The evolution of the Common User Access Workplace Model

by R. E. Berry
C. J. Reeves

This paper describes some of the influences contributing to and issues in dealing with the evolution of user interface guidelines over time. In particular, we focus on the evolution of IBM's user interface architecture, the Common User Access™ (CUA™) interface, over a period of six years. Discussed are the key architectural and design elements of the CUA Workplace Model, the fundamental shifts in computer-human interaction that have occurred since the first publication of the guidelines in 1987, and how user interface design, operating systems, and tools have interacted in the evolution of the guidelines.

The information should help designers of user interfaces and developers of user interface guidelines to appreciate some of the factors involved in the long-term evolution of a user interface style. The paper provides an introduction to the most recent evolutionary step in the CUA style (the Workplace Model) to help the reader place these factors in perspective relative to the degree of evolutionary change.

User interface guidelines are intended to help product designers and developers create a user interface that users will find easy to learn and use. The *user interface* is the means by which users and computers communicate with each other. It supports a dialog, much like a conversation between people, but this dialog occurs between a user and a computer.

The Common User Access* (CUA*) interface guidelines are based on sound user interface design principles and object-oriented relationships. They specify common user interface components and techniques, and guidelines for applying them.

The CUA interface guidelines are general guidelines that are intended to apply to aspects that are common across many products. However, applying these guidelines alone is not enough. Many aspects of a user interface for a product pertain to specific product functions and are not addressed by generalized user interface (UI) design guidelines. These are considered *product-specific* aspects of the user interface. Two important points to remember regarding product-specific aspects are:

- Generalized guidelines, such as CUA guidelines, should not limit product design creativity and ingenuity for product-specific design issues. For example, IBM's CUA guidelines[1] do not provide direction about how an accounting product should implement a balance sheet or how formulas are handled in a spreadsheet.
- To make good decisions for product-specific design issues, designers need to understand user interface design principles, models, and methods, in addition to applying the CUA guidelines. The user interface design principles, models, and methods described in IBM CUA publications[1,2] are

generally applicable to product-specific interface aspects as well.

Evolution of the CUA guidelines

Three publications containing CUA guidelines have appeared in 1987, 1989, and 1991, respectively. The guidelines have changed in response to two related factors: the fast-growing technology of the personal computer, and increasing demand from users that the computer match their way of thinking, rather than the other way around.

The CUA interface has always emphasized the user's needs, but with different assumptions about the technology that would meet them. The guidelines appearing in 1987 (CUA87), for example, assumed a world of personal computers intermixed with host-attached nonprogrammable terminals, like the IBM 3270 Information Display System. CUA87 had a goal of consistency and transfer of a user's knowledge between those systems. But personal computer technology and capabilities advanced rapidly in the 1980s and the gap between how a user could interact with a terminal and what was possible using a personal computer began to widen significantly.

The CUA interface was updated in 1989 (CUA89) to separate and focus on the guidelines that are unique to the needs of a personal computer user. CUA89 applied the principles inherent in CUA87 to a personal computer environment that provided a rich set of user interface mechanisms, like sizable and movable windows, standard menus, user interface controls, and dialogs. Equally as important, the personal computer operating system, exemplified by OS/2* and the Presentation Manager*, offered a graphical view of available programs and data, and the ability to run multiple applications concurrently. The user was no longer constrained to working with one application at a time, within the limitations of character-based presentation. Suddenly all of the resources of a powerful system were available to users.

The 1991 guidelines (CUA91) built on this advancement, with tools and techniques that moved the interface closer to the way users accomplish work in the real world. Together, the CUA interface and the OS/2 Workplace Shell* redefine data and application programs 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 users' tasks.

CUA's evolution since 1987 marks a growing ability for the adaption of computers to a user's world and the extension of that world. From a model of interaction in which a user takes actions to accomplish a single task, in a sequence strictly controlled by the computer application, the user is now able to take on a wide variety of tasks and move smoothly between them at a comfortable pace. Contrary to a view of an application program as the dominant and controlling factor in the interface, the user is in control and deals with objects that relate to the user's view of the world. Figure 1 summarizes CUA's change in focus over time.

The CUA interface in 1987

In 1987, CUA was influenced by the goals of consistency and transfer of knowledge between nonprogrammable terminals (NPTs) like the IBM 3270 Information Display System and personal computers. These goals were difficult to maintain as the capabilities of the two environments diverged.

An NPT device is usually connected to a host mainframe processor shared by many other users. A goal of application design in this environment is to optimize the use of the connection, and to minimize the number of interactions with the host. For example, a user fills in a form, selects an action from a menu, or enters a command without interacting with the application program. When the user input is complete it is submitted to the computer for validation and processing. The practical limitations of the terminal environment limit the bandwidth of communication between the user and the computer. This results in a polarization of interface usefulness: simplicity with fixed and inflexible access for the novice, and arcane but powerful command languages for the expert.

The personal computer (or PC) offers two fundamental advantages for users: the PC can detect and provide feedback to a user's actions immediately; and it can keep data and applications that are private and controlled by the user. The ability to provide immediate feedback offers a far higher rate of interaction between user and machine: results can be shown and refined immediately according to the user's actions; and the higher feedback rate supports devices and techniques that make the interface more natural and intuitive, like the point-and-select techniques provided using a mouse. Standard mechanisms like menu bars, push buttons, and dialog boxes are implemented in PC systems like OS/2. CUA87 also prescribes their use in the NPT environment, but without instant feedback, the advantages are diminished considerably.

