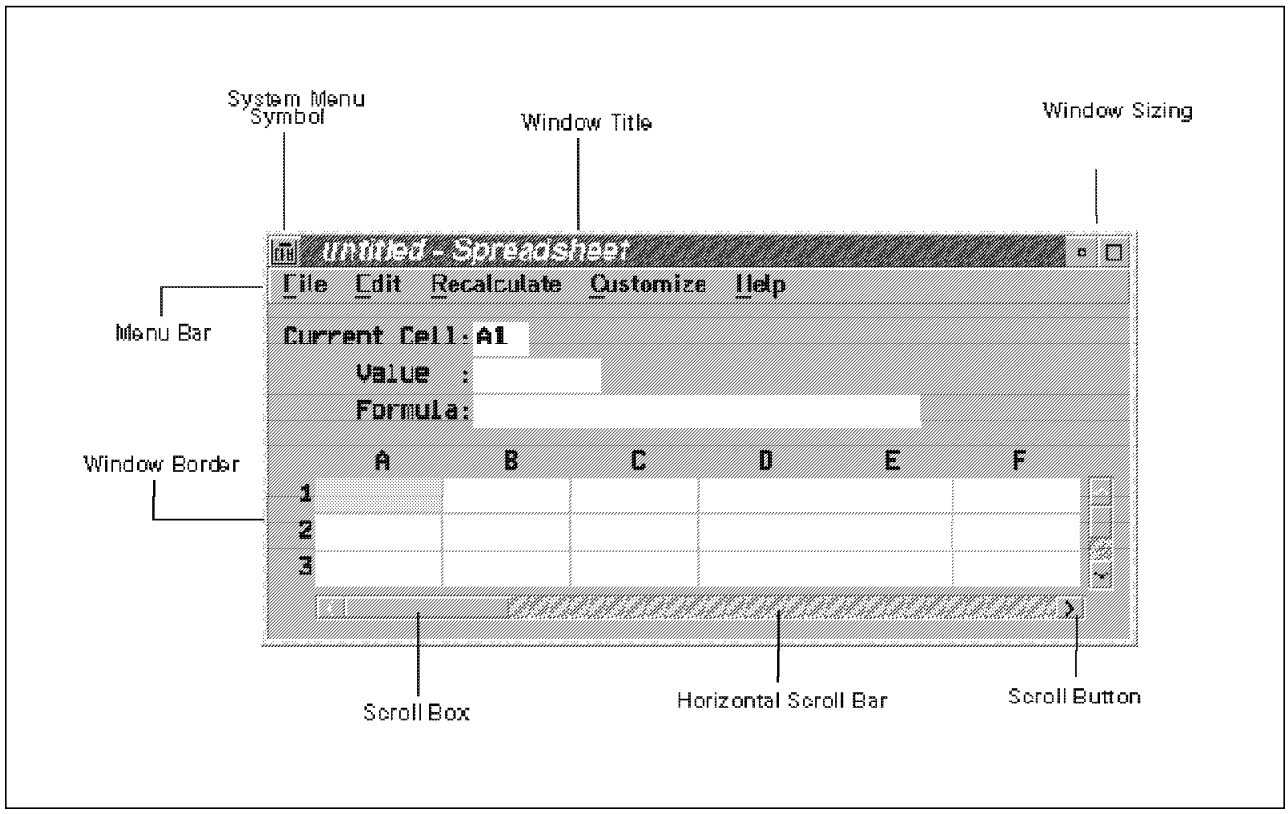


Figure 9. Window and its Components

A secondary window appears when a user needs, or needs to provide, information related to an object in a primary window. For example, a secondary window might contain a message or help information.

The concept of a window is the same in both CUA and Motif. CUA tends to be more specific and Motif more general. CUA window components have an action bar and scroll bar. These are not considered fundamental in Motif.

3.2.6 Icons

An icon is the small graphic image that represents an object. It should convey the information to its corresponding object, and its appearance can change when something about the object changes. For example, the icon of an printer can change to indicate that printer has run out of paper. An icon does not have to be a static image. It can be an animated image or even a video image.

CUA uses the term “icon” to cover the pictorial representation of an object or a selection choice. In CUA, icons can represent objects that users want to work on or actions that users want to perform. A unique icon also represents an application when it is minimized. Motif definition and use of terms relating to icons vary slightly. The Motif icon represents the application or window and not a pictorial representation of an object. This is because Motif basically tries to define Window Manager rather than an object-oriented environment. Motif supports pushbuttons with graphical labels. This results in Motif using buttons in many places where CUA uses the term ″icon″.

3.2.7 Drag and Drop

Drag and drop is the interaction technique that enables direct manipulation of the object. It is called drag and drop because it involves moving an object from one place (dragging) and leaving it at another place (dropping). This technique often involves a source object and target object. A source object is usually the object a user is working with. A target object is the object that a user is transferring information to. For example if a user drags the spreadsheet object to the printer object to print a spreadsheet, then the spreadsheet is the source and the printer is the target object.

The results of drag and drop can change depending on what the source object is. For example, if a user drags a spreadsheet object from one folder object and drops it onto another, the spreadsheet is moved to the target folder. However, if a user drops the same spreadsheet object onto the printer object instead of a folder, the operating environment makes a copy of the spreadsheet and puts it into the printer′s queue to be printed. The original spreadsheet remains at the original location.

3.2.8 Dialog Box

A dialog box is a movable window, fixed in size, that asks the user for information to complete a dialog. The dialog box is always associated with another window. CUA defines two types of dialog boxes : Modal and Modeless.

Modal dialog boxes require the user to complete the dialog before continuing to work on the application. The required input is essential to further processing. Modal dialog boxes are ended by selecting an action that submits the dialog box task for processing. These boxes also contain “Cancel” and “Close.”

Modeless dialog boxes allow the user to continue without providing the required information. He/she can leave the modeless dialog box to work in another window because processing can continue without information. These boxes are used for repeatable actions such as find and change.

Dialog boxes are similar between CUA and Motif. CUA explicitly restricts any sizing of the dialog box, whereas Motif has no specific restriction. Both support the concept of modal and modeless dialog boxes.

3.2.9 Menus

Menu is a mechanism for presenting lists of choices to a user. This has remained one of the most widely used users′ interface with computer applications for a long time. The menus′ look and use both changed dramatically with the concept of GUI. On PWS There are usually four types of menus :

1. Menu bar
2. Pull-down menu
3. Cascaded menu
4. Pop-up menu

Menu Bar: A menu bar appears across the top of most windows, just below the window title. It is a horizontal list of routine choices. When a user selects the choice from the menu bar, an associated pull-down menu is displayed.

Pull-down Menu: The pull-down menu is displayed when a user selects a choice from the Menu bar. It contains choices that are related to one another in some manner. For example, all choices in the pull-down menu could apply to an entire object, a selected object within a window, help information or a view of the object.

Cascaded Menu: A cascaded menu is displayed beside a pull-down menu or a pop-up menu when a user selects a routing choice labelled with the --> symbol. A cascaded menu contains choices that modify or are related to the routing choice. Cascaded menus provide a way for the designer to layer choices so that a user can have access to a wide range of functions without being confused by lengthy lists of choices.