The OS/2 object-oriented Workplace Shell has evolved from the Graphical Model and the Workplace Model.

The personal nature of the PC environment, including the skill, training, and motivation levels of typical users, are additional reasons for the divergence between the PC and NPT interface styles. With a PC, the user has open access to the data and programs on the local system, and maybe even those shared on a network. As a result, the challenge in the interface is not simply how to interact with an application, but how to find the applications that are available, how to start and stop them, and how to locate and access data. To a great degree users in the PC environment expect to be self-sufficient and not to rely on operational support that is typically available in NPT environments.

These needs contributed to the development of a graphical shell for OS/2. The shell offers a pictorial view of applications and data files, standard ways to start and stop applications in windows, ways to find and access data files, and ways to move between the application windows.

CUA87 established a user interface architecture built on sound principles of user interface design, and it established IBM as a participant in the interface design field. However, the CUA87 guidelines were quickly outdated by rapidly emerging PC capabilities. The goals of consistency and transfer between the NPT and PC environments tended to inhibit the designers of PC applications, and raised a need for new development tools and techniques for application development.

The primary tool that implemented PC-terminal consistency was the OS/2 Dialog Manager (DM), which became available in 1989. This combination of rapidly evolving PC capabilities and a shortage of development tools led to mixed acceptance of CUA87. However, CUA87 established foundations that would lead to two significant benefits in the future: the CUA87 architecture and design principles provided a sound basis for converging IBM's PC user interface with Microsoft Corporation's early work on Microsoft Windows**, contributing to designs of the Presentation Manager and OS/2 Workplace Shell; and the limited success of CUA87 underscored the importance of providing tools to support application creation and execution. Figure 2 shows an example of the CUA87 style.

The CUA interface in 1989

In 1989, IBM placed the CUA personal computer and nonprogrammable terminal interfaces on clearly separate paths, and synchronized delivery of the CUA guidelines publication with enabling components, such as OS/2 Version 1.2, with its graphical shell, and the EASEL** application development tool. The CUA89 interface established an application-oriented *Graphical Model* as the primary style for the PC environment. It also introduced an object-oriented *Workplace Model*, which extended the Graphical Model with notions of hiding computer concepts, focusing users on their objects and tasks, enhancing user control, and providing richer interaction with the computer.

In comparison, the CUA87 guidelines had as a goal the consistency between NPTs and PCs, while the CUA89 guidelines focused on consistency between applications within the PC environment, and optimizations made possible through PC-unique capabilities. The guidelines dealt with the issues of locating applications and data, starting and stopping applications, interacting with applications in standard ways, and manipulating application windows. The CUA Graphical Model was so called because it exploited the capability of the PC to display graphical rather than character-based information. It offered a pictorial view of the system and was supported by a series of ready-to-use user interface components supplied by the OS/2 Presentation Manager. These components, called *controls*, automated and made consistent most of the common techniques for interaction with the system, such as entering and editing text, selecting from lists, setting options, and interacting with windows and dialog boxes.

The CUA89 interface became a de facto standard for PC user interfaces largely because of the convergence of five factors:

- It was endorsed by both IBM and the Microsoft Corporation.
- It was supported by standard components (controls) in both OS/2 Presentation Manager and Microsoft's Windows for the Disk Operating System (DOS).
- It was exemplified by the two companies' respective system shells and a series of popular applications like Word**, PageMaker**, Designer**, and Excel**.
- Application development tools such as EASEL, which facilitated implementation, began to appear.
- Use of the guidelines was encouraged as an open standard.

The Open Software Foundation, Inc. (OSF) adopted CUA as the model for its OSF/Motif** standard for UNIX** systems, and leading vendors with divergent interface styles, like Lotus Development Corporation and WordPerfect Corporation, produced CUA-oriented versions of their key products, like the Lotus 1-2-3** spreadsheet, and WordPerfect** word processing package.

The CUA89 interface was initially realized in the OS/2 Version 1.2 graphical shell and in personal productivity tools, such as spreadsheets, graphics programs, and word processors. In addition, the IBM Systems Application Architecture* strategy called for the industry to adopt the interface for business applications developed by in-house information system organizations and third-party software vendors. However, programming to the OS/2 Presentation Manager was complex, time-consuming, and required specialized skills. Application development tools, such as EASEL, were key to the creation of business applications. For example, EASEL had been originally developed as a means to put a more attractive PC interface on existing host-based terminal applications. In this timeframe it was extended with facilities to produce new applications that embodied the key elements of the CUA89 interface. Available for both Windows and OS/2 Presentation Manager, EASEL moved acceptance of the CUA interface into the information systems department, IBM's traditional customer. Figure 3 shows an example of the CUA89 OS/2 Version 1.3 desktop.

The CUA89 graphical model mapped the key architectural elements and principles on which the CUA87 interface had been based into OS/2, its tools, and applications. It did not significantly advance the concepts behind nor the fundamental style of the interface. The CUA89 publication also included a brief description of a user interface style that originated in IBM's OfficeVision*/2 composite applications. This style, which evolved to the CUA Workplace Model, extended the Graphical Model style with the notions of object orientation, consistency across different types of objects, and richer user interaction with the computer. This interaction was characterized by a transformation of the user's view of the system from one of applications and data to one of familiar objects, such as telephones, calendars, memos, and mail baskets. This Workplace Model has been implemented in OS/2 Version 2.0 and its Workplace Shell, and is the basis for the CUA91 guidelines, published in September 1991.

The CUA interface in 1991

The CUA91 interface, or *Workplace Model*, hides computer-based concepts, focuses users on their own objects and tasks, and increases the user's control of the computer-human interface. It provides the user with access to information in context, that is, in the way the user wishes it rather than in some limited application-defined form. The Workplace Model places an emphasis on direct manipulation of objects and making the computer transparent to the user's tasks. Norman states:

> When I use a direct manipulation system—whether for text editing, drawing pictures or playing games—I do think of myself not as using a computer but as doing a particular task. The computer is, in effect, invisible.[3]

The CUA91 guidelines emphasize this perspective in application design.

The CUA91 interface has evolved using new tools and extensions to the capabilities of the underlying operating system. The extensions include drag and drop support, container and notebook objects, sliders, and other new user interface controls. OS/2 Version 2.0 has a new shell, called the *Workplace Shell*, which converges the previously separate file manager, desktop manager, and desktop into a single user concept: the *Workplace*. The Workplace Shell user is free to organize work in any way that is convenient, and to use direct manipulation to perform common tasks, such as moving, copying, printing, mailing, and establishing relationships between objects.

Tools to implement the interface are increasingly object-oriented. As noted in Graham's recent book:

> Most of the current GUIs [graphical user interfaces] just could not have been written without the use of object-oriented techniques . . . The API [application programming interface] in such systems is huge and the event-driven style of user interaction makes programming doubly difficult . . . there is a profound need for object-oriented solutions to the problems of the construction, maintenance and use of GUIs.[4]

The OS/2 Presentation Manager embodies some object-oriented concepts, and the IBM Systems Application Architecture strategy has extended that capability to a broader audience of developers by adding tools such as Smalltalk V/PM** for OS/2 to the strategic Systems Application Architecture toolkit for AD/Cycle*. In the case of Smalltalk V/PM, it provides a basis for additional tools that will extend to end users the ability to create advanced graphical user interfaces. Figure 4 shows an example of the CUA91 user interface as implemented in the OS/2 Workplace Shell.

The CUA today

The evolution of graphical user interfaces has had profound effects on the usability of personal computer systems, and the evolution must continue.

The evolution to more user-driven interfaces means that more attention must be given to understanding users and their tasks, to creating user interface architectures that match users' expectations, and to providing application design and enabling tools that facilitate implementation of the interface model. The remainder of this paper and the paper detailed in Reference 5 expand on these aspects.

What is the CUA interface?

The CUA interface today is described by two models, the Graphical Model, which specifies visual representations and interaction techniques, and the Workplace Model, an object-oriented extension that specifies a standard environment for user interface objects. The CUA interface is fully described in two publications (see References 1 and 2).

CUA Graphical and Workplace Models. As new and better methods of interacting with computers have been developed, user interfaces have evolved to take advantage of those methods. Most of the currently popular interfaces have reached a plateau characterized by a graphical user interface or GUI. The CUA interface defines two levels of GUI. The CUA Workplace Model is the latest example of that evolutionary process.

The Graphical Model (as defined by CUA89) defines a graphical user interface in which the user starts application programs and then uses application-provided facilities, such as menus and File Open windows, to find and open data files used to accomplish the user's task. This model allows users to take advantage of operating systems having multitasking capabilities by letting them run several applications concurrently. Applications are initially represented on the display screen as small graphic images called *icons.* When a user starts an application, a portion of the screen called a window is opened to give the user access to the functions in the application. Many windows can be open on the screen simultaneously and a user can have many applications running at the same time. The Graphical Model is called an *application-oriented model* because the user's focus is on finding and starting the appropriate application program first, then finding and opening data. Figure 3 shows an example of the Graphical Model user interface, as implemented in OS/2 Version 1.3.

The Workplace Model is an object-oriented extension of the Graphical Model. It also allows users to take advantage of multitasking capabilities. The Workplace Model, however, is an *object-oriented user interface*, which means that the

Figure 1 [illegible] consistency

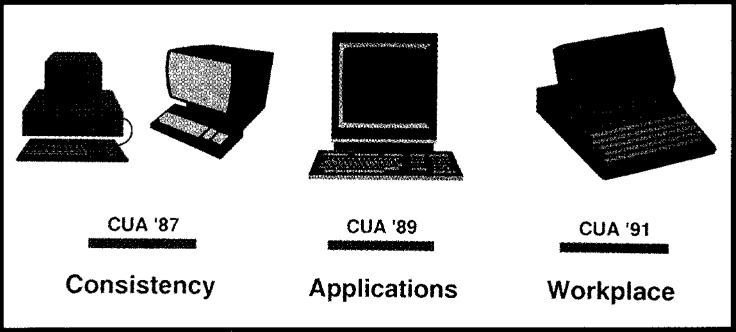
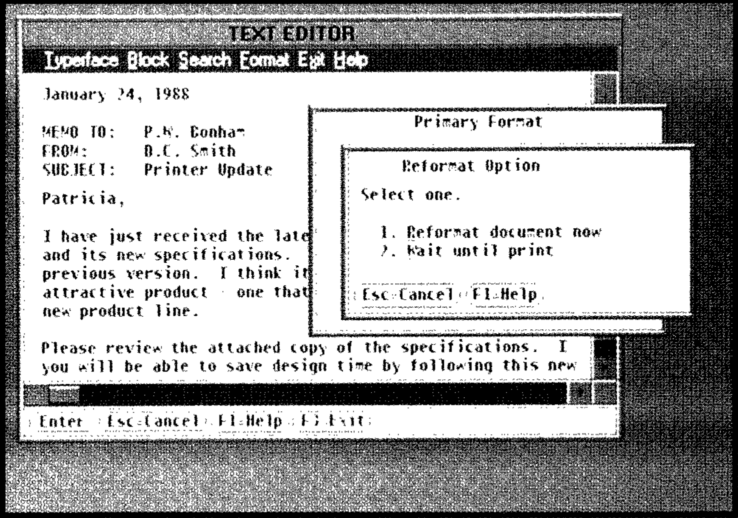