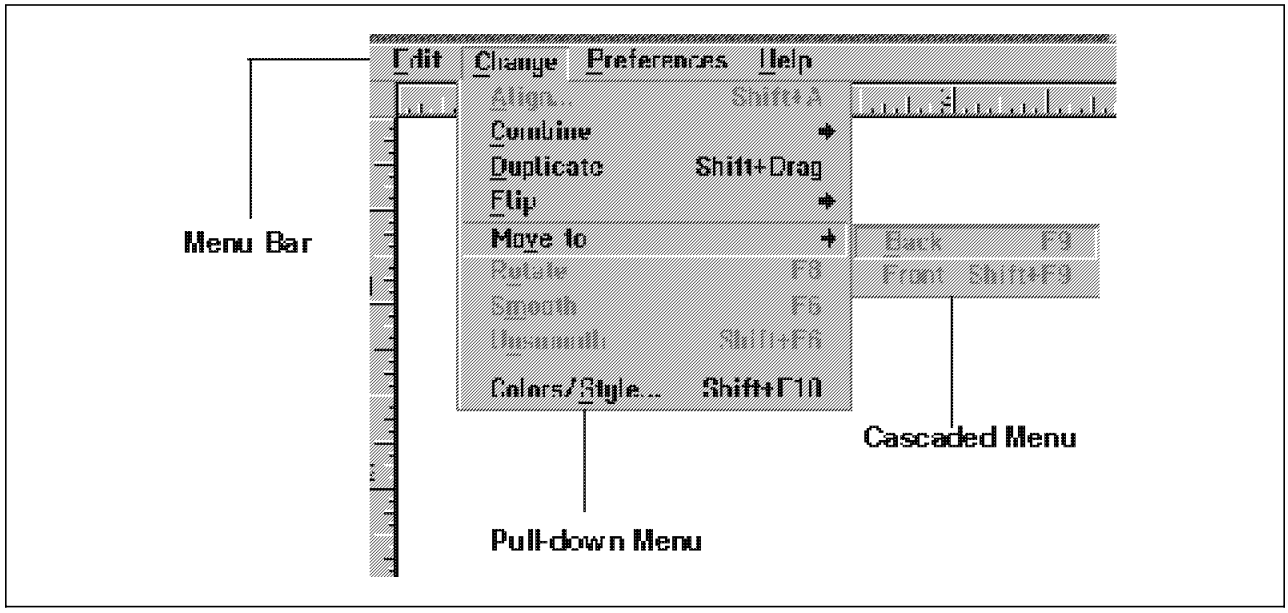


Figure 10. Menu bar, Pull-Down Menu and Cascaded Menu

Pop-up Menu: The pop-up menu contains only those choices that pertain to an object at the time the menu is displayed. It is called the pop-up menu because they appear to “pop up” next to an object when a user presses the appropriate key or mouse button. A pop-up menu is available for each object in the interface. Access to an object’s action by way of a pop-up menu is more direct than access by way of other types of menus mentioned above because a user does not have to select a choice from the menu bar first. Also, a pop-up menu has the advantage of not using up screen space.

The contents of a pop-up menu is based on an object’s context. Variation in an object context leads to variation in its pop-up menu. When a user displays a pop-up menu for a group of objects, the menu contains only those choices that are common to all objects in that group.

3.2.10 Control Elements

The following section describes control elements. They are primarily used in dialog boxes but may be used in any window. They have to be visually unique and present a specific type of choice.

3.2.10.1 Radio Buttons

Radio buttons consist of circles with their associated “choice text.” They provide a single selection field of mutually exclusive choices.

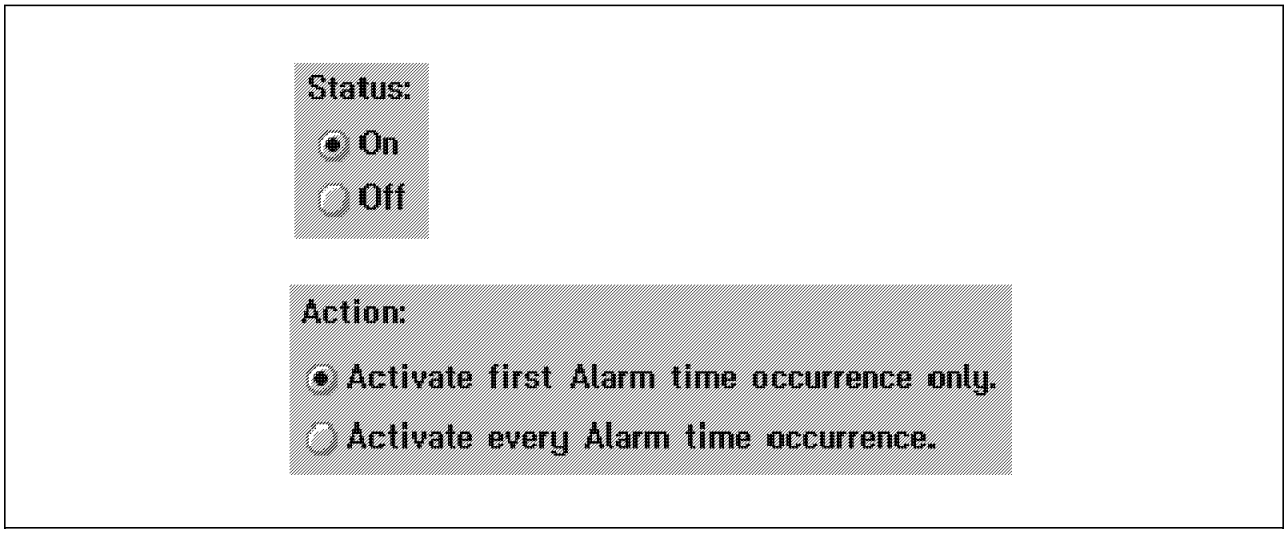


Figure 11. Radio Buttons

3.2.10.2 Check Box

A check box consists of a square box and choice text. Several check boxes can be used together to provide a multiple selection field. Selection are not mutually exclusive but are complimentary. This means that more than one choice can be selected. A selected check box is filled with an “X” as the visual indicator.

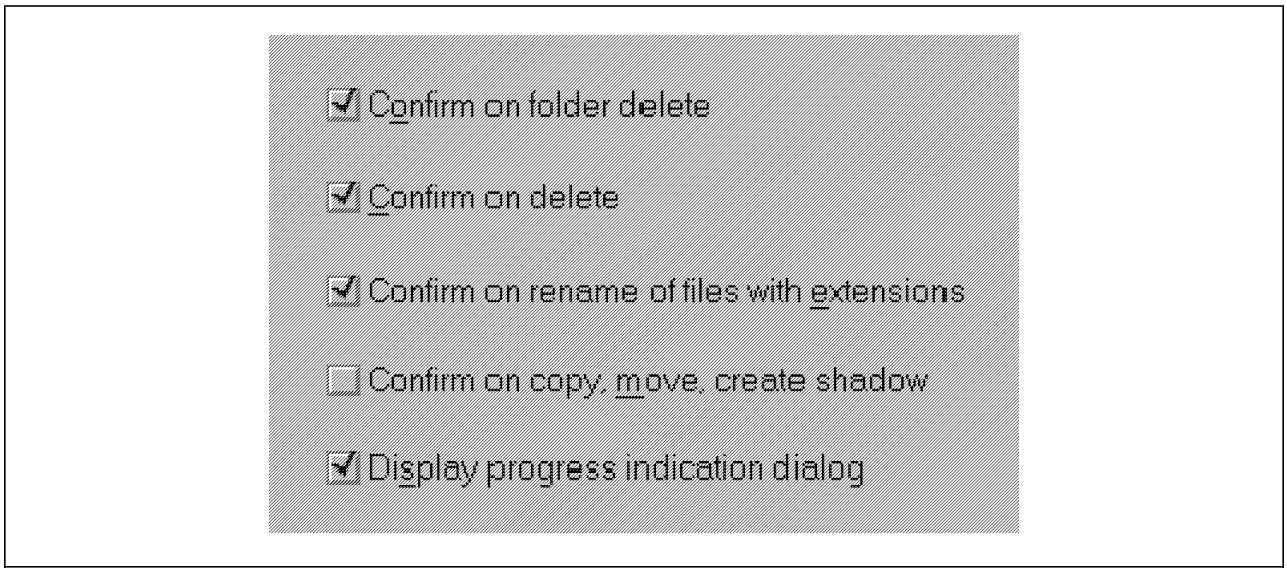


Figure 12. Check Boxes

3.2.10.3 List Box

A list box is used If the list of choices is lengthy and would consume too much space. It can be scrolled both vertically and horizontally. The list box may contain more than one smaller box. For example in a “OPEN” list box there could be a smaller box for files and another smaller box for directories.

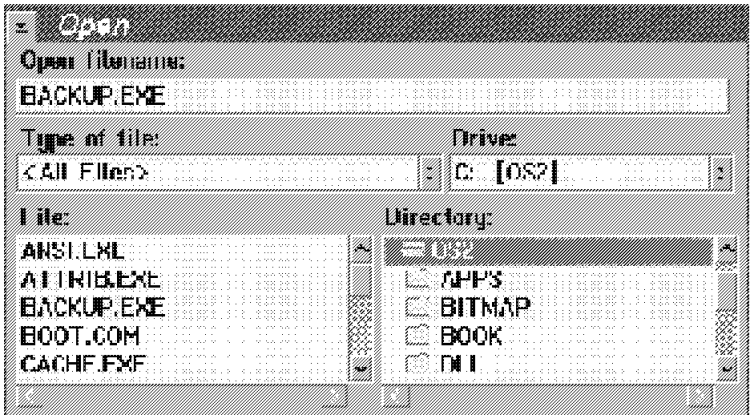


Figure 13. List Box

3.2.10.4 Push Buttons

The push buttons are used in dialog boxes, secondary windows and message boxes. They consist of a command surrounded by a rounded border and are usually used as an alternative to an action bar. Once selected the action occurs immediately. If several pushbuttons appear in the same window one of them should be the default action which is indicated by a bold border.

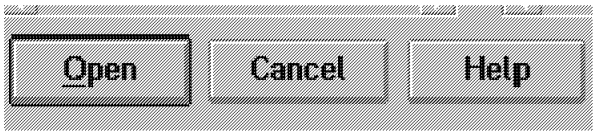


Figure 14. Push Buttons

3.2.10.5 Combination Box

A combination box is a combination of an entry field with a list box. The list box contains a list of possible choices available to the entry field. This entry field collects information from the user. The user points to an item in the list box to place that item in the entry field.

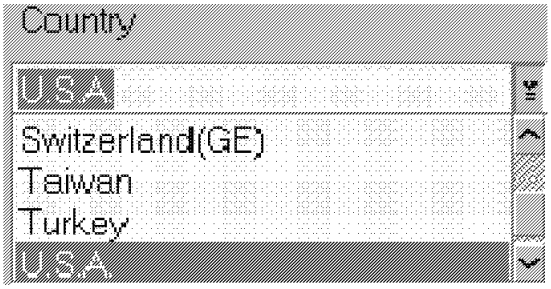


Figure 15. Combination Box

3.2.10.6 Drop-down Combination Box

A drop-down combination box is a variation of a combination box where the list box is hidden until the user requests it. A prompt-box with a downward pointing arrow appears to the right of the entry field to indicate the presence of the hidden list box.

3.2.10.7 Spin Button

Spin buttons are used to select a desired value from the list by scrolling through the consecutively ordered choices. The values in a spin button are displayed as if they are arranged in a ring. When a user presses the down arrow, the value displayed increases. And when a user presses the up arrow, the value displayed decreases.

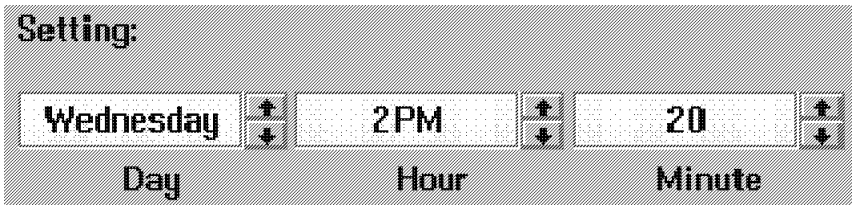


Figure 16. Spin Buttons

3.2.10.8 Progress Indicators

The progress indicator provides feedback to the user during longer processes. It is a visible cue that indicates progress toward the completion of a process, for example copying, formatting, etc. A progress indicator could consist of a digital clock and display the time remaining in a process. It can also be a slider that fills gradually as process continues. When the process is complete the slider is completely filled. A progress indicator can appear in its own window or in the window of the object that is undergoing the process.

3.2.10.9 Value Set

A value set is a single selection field, similar to a radio button but without the choice text. It is used to present mutually exclusive choices. The choices are arranged in a matrix. The value sets are useful for creating palettes of tools.

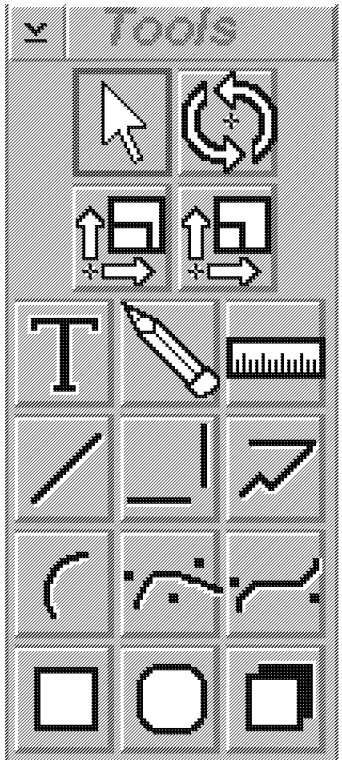


Figure 17. Value Set

The controls are quite similar between CUA and OSF/Motif specifications with some variations. Motif defines a “scale” control that is similar to CUA “progress indicator.” The CUA value set control is not specifically defined in the OSF/Motif Style Guide but the subarea control panel would be a place to implement that function. Motif defines the “stepper button” which is the equivalent of CUA′s spin button.

3.2.11 Cues

The system informs users about the exceptional situation through cues. CUA defines three type of cues:

Audible Cues: An audible cue is the sound generated by a user′s computer to draw a user′s attention. A beep is an example of a simple audible cue. If the product can be used with computer hardware that has advanced audio capabilities, a designer can be imaginative in specifying more elaborate audible cues, such as speech synthesis.

Visible Cues: A visible cue is a change in the appearance of a product′s components. For example, if a user places inappropriate type of information in an entry field, the color of the entry field could change to alert the user that the information given is inaccurate.

Textual Cues: When a user needs more information than can be conveyed with the audible or visible cues, a designer can use a textual cue. This can be displayed in a message window.

3.3 Concluding Remarks

The IBM SAA, CUA and OSF/Motif user interface specifications provide similar directions to user interface development between SAA and AIX systems. Enough information is provided both in the CUA Advance Interface Design Guide and OSF/Motif Style Guide that can provide additional direction for development of a complementary user interface to span SAA, AIX and other UNIX systems. This should be possible while maintaining affinity to these environments so that applications that are developed either within customer enterprises or by software vendors can provide smooth, easy to understand interface to end users.

CUA and Motif user interface styles are evolving with experience and technological development. Each revision in both these specifications removes some differences between them. Thus, we see the emergence of common user interface specifications which will mask the underlying operating environment from users.

Chapter 4. Solutions

This chapter briefly introduces some solutions which are available to create, exploit and support todays Graphical User Interface. Before these products are described, the benefit and usage of Application Programming Interfaces are explained.

4.1 Introduction

Getting a computer to do what we want it to do requires a method which tells the computer how to behave. While in the 60′s it was sufficient to hard code a tabulating machine with a switchboard-like panel, today′s application and user front ends need more complex tools and products to develop an application program.

In the 70′s centralized computers were used to accomplish the every day computer work. To create a program the programmer had very rough tools on his/her hand. Normally a programming language with access to a database was used and later on a tool to define screen panels was used. The choice of tools was not overwhelming and more important was the result. Terms like ″ease of use″, ″consistency″ or ″software engineering″ were not known in those days.

This changed dramatically as the first personal computers entered the marketplace in the 80′s. This was the first time a computer on the desktop was able to take over specific tasks from the large computers. The flexibility of a programmable workstation and the availability of computer languages running on a PC encouraged to develop software for PC′s. This resulted in widespread use of PC programs such as spreadsheet and word processor applications.

Until now all the applications came with their own type of user interface. Users found it very difficult to switch between the different applications. This resulted in a higher learning curve and reduced productivity. At the same time software vendors were confronted with new and more complex applications which should run on different platforms. It was now necessary to develop software in a more strict manner and to keep track of different versions running on different platforms.

In March 1987, IBM announced its System Application Architecture (SAA). SAA describes the use of common methods, products and interfaces for IBM′s 3 hardware platforms. In order for applications to conform to SAA they must:

1. Present applications with a common ″look and feel″
2. Use common programming interfaces and languages which makes it easy to port a application from one system platform to another
3. May be created with standard development tools for increasing productivity such as the AS/400 Developer Utilities.

Based on SAA, ISVs and IBM offers development tools for different hardware and the operating system environment. This enables ISVs to create consistent and maintainable products within a reasonable time frame. This chapter describes some of the tools available.

4.2 Overview to Application Programming Interface (API)

The purpose and usage of an API is best illustrated by the example of DOS and OS/2. DOS offers a rich set of interrupt calls (Int XXh) to perform specific functions. To display for instance, a specific character on the screen the Int 21h may be used. The programmer supplies in register AH the function code 02h for ″Display Output″ and in register DL the character itself. The character is displayed after the Int 21h request is issued to DOS. DOS itself has to perform a lot more instructions to perform this function. The main advantage of this type of access is the unified structure calling this service and the ease of use for the programmer.

Unfortunately there are ways to issue interrupt calls in DOS which may crash the complete system. While this may be acceptable in a single task system it is not in a multitasking system such as OS/2. OS/2 has its own scheme and usage of APIs. In OS/2 protect mode, the program pushes parameters onto the stack and then issues a CALL to an OS/2 API which is in fact a call to an external routine. OS/2 has over 700 APIs available which are grouped in functional areas such as:

1. Device handling (Keyboard, Mouse, Video)
2. General functions (Memory, File, Multitasking)
3. Windows (PM user interface)
4. Graphics

The advantage of APIs are that the developed application:

1. Can execute only documented and allowed functions (OS/2 takes care)
2. Is upward compatible to new versions of OS/2
3. Is easy to maintain.