Figure 2 CUA '87: a focus on consistency

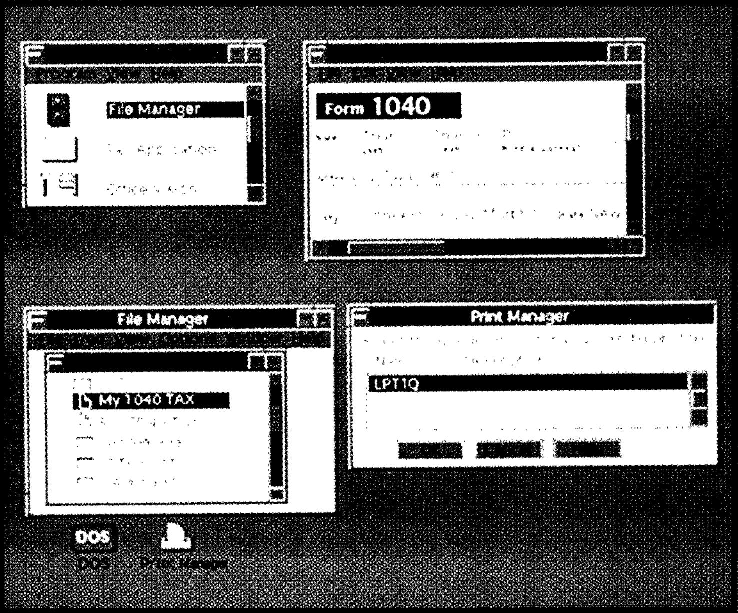


```
January 24, 1988

MEMO TO:   P.W. Bonham
FROM:      B.C. Smith
SUBJECT:   Printer Update

Patricia,

I have just received the late
and its new specifications.
previous version.  I think it
attractive product - one that
new product line.

Please review the attached copy of the specifications.  I
you will be able to save design time by following this new
```

```
Primary Format

Reformat Option

Select one.

  1. Reformat document now
  2. Wait until print

Esc=Cancel  F1=Help
```

```
Enter  Esc=Cancel  F1=Help  F3=Exit
```

Figure 3 CUA '89: the OS/2 Program Manager and File Manager—a focus on applications and data



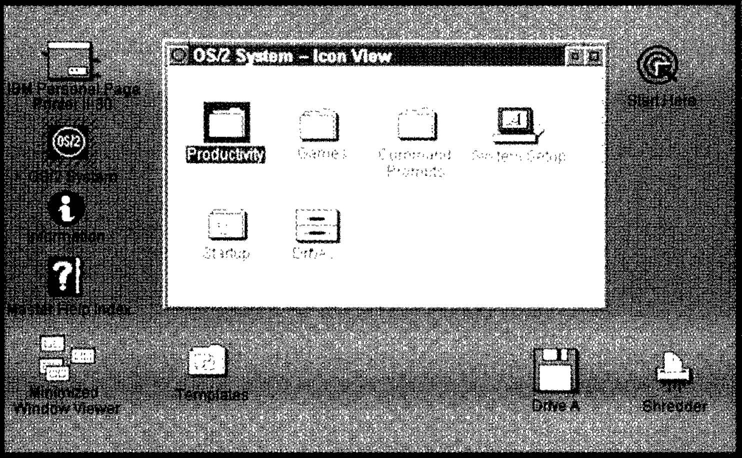


Figure 4 [illegible] objects in the Work- [illegible]

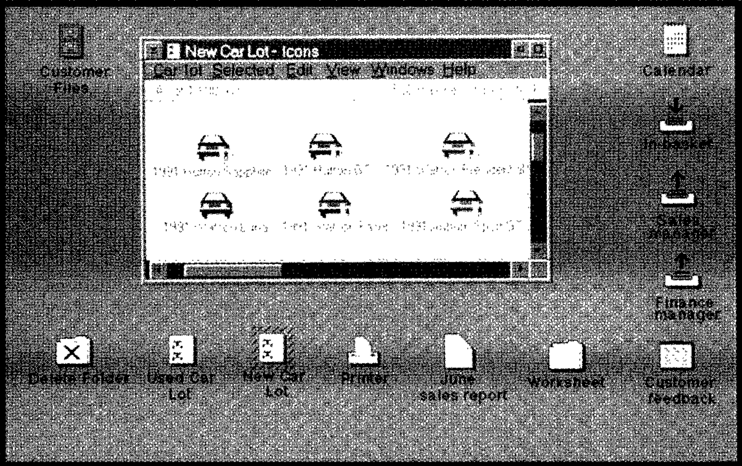


Figure 5 An application using the CUA Workplace [illegible]