   In fact, not only DOS and OS/2 provides their own APIs. For example, to manipulate host sessions within Communications Manager/2 an API package called High-Level Language Application Programming Interface is available. This API can be called from many languages such as C or even REXX.

4.3 Tools and Products

The PC primarily developed for exclusive use for end user purposes is seen today in nearly every combination with other computers. Also, added functions in the operating systems allow applications to communicate in a different way with the user. The text-based applications are being replaced by graphics-based ones, which makes applications much more usable.

Trying to keep pace as an application developer in this changing environment is very difficult. Therefore, it is very important for software developers to get application development tools that will simplify a developer's job.

There are different ways an application is used on PCs. This requires different tools and techniques for application development:

1. The application is still full-screen and runs in a host emulator window on the PC.
2. The application conforms to CUA 89 and is written for the Entry or Text model.
3. The application runs as a client/server application and utilizes on the PC the Graphical User Interface conforming to CUA 91.
4. The application is written for the exclusive use on a PC and runs, for example, as a OS/2 application.

All of the above usages have their own tools. Some of them are listed in the following section. The list is not to be seen as a recommendation or preference for any of the products. The technical facts are based on the product description available from the vendor.

4.3.1 Host Environment

4.3.1.1 ISPF Version 4 Release 1

ISPF Version 4 Release 1 for MVS includes ISPF Client/Server, which enables any existing application using ISPF display services on the mainframe (customer written, independent software vendor, and IBM applications) to be displayed on OS/2 or Windows workstations. This gives the application the ability to display common user access (CUA) constructs (action bars, multiple windows, push buttons, and check boxes) in a graphical mode. In this way, ISPF Version 4 Release 1 has become an enabler for CUA 89 both at the workstation and on a nonprogrammable terminal (NPT). ISPF applications are displayed in this GUI (Graphical User Interface) window, which allows the application to run on the workstation unchanged. Using ISPF V4 Release 1 for MVS to write GUI applications has all the advantages of host applications including security, central distribution, and maintenance while also having the advantages of the GUI at the workstation. Communication between the mainframe and workstation is through popular products including Transmission Control Protocol/ Internet Protocol (TCP/IP) and advanced program-to-program communication (APPC).

Presently, the VM version of ISPF has support for the CUA 89 Entry and Text models of EUI.

Execution Environment

- MVS
- OS/2 2.x (with ISPF client and server)
- Microsoft Windows 3.1 (with ISPF client/server)

Program Number (Order Number)

- 5655-042

GIM Form Number

- GC34-4439 *ISPF Specifications V4.1 for MVS*

4.3.1.2 Screen Definition Facility II

Screen Definition Facility II (SDF II) is a tool for developing and maintaining screen and printer oriented objects (panels, panel groups, partition sets, AID tables, and operator control tables) interactively for applications developed with or using CICS/BMS assembler macros, definition, GDDM-IMD, CSP/AD, CSP/370AD and CSP/2AD. Objects can be developed online on all display devices supported by ISPF (V2R2 or later), which is the dialog manager for SDF II.

Development Environment

SDF II operates in the following interactive development environments:

Execution Environment

SDF II allows its definitions, to make use in applications running in the following systems:

- CICS (MVS, VM, OS/2)
- ISPF (MVS and VM)
- CSP (AD, 2AD and 370)

Program Number (Order Number)

- 5665-366 (for MVS)
- 5664-307 (for VM)

GIM Form Number

- GH19-6114 *SDF II General Information*

4.3.1.3 CSP/2AD and CSP/2RS

CSP/370AD and CSP/2AD provide application developers with the facilities to quickly learn the development products, develop an application, and maintain development libraries using interactive panels on the host or windows on the workstation.

You can define and test applications on your workstation using CSP/2AD. Applications developed on the workstation can be transferred to the host for generation. Applications and member definitions created on the host can be transferred to the workstation for further definition, testing, or maintenance (to the workstation where CSP/2RS is installed for running in the CICS).

To develop and maintain an application with the Cross System Product set, an application developer does the following:

- Defines the data, maps, and application logic, getting online help when necessary
- Tests the application for errors
- Maintains the application with the provided utilities and facilities

The application developer then generates a COBOL application with the CSP/370AD generation facility. The generated COBOL application is then ready for running in the production environment where CSP/370RS or CSP/2RS is installed.

Development Environment

CSP/2AD requires the following software:

- CSP/370AD for application generation
- MVS/TSO for installing from a MVS host system
- OS/2 1.3 or later release
- OS/2 EE or Communications Manager/2
- OS/2 EE Database Manager or DB2/2

Execution Environment

- CICS OS/2 Version 1.2
- OS/2 1.3 or later release
- OS/2 EE or Communications Manager/2
- OS/2 EE Database Manager or DB2/2
- Micro Focus COBOL Version 3 or later release

Program Number (Order Number)

- 5688-205 (CSP/2AD)
- 5688-195 (CSP/2RS)

GIM Form Number

- GH23-6515 *IBM SAA Cross System Product*

4.3.1.4 IBM SAA AD/Cycle in S/370 Environment

AD/Cycle CODE/370 is an integral component of IBM′s AD/Cycle new framework for high-level language application development and maintenance across all IBM SAA platforms.

IBM SAA AD/Cycle Cooperative Development Environment/370 is an SAA cooperative processing implementation. It provides a language-sensitive editor and an interactive debugger. The editor operates on the OS/2 programmable workstation and interacts cooperatively with the host to provide application development functions. The interactive debugger allows for OS/2 workstation-based cooperative debugging of host applications, and host-based full screen, line mode and batch debugging.

AD/Cycle CODE/370 allows the application developer to migrate from the nonprogrammable terminal environment to the OS/2 workstation environment. With the OS/2 workstation, the OS/2 Presentation Manager graphical user interface is exploited to deliver the SAA CUA architecture.

Execution Environment

CODE/370 runs under the latest releases of these programming systems:

- MVS/SP 3.1, MVS/ESA 4.1
- VM/ESA 1

The IBM OS/2 CODE Workstation Feature requires:

- IBM OS/2 Extended Edition 1.3 or a later release

Program Number (Order Number)

- 5688-194

GIM Form Number

- GC26-4661 *IBM SAA AD/Cycle CODE General Information*

4.3.1.5 IBM SAA AD/Cycle in AS/400 Environment

AD/Cycle CODE/400 provides a cohesive edit, compile and debug facility and a Data Description Specifications Design Utility (DSU). The AD/Cycle CODE/400 tools operate on the IBM Operating System/2 Version 2 (OS/2 2.0) or higher, with a programmable workstation and interacts cooperatively with the host to provide application development functions. It will support Control Language (CL) and will include the CODE Object Lists that provide the interface to Application Development Tools and Application Development Manager/400 for parts, a new enhanced editor and DDS (Data Description Specification) support in DSU for Enhanced Nonprogrammable Terminal User Interface (ENPTUI).

CODE/400 enhances the ability of the application developer to maintain and extend the useful lives of existing applications. The DDS support for Enhanced Nonprogrammable Terminal User Interface (ENPTUI) in DSU allows existing and new applications to be designed to use graphical user interfaces in their applications such as menu bars, pull-downs, single choice selections, etc. The DDS keywords for ENPTUI functions can be incorporated into any application and does not require new hardware. The DSU WYSIWYG interface for creating these new keywords is extremely functional and easy to use.

Execution Environment

CODE/400 2.3 runs under:

- OS/400 2.3

The CODE Workstation Feature requires:

- IBM OS/2 2.0 with
- IBM Extended Services for OS/2 or
- IBM Communications Manager/2 or
- IBM Communications Manager/400

Program Number (Order Number)

- 5738-CD1

GIM Form Number

- GC26-4661 *IBM SAA AD/Cycle CODE General Information*

4.3.1.6 IBM RUMBA/400

RUMBA/400 provides a set of functions (including display, print, and file transfer) that run entirely within a graphical windows environment which allows users the ability to point-and-click for quick and easy access to host resources. Starting an application, moving information, and simultaneously working with multiple applications is easily handled by moving a mouse.

RUMBA/400 is packaged as a separately orderable, separately priced feature of PC Support, and is designed to take advantage of all the capabilities of a windows environment. There are two versions of RUMBA/400. One runs entirely on Microsoft Windows and one runs in OS/2 PM. These two RUMBA/400 versions are contained within the single feature - thus, providing a flexible and inexpensive migration path between the PC operating systems.