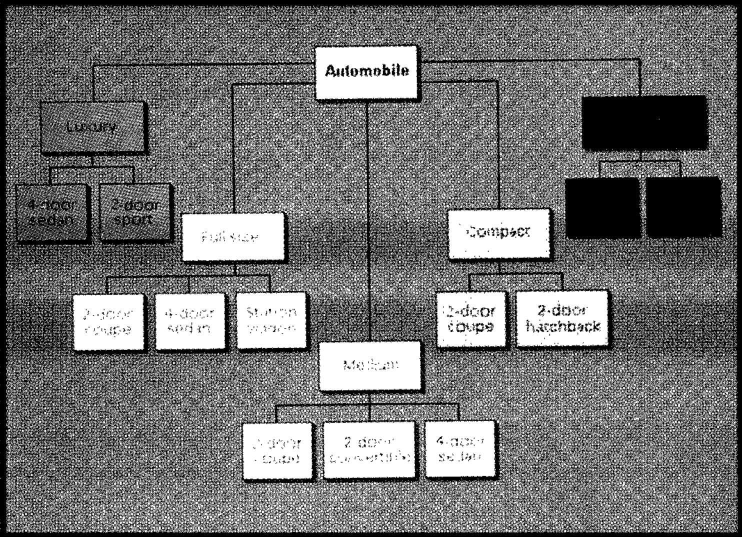


Figure 6 [illegible]

user's focus is on finding, opening, and manipulating data objects. Application programs and the concepts of starting and running programs are transparent to most users. Icons represent objects instead of application programs, and windows provide one or more views of those objects. This method of interaction gives a user the ability to modify objects without having to explicitly find and start a program first.

We call this model the Workplace Model because the background of the screen, which was called the *desktop* in the Graphical Model, can contain objects for performing a variety of tasks, such as preparing reports, selling cars, or controlling manufacturing processes. All kinds of objects can be used on the workplace.

Work areas are container objects that can be used within the Workplace Model to group objects used in specific tasks. For example, a user might create one work area to contain the objects used in preparing a monthly activity report and another work area to prepare an annual sales analysis report. The Workplace Model allows an object to have several icons that can appear in different places in the interface at the same time. If an object, such as a personal calendar, is needed in several different tasks, additional icons, each representing the calendar, can be created and placed in the appropriate work areas.

Work areas help the user maintain context and distinction between the two tasks. For example, the windows for objects opened from a particular work area are treated as a group. They can be closed and opened as a group as the user desires, such as when switching between tasks.

By grouping objects in this manner, users can have all of the objects and views of those objects needed to perform a particular task in one convenient place instead of having to search the system to locate the objects each time they are needed. Figure 4 shows an example of the Workplace Model user interface, as described in CUA91.

The Workplace Model extends the Graphical Model primarily in the following ways:

- It is object-oriented.
- It defines standard CUA objects, which are provided by the system.
- Objects are composed of other objects.
- A user can see multiple views of each object.
- Direct manipulation allows users to perform basic actions on objects directly without interacting with menus.

The 1989 version of the CUA design guide described the Graphical Model, and introduced the Workplace Model. The current CUA design guide and design reference publications[1,2] focus on the design of the Workplace Model, but also describe the Graphical Model and the migration from the Graphical Model to the Workplace Model.

An object-oriented user interface focuses the user on finding, opening, and manipulating data objects.

Migrating from the Graphical Model to the Workplace Model. When designing the user interface for products, designers can use either the Graphical Model or the Workplace Model. We recommend that those designers who are either in the early design stages of a new product or who have not yet begun that design process should plan to implement the Workplace Model.

However, moving from an application-oriented user interface to an object-oriented user interface (OOUI) may be a major step for designers who are well into product development or who support products that have current versions being used by customers. Designers who fit into one of these categories will need time to plan for and implement the transition from application-orientation to object-orientation. Therefore, CUA allows for both the Graphical and Workplace Models.

The following sections explain some of the concepts on which an OOUI model such as the Workplace Model is based, including definitions of objects, object classes, and object containment, and a description of the benefits of using an OOUI.

What is an object-oriented user interface? When using an object-oriented user interface, the user's focus is on objects. Users see and use representations of their data objects, and each different kind of object supports actions appropriate for the object. Typical users need not be aware of applications, programs, and programming concepts.

Different kinds of objects are said to belong to different *classes.* Class distinctions are based on behaviors that are common within a group of objects and that differ between groups of objects. Users can perceive common characteristics despite specializations within a group of objects. Specializations are achieved by using *subclasses*. For example, a folder is a general-purpose container that can contain objects of many different classes. Users learn the properties and behaviors of folders that allow them to add to, arrange, and view a folder's contents, and see how these behaviors can be used to accomplish their tasks. Another type of object, such as a portfolio, may be a specialization, or subclass, of the folder class. A subclass *inherits* properties and behaviors from its parent class. New properties and behaviors are added to the subclass by its developer to create the desired specialization. Users of the subclass object benefit from a transfer in learning achieved by the inheritance of properties and behaviors they usually already understand. They need only learn the distinguishing features of the new type of object to take advantage of its intended benefits.

Objects are composed of and contain other objects, which can be used individually or collectively. That is, objects are composed of other objects, which in turn are composed of yet other objects, all the way down to elemental-level objects, which cannot be further decomposed by the user.