Execution Environment

RUMBA/400 runs under:

- OS/400 2.2 or a later release

The PWS-related requirements are:

- IBM OS/2 2.0 or later release
- IBM PC DOS 5.0 or later release
- Microsoft Windows 3.1 or later compatible release

Program Number (Order Number)

- 5738-PC1 Feature 0795

Manual

- SC41-0135 *AS/400 PC Support/400 RUMBA/400 Guide & Reference*

4.3.1.7 IBM ENVY/400

The IBM ENVY/400 licensed program offering is an object-oriented client/server integrated cooperative application development environment. ENVY/400 supports the Microsoft Windows(DOS) environment as well as the OS/2 environment for development. Customers have the flexibility to create applications that are portable between these environments with few, if any, source code changes. ENVY/400 applications are source compatible between the OS/2 and Windows environments.

ENVY/400 provides all the same development features for developing applications from either environment. The notebook and container CUA 91 controls are also provided in both the OS/2 and Windows environments. Using ENVY/400′s collaborative environment, a development team can create applications in a mixed environment of development workstations. The applications thus developed can run on host AS/400 from Digitalk, Inc. for the workstation environment(s) ordered. ENVY/400 applications contain all necessary ENVY/400 run-time support. ENVY/400 applications do not require any run-time license fees on the AS/400 system or on the workstation.

Development Environment

- OS/400 2.2 or a later release
- IBM PC Support/400

The PWS-related requirements are:

- IBM OS/2 2.0 or a later release
- IBM Extended Services for OS/2
- IBM PC Support/400
- or
- IBM DOS 5.0
- Microsoft Windows 3.1 (DOS)
- IBM PC Support/400

Execution Environment

- OS/400 2.2 or a later release
- IBM PC Support/400

The PWS-related requirements are:

- IBM OS/2 2.0 or a later release
- IBM Extended Services for OS/2
- IBM PC Support/400
- or
- IBM DOS 5.0
- Microsoft Windows 3.1 (DOS)
- IBM PC Support/400

Program Number (Order Number)

- 5733-CS6

GIM Form Number

- GG24-4126 *Introduction to ENVY/400*

4.3.2 Client/Server Environment

4.3.2.1 CICS OS/2 Version 2.0

CICS OS/2 Version 2 is available as a single or multiuser license. If CICS OS/2 Version 2 multiuser is licensed then distributed, features may also be licensed for the support of LAN attached clients.

CICS OS/2 Version 2 single-user may operate attached to a mainframe, stand-alone, or attached to a LAN. CICS OS/2 Version 2 single-user is a subset of CICS OS/2 Version 2 multiuser; it supports the same CICS API, and contains all functions except for those required for the support of multiple users, though it supports ASCII terminals with native attachment (up to 8, subject to hardware limitations).

CICS OS/2 Version 2 multiuser contains functions in a distributed feature which may be installed or downloaded into client machines on the LAN. These are provided for OS/2, DOS, and Windows client machines, and are optimized for these environments; they can occupy less than 100K of storage.

In addition to requesting CICS function from the server applications, running in the client machines may interface to other local applications and may run their own user interfaces, including advanced Graphical User Interfaces (GUIs). They will access a server through the LAN.

Execution Environment

CICS OS/2 Version 2.0 multiuser server and single-user will require the following operating system:

- OS/2 2.0 or a later release

CICS OS/2 Version 2.0 multiuser clients will run under any of the following:

- OS/2 2.0 or a later release
- Windows 3.1
- DOS 3.3 or a later release

Program Number (Order Number)

- 5871-AAA Feature 3127 Single-user (US)
- 5871-AAA Feature 3130 Multiuser (US)
- 5648-036 (Single and Multiuser)

GIM Form Number

- GC33-0637 *IBM CICS OS/2 General Information*

4.3.2.2 VisualAge for OS/2 Version 1.0

VisualAge* is an integrated, application development environment designed especially for client-server, mission-critical, line of business applications through visual programming and construction-from component technologies. It provides a series of high productivity, OS/2 based power tools for the development of applications targeting OS/2 execution systems.

There are two base products in the VisualAge family:

1. 'VisualAge' for the individual user
2. 'VisualAge Team' for team development

'VisualAge Team' provides all the functionality of 'VisualAge' plus support for team programming.

VisualAge provides the following functionality:

- Visual programming (construction-from-components) which enables the development of complete applications from pre-existing or custom-built components with little or no knowledge of the underlying language.
- Support for reusing programs developed in C, COBOL or any language which creates DLLs. This capability promotes the reuse of existing code, reducing development cycle time and future maintenance requirements. C support is included in the base VisualAge and VisualAge Team products. COBOL support may be ordered separately.
- Advanced graphical user interface (GUI) capability, including support to implement CUA '91 user interface controls.
- Communications and transaction processing components which provides a diverse menu of protocols with a simplified common access, including TCP/IP, APPC, and CICS OS/2 ECI.
- Database components for interfacing with both IBM and non-IBM databases which provides a menu of databases with a simplified common is provided in the base VisualAge and VisualAge Team products. The additional database support may be ordered separately.

IBM plans to make VisualAge for Windows and VisualAge System Object Model (SOM) support for OS/2 available during the first half of 1994.

Development Environment

- IBM OS/2 2.1

Program Number (Order Number)

- 5871-BBB Feature 7280 VisualAge (US)
- 5871-BBB Feature 7281 VisualAge Team (US)
- 5621-387 VisualAge
- 5621-388 VisualAge Team

GIM Form Number

- G325-0801 *VisualAge Technology Booklet*

4.3.2.3 IBM Application System 3.2

IBM Application System (AS) is the Fourth GL decision support product for VM and MVS environments. The design criteria of AS is to help professionals and managers to meet their information processing needs without having to learn computers. AS provides very easy-to-use interface to its users. The conversational end user interface provided by AS stands out in terms of ease-of-use and acceptability. This interface allows users to work with AS in conversational mode. The user responds to questions asked by AS and based on these responses AS builds applications. For more experienced users AS provides GUI (on PWSs) and also command language interfaces.

AS provides a wide range of facilities which include : Query, Reports, Charts, Data Management, Project Management, Business Planning and Statistics modules. It also provides very powerful set of AS language commands for application development by experienced IS professionals.

AS combined with PAS/2 provides ideal cooperative processing facilities in the client/server environment. The PS/AS running on OS/2 exploits API facilities of OS/2 and enables AS to act as an intelligent server on the mainframe. The client/server module of AS increases the customers choices for the development of client/server applications. These applications are able to take full advantage of GUI workplace models.

AS provides all these facilities and still masks the complexities of computerization from its users.

Execution Environment

- IBM VM/ESA, VM/XA and VM/SP
- IBM MVS/ESA and MVS/XA

To make use of the Client API:

- IBM OS/2 2.0 or a later release
- IBM Communications Manager Client Server/2 Version 1
- IBM VM PWSCS 1.2 (suitable for both VM and MVS)
- IBM Communications Manager/2 1.x or
- IBM Extended Services for OS/2

Program Number (Order Number)

- 5648-018

GIM Form Number

- GH45-5520 *General Information*

4.3.3 Programmable Workstation (PWS)

4.3.3.1 IBM OS/2 Version 2.1

OS/2 (R) Version 2.1 is available as the desktop operating environment of choice. This version of OS/2 is a robust and stable platform for developing and delivering all types of applications: productivity, mission-critical, educational and entertainment.

Highlights:

- OS/2 Version 2.1, with a 32-bit foundation, runs a wide variety of DOS, Windows and OS/2 applications simultaneously in a protected environment.
- OS/2 Version 2.1 is the platform for mission-critical applications, advanced client/server environments, wide area communications support, relational database support and systems management.
- OS/2 Version 2.1 has superior stability and reliability backed by unmatched IBM service and support.
- Implementation of CUA 91 Workplace model such as Workplace shell.

The OS/2 Version 2.1 Workplace Shell sets the standard for ease-of-use in the personal computer industry. The object-oriented approach allows users to manage many types of objects, programs, data files, printers, network servers and drives from a single interface called the desktop. From the desktop, users can directly manipulate objects so that printing, for example, becomes as simple as dragging and dropping a picture (or icon) representing a letter onto an icon representing a printer.

The Workplace Shell works the way users do, and it can look the way users want it to look. More flexible than ever, OS/2 Version 2.1 allows users maximum freedom to customize desktops, including colors, fonts, object locations and many other aspects of appearance.

The Workplace Shell represents the culmination of earlier technologies evolving over time from the command line to graphical icons, to the current object-oriented interface technology which implements the workplace model defined by Common User Access (CUA) 91. OS/2 Version 2.1 is the Systems Application Architecture environment for the workstation. The Workplace Shell makes OS/2 Version 2.1 an excellent desktop operating environment for all users today and advances operating system technology for tomorrow.