An OOUI allows a user to focus on objects and work with them directly, which more closely reflects the user's real-world way of doing tasks rather than having to go through an application to get to objects.

Objects. An *object* is something that a user needs to work with to perform a task. It is any entity that can be manipulated as a unit, or that can be thought of by a user as capable of existing independently, such as a spreadsheet, a cell in a spreadsheet, a bar chart, a bar in a bar chart, a report, a telephone number, a folder, a printer, a string of characters, or even a single character.

Each of us deals with objects daily. Some objects, such as a telephone, are so common that we find them in many places. Other objects, like the folders in a file cabinet or the tools we use for home repairs, may be particular to a certain place.

The Workplace Model allows users to organize objects in the computer environment similar to the way they organize objects in the real world. Users can keep objects used across many tasks in a common, convenient place. And, they can keep objects used for specific tasks in specific places.

The CUA design guide[1] describes how we designed a sample CUA application for a car dealership. The following list contains examples of objects that are used in that application.

- A work area for selling cars that contains worksheets
- A worksheet that contains the details about a car that a customer wants to buy
- A list of the new cars in stock
- Pictures of the cars in stock and those that can be ordered, for use in the new car stock list
- A printer to print the worksheet

These objects are also shown in Figure 5. The figure is taken from a Smalltalk V/PM prototype, one of the tools used in the design and testing of the CUA91 interface.

Objects are often represented on a user's screen as icons, small graphic images that help a user identify an object. Icons are used to provide a concise, easy-to-manipulate representation of an object regardless of how much additional information the object may contain. A user can *open* an icon to see a view of this additional information in a window if desired.

Users can perform actions on objects by using various techniques, including point-and-select and direct manipulation.

Object classes. Object classes are used to distinguish one type of object from another. Object classes are very useful because they help designers make clear distinctions between the types of objects that their products need to provide. These distinctions, in turn, make it easy for a user to learn and predict how an object will behave.

Different properties and behaviors are used as the basis for making distinctions between classes and subclasses. The basis for distinction must be clearly defined and be relevant to the users of the objects. This is easy to see when parallels are drawn between computer objects and objects that are found in the world in general. For example, "automobile" could be a class that includes properties such as body style, price range, and optimizations for intended uses, among others. Figure 6 shows an automobile class hierarchy based on body style and size for a particular manufacturer.

CUA uses a class hierarchy for computer objects. "Data," for example, is a class that includes document, chart, and picture as subclasses.

Each object the user interacts with, then, is an *instance*, or unique occurrence, of an object class or subclass. The automobiles on a car dealership's lot are instances of the luxury, full-size, medium, compact, and utility subclasses, just as a written memo is an instance of the "document" subclass.

The distinctions that allow objects to be grouped into classes are their characteristics and uses or, to use OOUI terminology, their *properties* and *behaviors.*

Just as the real cars on the car dealership's lot are instances of automobiles, the car icons that could be shown in a computerized version of the dealer's lot represent instances. Each instance has the same properties: year, make, model, and so forth. Also, each instance follows the same rules of behavior. An action performed on one of these instances, such as changing the price, could be performed on any other instance of the automobile class.

Icons help to depict the class of an object by providing a pictorial representation of the object. For example, icons help a user to see that the objects in a list belong to the "automobile" class.

Although users create and manipulate objects, many users will never have to be consciously aware of which class an object belongs to. For example, a person approaching an office chair does not need to stop and think, "This is an office chair, which belongs to the class *chair.* Therefore, I can sit in it." Likewise, a user can work with charts and come to expect that all charts will behave in the same way without caring that the charts belong to the data object class.

Object classes are also very useful to product designers because they prompt designers to think about making clear distinctions among the types of objects that products need to provide. Object classes must be carefully defined with respect to users' tasks and distinctions users currently understand that are useful. When object classes are carefully defined, these distinctions make it easy for a user to learn and predict how an object will behave.

Object containment. All objects except the most elemental ones are composed of and may contain other objects. For example, a spreadsheet is an object that is composed of cells, and cells are objects that can contain text, mathematical formulas, video, and so forth. The breaking down of objects into the objects from which they are composed is called *decomposition.*

How far object decomposition should be extended depends entirely on what a user finds practical or useful for performing a particular task. A user who is writing a report, for example, would probably not be interested in dealing with objects smaller than characters, so here characters would be considered elemental objects. However, a user who is creating or editing a character font might need to manipulate individual pixels. In this case characters would be composed of pixels and therefore would not be elemental objects.

Benefits of using an OOUI. By extending the Graphical Model into the realm of object orientation, the object-oriented user interface provides the following specific new benefits:

- 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 the programming that is providing the functions that the user needs. Instead, the user can concentrate on the objects and the actions that the user wants to perform. The aspects of starting and running programs are hidden to all but those users who want to be aware of these aspects. A user should only need to know which objects are required to complete the task and how to manipulate those objects to achieve the desired result.

- Removal of user interaction requirements and obstacles

  An OOUI removes user interaction requirements and obstacles that some existing graphical 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. A computer is a tool and, as with any other tool, it has to be learned to be used effectively. However, when we can help a user by simplifying the process of learning to use a tool, we should do so.

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. Each object class that a product provides should be distinctly different from the other object classes provided by that product in a way that users find natural, easy to remember, and useful.

The more similar object classes are, the harder it is for a user to remember their differences. Therefore, designers must provide obvious and useful distinctions between object classes so that an object's behavior is obvious and useful to a user.

Key aspects of CUA's Workplace Model

The CUA91 guidelines are intended to support creation of an interface adhering to a set of object-oriented characteristics we call the *key aspects.* The key aspects of CUA's Workplace Model are:

- The Workplace Model is object-oriented.
- Standard objects and controls are defined.
- Objects provide multiple concurrent views of themselves.
- Objects are composed of and contain other objects.
- Objects can be interconnected.
- Direct manipulation is provided by all objects.
- Icons reflect dynamically changing properties of objects.
- Changes to objects are immediate, but reversible.
- Visual and interaction paradigms are pervasive.
- Window types are based on user-task distinctions.
- Users can control groups of related windows.

The key aspects of the CUA Workplace Model are introduced briefly here. Many are addressed in greater detail in Reference 5, which describes the CUA Object Model. The CUA91 publications support many of these key aspects. A few, however, have only initial levels of guideline definition. These will be extended and refined over time.

Object orientation. The Workplace Model focuses on user objects required to accomplish user tasks. The aspects of starting and running programs are hidden, but they are accessible to those users who wish to be knowledgeable of these aspects themselves. In Figure 5, the objects that a car salesperson might use are shown in the workplace, but no application programs are visible because the salesperson interacts directly with objects with which the salesperson is already familiar.

Notice the graphic images of objects (New Car Lot work area, Worksheet, Customer feedback, In-basket, Printer, and Delete Folder) on the bottom of the screen. The graphic images of objects are represented by icons. Icons represent objects that a user can manipulate to accomplish a task, such as selling a car, and that can be placed in a container, such as a folder, a work area, or the workplace. Designers should allow any object to be placed on the workplace, as a temporary "parking place," and in general-purpose containers such as folders. Therefore, an icon should be available for every object. For example, if a user selects an arbitrary string of text in a document, then drags the string and drops it onto the workplace (drag/drop) for temporary placement in the course of locating and opening the final intended destination, an icon representing the string of text should appear. Its title might be the first few words of the text string.

The icon labeled "New Car Lot" represents a salesperson's *work area.* Notice the diagonal stripes on the New Car Lot icon. This is a visual paradigm indicating that the work area is open and being used.

Standard objects and controls. The CUA interface defines standard objects and controls for use by many products. Examples of standard objects include workplaces, folders, work areas, and printers. Examples of standard controls include entry fields, list boxes, radio buttons, push buttons, and pop-up menus. The standard objects and controls have CUA-defined visual representations and support standard interaction techniques, which provide for consistency across products.

A user typically has one workplace containing objects (e.g., printers) and other containers (e.g., work areas).

Each standard CUA-defined object has been designed for a particular role in the user interface. For example, work areas are CUA-defined standard containers that are used to group objects that are used together to perform a task. A user can place any object in a work area. Windows that are opened from objects in a work area are grouped together for opening and closing. This allows a user to avoid excessive window clutter when switching between multiple concurrent tasks. All windows opened from a work area are automatically closed together; and they are reopened together when the work area is reopened.

Work areas are typically created and arranged by users, although products can provide initial work areas to help users get started with specific tasks.

The background of the screen, on which windows and icons are displayed, is called the *workplace*. The workplace is a container whose view fills the entire screen. An individual user typically uses a single workplace and that workplace contains all objects accessible by the user, including objects that reside on remote servers and host computers.

Multiple views of objects. Objects can provide different *views* of themselves, and multiple views of an object can be displayed concurrently. A view is a representation of details about an object, such as what it contains, how it is composed, or what its properties are.

Although an icon also represents an object, it is intended to provide a concise representation that is easy to manipulate. The emphasis in using icons is to provide sufficient and useful information for dealing with the object as a whole. Views of an object displayed in a window allow a user to "look inside" and manipulate the contents, composition, and properties of an object. Figure 5 shows a window that displays a view of the New Car Lot object.

Objects support multiple concurrent views. Each view can show different aspects of the object, such as its contents or its settings, or the same aspects in different ways, such as listing its contents in an iconic format or in a format that uses small icons and provides details about the contents.

Object composition and containment. All but the most elemental objects are composed of and may contain other objects. For example, a folder can contain spreadsheets, graphs, printers, and other folders. A spreadsheet is composed of cells but a cell might contain text, graphics, video, or even another group of cells.

A queued printer might be presented to the user as a composition of a container, representing the queue, and a printer device. Together they would appear as a single printer icon on the workplace, but the user might be provided with a view of the printer that showed this composition. This view would provide additional capabilities and user control by allowing the user to manipulate the composition. For example, the user might be allowed to manipulate the connection between the queue and the printer device by dragging a connecting line between them, thus providing a direct manipulation approach to holding and releasing the queue.

Composition and containment aspects of objects are represented by providing appropriate views. Composite objects, such as a newsletter that contains text, figures, and photographic images, provide views that identify and support manipulation of these individual components. Container objects, such as folders, provide views that identify and support manipulation of the objects they contain. All objects provide consistent techniques for accessing and working with objects regardless of how they are composed or what they contain.

The Workplace Model supports these aspects by defining different types of views so users can look at objects in different ways, such as composed views and contents views, and a menu bar style that allows users to work with the individual parts of an object as well as its contents.

Object connections. Objects can be connected for navigation (hypermedia) and data transfer (linking). *Navigation connections* allow users to specify connections between objects so they can access one object from another. For example, the model name of a car in a text description can be connected to a picture of the car, which would appear in a window when requested by a user clicking a mouse button on the car's model name.