Hardware Requirements

- Personal Computer with an 80386SX (recommended 386DX or a later release)
- 4-8MB Memory
- 60MB hard disk (20 - 40MB available for OS/2)

Program Number (Order Number)

- 5871-AAA Feature 4999 IBM Operating System/2 Version 2.1 (US)
- 5604-467 IBM Operating System/2 Version 2.1

Manual

- G326-0169 *OS/2 V2.1 Facts and Features*

4.3.3.2 IBM Personal Application System/2 Version 3

IBM Personal AS/2 Version 3 provides multifunction, data analysis solutions for the business professional in an office environment, fully exploiting the advanced facilities of OS/2 Version 2 (for example, 32 bit workplace shell, drag/drop and DDE). When installed on a PC, IBM Personal AS/2 Version 3 is able to operate across a LAN or WAN if connected. This provides access to a wide range of data on host systems and personal computers, including access to OS/2 Database Manager.

IBM Personal AS/2 Version 3 is fully integrated with OS/2 Version 2, exploiting both 32 bit and the workplace shell. This means that functions such as drag/drop to OS/2 objects such as Mail, Folders, Printers and the Shredder are supported. Use of the clipboard and Dynamic Data Exchange (DDE) links can be established with other products running in the OS/2 environment.

Using IBM Personal AS Builder/2 Version 3, 32 bit applications can be built that integrate with the work-place shell of OS/2 Version 2, enabling the creation of fully integrated customized solutions to business problems by using:

1. The powerful analysis and presentation functions of IBM Personal AS/2 Version 3.
2. The application creation facilities of IBM Personal AS Builder/2 Version 3, which can be used to extend the standard decision support function provided, in order to meet specific customer requirements.

Execution Environment

- IBM OS/2 2.0 (with Service Pak XR06055) or later release
- IBM OS/2 J and OS/2 K are needed for Japanese and Korean versions

Program Number (Order Number)

- 5875-XXX Feature 5401 IBM Personal AS/2 Version 3 (US)
- 5871-BBB Feature 5400 IBM Personal AS Builder/2 Version 3 (US)
- 5622-103 IBM Personal AS/2 Version 3
- 5622-104 IBM Personal AS Builder/2 Version 3

Manual

- GH45-5054 *Personal AS/2 V3: Making more*

4.3.3.3 IBM Communications Manager/2 Version 1.1

Communications Manager/2 Version 1.1 is IBM's premier communications manager for OS/2 workstations. It contains the communication functions from Extended Services for OS/2 and Communications Manager/2 Version 1.0, and includes major enhancements over these products including:

- Conversion to Presentation Manager
- Configuration and installation aids
- Improved 3270 emulation
- Conversion of 5250 emulation to Presentation Manager interface
- Improved problem determination functions
- Addition of ISDN communication services

The Communications Manager/2 3270 and 5250 emulators are being enhanced to further provide a more consistent look and feel with the IBM family of emulators. This cross emulator consistency reduces end user training time and improves the ease with which end users can move between emulators. The end user functions being added are:

- Integrated mouse support
  - Pop-up key pad window
  - Screen hot spots
- Menu bar and pull-down menus
- Automatic font selection option
- Improved keyboard remapping
- File transfer usability improvements

Execution Environment

- IBM OS/2 1.3.1 SE and EE (with CSD 5050, or a later release)
- IBM OS/2 2.0 or a later release

Program Number (Order Number)

- 5622-078
- 5781-AAA Feature 6485 (US)

Manual

- SC31-7007 *CM/2 1.1 Information & Planning Guide*

4.3.3.4 IBM Database 2 OS/2 Version 1

IBM Database 2 OS/2 Version 1 is a relational database management system and a member of the IBM Relational Database family of products, OS/400.

DB2/2 Version 1 supports access to OS/2 database servers from OS/2, DOS and DOS Windows database client workstations. DB2/2 Version 1 is a 32-bit product and includes the functions previously provided in the OS/2 Extended Services (ES) Version 1.0 Database Manager products. DB2/2 Version 1 also includes additional new functions focused on application portability, DB2 compatibility, SQL and industry standards compliance, new connectivity options, integrity enhancements, reliability, availability, systems management and performance.

Application Programming Interfaces (APIs) and tools are provided for programmers and a database command line processor. Query Manager is provided for use by novice and experienced users.

Execution Environment

- IBM OS/2 2.0 (with Service Pak XR06055) or a later release

Program Number (Order Number)

- 5622-044
- 5781-AAA Feature 4864 (US)

Manual

- S62G-3662 *DB2/2 Information & Planning Guide*

4.3.3.5 Smalltalk/V PM

Smalltalk/V PM is a pure, object-oriented, development tool that implements classes for all controls available in OS/2 V2 Workplace Shell. These controls include the new CUA '91 elements, such as Container, Notebook, and Slider. Smalltalk also supports DDE, OS/2 Help Manager, the OS/2 drag/drop APIs, DBCS, and access to DLL-routines (including the APIs available with PM). Application development in Smalltalk/V PM is highly interactive because of the incremental compiler that allows the changes to take effect immediately.

Smalltalk/V PM supports direct access to DLL routines, so the product can be easily extended using the C language, for example. Database Manager support is available from Digitalk as a separate offering.

The run-time environment consists of three base Smalltalk DLL libraries, one executable file, and one or more application DLL files. Each holds binary images of one or more classes. Run time DLL modules are built using the Object Library Builder, which is shipped with the product as a separate DLL.

Development Environment

- OS/2 Version 2 or a later release

Execution Environment

- OS/2 Version 2 or a later release

Vendor

- Digitalk
  Los Angeles, CA 90045

Manuals (provided with product)

- *Tutorial*
- *Programming Reference*
- *Encyclopedia of Classes*

4.3.4 AIX-/X-Window Environment

4.3.4.1 AIXwindows Environment/6000

AIXwindows* Environment/6000 Version 1.2.5 provides a graphical and compatible with the industry-accepted X Window System and the OSF/Motif 1.2.2 graphical user interface, and can interact with other AIX and other equipment manufacturer systems implementing the X Window System and OSF/Motif interfaces. AIXwindows Environment/6000 provides a sophisticated graphical desktop (AIXwindows Desktop) that can be tailored for integrating and launching applications. AIXwindows Environment/6000 provides the facilities to execute and develop X applications, OSF/Motif applications or applications requiring Display.

The graphical user interface included in AIXwindows is based on the OSF/Motif user interface offering and has been enhanced to support internationalized applications that can handle MBCS (Multi-Byte Character Set). AIXwindows will run in the Enhanced X-Windows environment. The AIXwindows user interface is comprised of the AIXwindows run time environment and the AIXwindows application development environment.

The AIXwindows run time environment consists of an OSF/Motif window manager and AIXwindows Desktop, a sophisticated graphical OSF/Motif-based desktop that provides an iconic view of the file system and the ability to shield the user from the low-level complexities of the AIX operating system. AIXwindows Desktop is an enhanced version of Xdesktop Version 3.0. Simple file maintenance functions can be performed on the files via direct manipulation of the icons. The AIXwindows Desktop can be tailored by the user to integrate and launch applications.

AIXwindows Environment/6000 Version 1.2.5 is compatible with both XWindow System, Version 11 Release 4 and Version 11 Release 5 from Massachusetts Institute of Technology (MIT). AIXwindows Environment/6000 can be accessed by the IBM Xstation 120, 130, 140, (LAN).

AIXwindows Environment/6000 Version 1.2.5 is designed to execute on Version 3.2.5 for RISC System/6000.

Execution Environment

- AIX Version 3.2.5 for RISC System/6000

Program Number (Order Number)

- 5601-257

Manual

- GC23-2202 *AIX General Concepts and Procedures for RISC/6000*

4.3.4.2 AIXwindows Environment/ESA Version 1.2

IBM AIXwindows Environment/ESA Version 1 Release 2, is a state-of-the-art graphical user interface that runs on the AIX/ESA* Version 2 Release 2 operating system for the IBM ESA-capable processors. AIXwindows Environment/ESA Version 1.2 includes several new features over Version 1.1 such as Enhanced X-Windows support based on X Window System Version 11 Release 5, enhanced OSF/Motif support based on OSF/Motif Version 1.1.4, scalable fonts and a font server, and internationalization support based on industry standards and defacto standards.

The enhanced X-Windows client function, based on the X-Window System Version 11 Release 4 is designed to increase the usability of the application processing environment by providing an end user graphical interface. Since this user interface is consistent across multiple UNIX-based platforms, a customer can become productive at a rapid pace. The AIXwindows Desktop function represents the computer as an extension of the real world allowing a user an organizational environment that makes it easier to manage work effectively.

Xdesktop represents the computing system as an extension of the real world. It allows manipulation of the data directly with a mouse in ways that will seem familiar and natural. With this function the screen is seen as a desktop, on which are the directories, programs and files to work with. The Xdesktop function gives an organizational environment that makes it easy to manage work efficiently. An experienced UNIX or AIX user will have all of these benefits plus access to a UNIX shell whenever needed.

Execution Environment

- AIX/ESA Version 2.1

Program Number (Order Number)

- 5696-150

Manual

- GC23-3065 *AIXwindows/ESA General Information Guide*

4.3.4.3 AIXwindows Environment for PS/2

AIXwindows Environment for PS/2* Version 1.3, is a graphical user interface environment that provides the ability to develop and run AIXwindows and X-based applications. It contains AIXwindows Desktop, that is an iconic front end to ease productivity.

Improvements in the windowing and Graphical User Interface (GUI) areas are highlighted with the introduction of the X Windowing System V11 R5 from MIT, available in the following products:

1. AIX PS/2 X-Windows Version 1.3
2. AIXwindows Environment for PS/2 Version 1.3
3. OSF's Motif 1.1.3 (which is available in AIXwindows Environment for PS/2 Version 1.3)

Support for the IBM Xstation 120 and Xstation 130 is provided in the

Execution Environment

- AIX PS/2 Operating System Version 1.3

Program Number (Order Number)

- 5765-170

Manual

- SC23-2251 *AIX PS/2 AIXwindows User's Guide*

4.3.4.4 IBM Emulator for the X Window System (X3270)

The IBM 3270 Emulator for the X Window System Release 2 (x3270) introduces new functions and combines multi-platform support in a single x3270 product. x3270 is an enhanced 3270 emulator that provides workstation access to 3270 applications on IBM System/370 and networks using the Transmission Control Protocol/Internet Protocol (TCP/IP).

x3270 offers substantially more function than most 3270 emulators. Existing x3270 functions include graphics support (GDDM), APL2 extended data stream support, user font selection, screen print, screen zoom, cut and paste, mouse support, programmable symbol support, keystroke buffering and full 16 color support. Comprehensive documentation and on-line manual pages are provided to facilitate use.

3270 operates in X Window System environments using the OSF/Motif graphical user interface (GUI) as well as OpenWindows. x3270 presentations can be displayed on all IBM Xstation models.

Execution Environment

- AIX Version 3.2 for RISC System/6000

Program Number (Order Number)

- 5765-011

Manual

- SC23-0579 *3270 Emulator User's Guide*.

4.3.4.5 Wabi 1.1 for AIX

Wabi** 1.1 for AIX is a new feature of the AIXwindows Environment/6000 Version 1.2.5 licensed program product. Wabi 1.1 provides an environment for the execution of many popular Microsoft Windows-based applications on the IBM RISC System/6000 with AIX. Wabi 1.1 for AIX currently certifies support for a specific set of Microsoft Windows-based applications. The certified applications supported by Wabi 1.1 install and operate like they would if installed and operated under Microsoft Windows 3.1 on an IBM or IBM-compatible personal computer.

Wabi 1.1 for AIX is a separately orderable feature of AIXwindows Environment/6000 Version 1.2.5. AIXwindows Environment/6000 Version 1.2.5 is a prerequisite for this product. For the majority of applications currently certified by Wabi 1.1 for AIX, Microsoft Windows is not required to execute Microsoft Windows applications.

Wabi 1.1 can perform as an excellent Windows application server using X terminals to split the workload of the Windows applications between the CPU and graphics intensive work. For graphics intensive applications, the performance of the application is greater than running the application all on the same system. The reason for this is because the X terminal off-loads the graphics rendering of the application from the server system.

Execution Environment

- AIX Version 3.2.5 for RISC System/6000
- AIXwindows Environment/6000 Version 1.2.5

Program Number (Order Number)

- 5601-257 (Europe, Middle Eastern, Africa only)

Manual

- SC23-2643 *Wabi 1.1 for AIX: Users Guide*

Appendix A. Customer Types and Their EUI Requirements

The table 2 included in this appendix summaries the EUI requirements of five primary customer segments. These five customer segments are :

Stable Host: The stable host segment is composed of those customers who run the majority of their applications on the host today, and continue to do so in the near future.

Platform Optimizing: These customers' applications run today primarily on the host, and will continue to do so for at least the next two years, but they are moving parts of or whole applications to the workstation environment.

Downsizing: Downsizing customers run the majority of their applications on the host today but in two years the execution environment for the majority of their applications will have migrated to workstations.

PC/WS LAN Host Server: These customers have primarily workstation applications and are using the host as a server.

PC/WS LAN PC/WS Server: These are PC/WS only customers. They use PC as workstations as well as servers. These are normally new entrant in IT.

Table 2. Customer Types and Their EUI Requirements.

Type of EUI Models	Stable Host	Platform Optimizing	Down-sizing	PC/WS LAN Host Server	PC/WS LAN PC/WS Server
Entry Model NPT based	M	M	M *	-	-
Text Model NPT based	M	M	M *	-	-
Text Model PWS based	O	M	M	O	O
Basic GUI NPT based	O	O	O	-	-
Basic GUI PWS based	O	O	O	O	O
GUI Workplace model	-	M	M	M	M
GUI with Multimedia support	-	O	O	O	O

Note:

Legend:

M = Mandatory
O = Optional
* = During transition phase

Appendix B. Environments / Tools Matrics

The following table provides a brief overview to the tools and programs we mentioned in Chapter 4, “Solutions” on page 37. The table shows the terminal type used as frontend and the operating environment on which this program/tool runs.

Table 3. EUI Tools and Their Use in Different Environments.

Tool	NPT	PWS	Environment for GUI					
			VM	MVS	OS/400	OS/2	AIX	MS-Win.
ISPF 4.1	√	√		√		√•		√•
SDF II	√	√	√	√		√		
CSP/2AD	√	√		√		√		
CICS OS/2		√				√		√
CODE/370		√	√	√		√•		
CODE/400		√			√	√•		
RUMBA/400		√			√	√•		√
ENVY/400		√			√	√		√
VisualAge for OS/2		√				√		
AS 3.2		√	√	√		√•		
OS/2 2.1		√				√		
PAS/2		√				√		
CM/2 1.1		√				√		
DB2/2 1.0		√				√		
Smalltalk/V		√				√		
AIXWindows Environment /6000		√•					√	
AIXWindows Environment /ESA		√					√	
AIXWindows Environment for PS/2		√					√	
x3270		√•					√	
Wabi for AIX		√•					√	

Note:
√ = Supported
• = Client-Part on Programmable Workstation
• = IBM Xstation or RISC/6000 Display

Glossary

Application programming interface (API). The term used to describe the set of functions by which an application program may gain access to operating systems or subsystem services.

Application schematic. The part of an application which deals with process definition and logic.

Audible cue. A sound generated by the computer to draw the user's attention to or provide feedback about an event or state of the computer.

Batch processing. The processing of data or the accomplishment of jobs accumulated in advance without user interaction. In such processing huge data is submitted to the computer usually on a sequential file. The system reads this data serially and processes it.

Cascaded menu. A menu that appears from, and contains choices related to, a cascading choice in another menu. This menu is used to reduce the length of a pull-down menu or pop-up menu.

Client. See client/server technology

Client/server technology. This is a new name given to distributed processing. More than one computer participates in carrying out one task. The basic principle in client/server processing is that the part of the task is performed at the processor where it is most suitable. More than one computer system is connected together and share each other's resources. A job running on one computer (client) requests another connected computer (server) to perform some task for it.

Common user access (CUA). Guidelines for interface between human users and computers through workstation or terminal. CUA is one of the three architectures of SAA. Please see SAA.

Control elements. Visual user-interface components that allow a user to interact with data and computers. Controls are usually identified by text, for example, heading labels in push buttons, field prompts and titles in windows.

Conversational mode. A mode of operation of a computer system or application in which a sequence of alternating entries and responses between a user and system takes place in a manner similar to a dialog between two persons. In this mode the user maintains a train of thoughts while working with the system.

CASE. An development environment consisting of a set of interfacing tools which enables automation of different phases of software development. The tools which help to automate analysis and design phases are called Upper CASE tools. The tools which generate code are called lower CASE tools.

Cursor. A visible indication of the position where user interaction with the keyboard will appear next.

Dialog. The interaction between user and computer.

End user interface (EUI). A set of techniques and mechanisms consisting of hardware and software that a person uses to interact with computer based systems.

Drag and drop. To directly manipulate an object by moving it and placing it somewhere else using a pointing device.

EUI. See End User Interface.

Fastpath. A method of doing something more directly and quickly than the usual way.

Graphical user interface (GUI). A type of user interface that takes advantage of high resolution graphics. In common usage, a graphical user interface includes a combination of graphics, object icons, the use of pointing devices, menus and overlapping windows.

GUI. See graphic user interface.

Icon. A graphical representation of an object, consisting of an image, image background and a label.

Intelligent work station (IWS). A term used for programmable work stations (PWS). See PWS.

Interoperability. The ability to interconnect systems from different architectures, and have them work together to satisfy a business requirement. Some examples of this requirement are: message interchange between systems, sharing of resources, such as data between applications running on different platforms, migration of applications across these platforms and migration of skills.

List box. A control that contains a list of objects or setting choices that a user can select.

Mainframe. A computer, usually in a computer center, with extensive capabilities and resources to which other computers may be connected so that they can share these resources.

Mode. A method of operation in which the actions that are available to the user are determined by the state of the system.

Multitasking. A mode of operation that provides for concurrent performance, or interleaved execution of two or more tasks.

Nonprogrammable terminals (NPT). A user terminal having no processing capability. NPTs are attached to host computer which controls most of the part user interface functions on NPTs.

Object. In EUI, an object is a visual component that a user can work with to perform a task. An object can appear as text or an icon. In Object-oriented technology an object is defined as a software packet containing a collection of data elements and a set of procedures that are only valid operations on that data.

Object oriented technology (OOT). The evolving technology which deals with object orientation. This leads towards software developments through software parts (objects).

Object Class. The categorization or grouping of objects that share similar behaviors and characteristics.

Object hierarchy. A way of illustrating the relationships among objects. Each object that appears in a level below another object is an example of the upper object and inherit the properties of its parent.

Open systems. A system whose characteristics comply with the standards made available throughout the industry and that, therefore, can be connected to other systems complying with the same standards.

OSF/Motif. The graphical user interface combining a toolkit, presentation description, window manager and style guide. It provides guidelines for developing user interface for Unix based systems.

Pointing device. A device, such as a mouse, trackball or joystick used to move a pointer on the screen.

Progress indicator. The control used to inform a user about the progress of a process.

Programmable workstation (PWS). A workstation that has processing capability and that allows a user to change its functions.

Relational database management systems (RDBMS). The data management system which manages relational data. The relational data is organized in two dimensional tables and accessed according to the relationships between data elements. SQL is normally used for manipulating relational data.

Scroll bar. A window component that shows a user that more information is available in a particular direction and can be scrolled into view. Scroll bars can be either horizontal or vertical.

Scroll box. The part of a scroll bar that indicates the position of the visible information relative to the total amount of information available in a window. A user clicks on the scroll box with a pointing device and manipulates it to see information that is not currently visible.

Server. See Client/server Technology

Shortcut key. A key or combination of keys assigned to a menu choice, even if the associated menu is not currently displayed. This is used by more experienced users to gain fastpath access.

Spin button. A control used to display, in sequence, a ring of related but mutually exclusive choices. It contains a field that can accept the user′s input, which allows a user to make a selection by typing a valid choice, or displays a field with a value that the user can merely accept. The user can change the value by spinning through the ring of choices.

System application architecture (SAA). SAA is the detailed architecture (specifications) about software interfaces, conventions and protocols that programmers use to create common applications. SAA specifications provide a structure that enables consistent, transparent access to information resources across IBM operating environments i.e. OS/2, OS/400, VM, MVS and any other which adhere to these specifications.

User interface. See End user interface (EUI)

User interface management systems (UIMS). This helps developers to create and manage all aspects of user interfaces in such a way that they do not have to code details of these aspects in application.

Value set. A control that allows the a user to select one choice from a group of mutually exclusive choices. A value set is used primarily for graphic choices.

Window. An area with visible boundaries that presents a view of an object or with which a user conducts a dialog with a computer system.

Workplace. A container that fills the entire screen and holds all of the objects that make up the user interface.

List of Abbreviations

AIX	Advanced Interactive Executive
API	Application Programming Interface
APPC	Advanced Program-to-Program communication
AS	Application System
ASCII	American National Standard Code for Information Interchange
BMS	Basic Mapping Support
CASE	Computer Aided Software Engineering
CCS	Common Communication Support
CPI	Common Programming Interface
CICS	Customer Information Control System
COBOL	Common Oriented Business Language
CODE	Cooperative Development Environment
CPU	Central Processing Unit
CSD	Corrective Service Delivery
CSP	Cross System Product
CUA	Common User Access
DB2	Database 2
DBCS	Double Byte Character Set
DDE	Dynamic Data Exchange
DDS	Data Description Specification
DLL	Dynamic Link Library
DOS	Disk Operating System
DSU	Data Description Specifications Design Utility
HLLAPI	High-Level Language Application Programming Interface
ENPTUI	Enhanced Nonprogrammable Terminal User Interface
ESA	Enterprise Systems Architecture
EUI	End User Interface
GDDM	Graphic Data Display Manager
GUI	Graphical User Interface
IBM	International Business Machines Corporation
IEEE	Institute of Electrical and Electronic Engineers
IPF	Interactive Productivity Facility
ISDN	Integrated Services Digital Network
ISPF	Interactive System Productivity Facility
ISV	Independant Software Vendor
ITSO	International Technical Support Organization
IWS	Intelligent Work Stations
JCL	Job Control Language
LAN	Local Area Network
MBCS	Multibyte Character Set
MIT	Massachusetts Institute of Technology
MVS	Multiple Virtual Storage
NPT	Nonprogrammable Terminal
OOP	Object Oriented Programming
OOUI	Object Oriented User Interface
OS	Operating System
OSF	Open Software Foundation (Inc.)
PAS	Personal Application System
PC	Personal Computer
PM	Presentation Manager
POWER	Performance Optimization With Enhanced RISC
PDF	Program Development Facility
PWS	Programmable Workstation
QBE	Query By Example
QMF	Query Management Facility
RDBMS	Relational Database Management System
REXX	Restructured EXtended eXecutor Language
RISC	Reduced Instruction Set Computer
SAA	System Application Architecture
SDA	Screen Design Aid
SDF	Screen Designing Facility
SOM	System Object Model
SQL	Structured Query Language
TCP/IP	Transmission Control Protocol/Internet Protocol
UIMS	User Interface Management System

VM	Virtual Machine
VSE	Virtual Storage Extended
WYSIWYG	″What You See Is What You Get″

Index

X

ITSO Technical Bulletin Evaluation RED000

The Library for System Solutions
End User Interface Reference

Publication No. GG24-4107-00

Your feedback is very important to help us maintain the quality of ITSO Bulletins. **Please fill out this questionnaire and return it using one of the following methods:**

- Mail it to the address on the back (postage paid in U.S. only)
- Give it to an IBM marketing representative for mailing
- Fax it to: Your International Access Code + 1 914 432 8246
- Send a note to REDBOOK@VNET.IBM.COM

Please rate on a scale of 1 to 5 the subjects below.
(1 = very good, 2 = good, 3 = average, 4 = poor, 5 = very poor)

Overall Satisfaction ____

Organization of the book	____	Grammar/punctuation/spelling	____
Accuracy of the information	____	Ease of reading and understanding	____
Relevance of the information	____	Ease of finding information	____
Completeness of the information	____	Level of technical detail	____
Value of illustrations	____	Print quality	____

Please answer the following questions:

a) If you are an employee of IBM or its subsidiaries:

Do you provide billable services for 20% or more of your time? Yes____ No____

Are you in a Services Organization? Yes____ No____

b) Are you working in the USA? Yes____ No____

c) Was the Bulletin published in time for your needs? Yes____ No____

d) Did this Bulletin meet your needs? Yes____ No____

If no, please explain:

_____

_____

What other topics would you like to see in this Bulletin?

_____

_____

What other Technical Bulletins would you like to see published?

_____

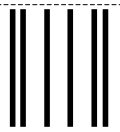
Comments/Suggestions: **(THANK YOU FOR YOUR FEEDBACK!)**

Name _____ Address _____

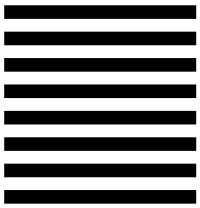
Company or Organization _____

Phone No. _____

Fold and Tape **Please do not staple** Fold and Tape



NO POSTAGE
NECESSARY
IF MAILED IN THE
UNITED STATES



BUSINESS REPLY MAIL

FIRST-CLASS MAIL PERMIT NO. 40 ARMONK, NEW YORK

POSTAGE WILL BE PAID BY ADDRESSEE

IBM International Technical Support Organization
Department 91J, Building 235-2
Internal Zip 4423
901 NORTHWEST 51ST STREET
BOCA RATON FL
USA 33431-1328

Fold and Tape **Please do not staple** Fold and Tape

Printed in U.S.A.