Data transfer connections allow users to transfer data between objects. Once a connection is established between two objects, users can specify how and when data transfer between the two should occur. For example, a user may identify some cells in a spreadsheet and connect them to a bar chart so that the bars automatically change size to reflect changes to the values in the cells, and vice versa.

Reference 2 specifies navigation connections for displaying help information. These guidelines will be extended over time and will address user tailoring of connection options. Guidelines for additional usages of navigation connections and for data transfer connections will also be added over time.

Direct manipulation is provided by all objects. A user can accomplish actions by using *drag/drop* and *pop-up menus* on most objects. The CUA architecture considers drag/drop and pop-up menus as two degrees of direct manipulation. Drag/drop is a more direct action mechanism than is a pop-up menu, but a pop-up menu is also more direct than is a menu bar.

Each object provides direct manipulation capabilities regardless of how it is used in relation to other objects. That is, drag/drop and pop-up menus are provided by objects at all levels of composition and containment, not just for icons on the workplace.

A user can display a pop-up menu for any object. A pop-up menu is a menu that is displayed next to, and contains choices appropriate for, a given object or set of objects in the current context. Create, move, copy, connect, print, and discard are typical actions that can be performed by using drag/drop or a pop-up menu.

Icons reflect dynamically changing properties of objects. Icons are a mechanism for representing objects visually and for conveying important aspects about an object, such as its class. Most objects undergo state changes that are of interest to a user. When a view of the object is being presented in a window the state changes are usually obvious. State changes that are of interest regardless of whether a window is open on the object should be reflected in changes to the icon for the object. For example, when a printer is out of paper the printer icon can be augmented with a visual indication of the condition. Similarly, containers, such as folders, queued printers, and mail baskets, should display a dynamically updated count of the number of objects they contain.

Immediate, but reversible, actions. The effects of changes are shown and recorded immediately, but are reversible. For example, when a user chooses a different font for some specified text, the text should change immediately to reflect the font chosen. If the result is not what the user wanted, the user can immediately choose a different font.

As technology improves, the distinction between the computer's temporary memory (random access memory, or RAM) and longer term memory (such as a disk) is being eliminated. Changes made by users will be saved without requiring users to perform an explicit saving action. These changes will also be reversible, for example, allowing users to create and manage multiple versions of a document.

Pervasive visual and interaction paradigms. Paradigms for displaying object states, moving the cursor, selecting, and editing, are consistent across object types and are appropriate to the type of view being presented. For example, designers must allow text to be edited by providing the same editing interaction techniques at all times, wherever text editing is available, such as in window titles, icon labels, text in entry fields, and text in an object view.

Task-relevant window types. A window's type is based on how it is used to support user tasks. Window types are defined in terms of:

- The type of information displayed in the window
- How the window relates to other windows with which it is used

There are three window types with respect to the type of information displayed in the window: object windows, action option windows, and message windows. *Object windows* display views of objects. They appear when a user opens an icon or requests additional views of an object by using a menu of available views.

Action option windows contain options that allow a user to further specify an action request. They appear when a user selects an action choice, such as "Print. . .", from a menu. Menu choices that cause action windows to be displayed are called *action choices* and are followed by ellipses. This provides a visual cue for users, indicating that no action will occur until further dialog is completed.

Message windows contain information about situations that arise unexpectedly and that may require intervention by the user.

Because each of these window types has a specific yet different role in the user-computer dialog, CUA specifies standard layout and interaction guidelines to aid consistency and make them easy to use.

There are two window types with respect to the relationships between windows: primary windows and secondary windows. Users can open and close primary windows independent of other windows that are currently displayed. A view of an object is typically shown in a *primary window*. Users can open and close views of objects as needed.

Other windows that are dependent on a primary window are called *secondary windows*. Action option windows and message windows are typically secondary windows because the information they display is usually not relevant or useful independent of information in a primary window.

When a user closes a primary window, all of its secondary windows close also.

User control of window groups. As mentioned previously, the work area is an object that provides users with a degree of automatic window management. Users can implicitly create groups of related windows by placing objects in a work area. The windows opened from a work area are grouped together for purposes of opening and closing. However, users should be provided with greater levels of control and flexibility, and explicit techniques should be available. Over time, the CUA guidelines will be extended to provide users with the ability to explicitly group and ungroup windows.

Summary

This paper has provided an example of factors that can drive and influence the evolution of a user interface style. In the evolution of CUA, significant shifts in focus have occurred in approximately two-year cycles. The initial goal of CUA was simply to achieve user interface consistency within and across the mainframe and personal computing environments. As the role of applications and application development tools became increasingly important, the focus shifted to emphasize application support in each environment. Finally, as the interface began to mature the focus shifted to advancing the user interface technology. We have also provided a glimpse of our goals for the future by introducing the object-oriented style of CUA91 and its Workplace Model. For a detailed description of the CUA Workplace Model, as well as a description of user interface design models used in the development of CUA91, see the paper by Berry (Reference 5).

*Trademark or registered trademark of International Business Machines Corporation.

**Trademark or registered trademark of Microsoft Corporation, Easel Corporation, Aldus Corporation, Micrografx, Inc., Open Software Foundation, Inc., UNIX Systems Laboratories, Inc., Lotus Development Corporation, WordPerfect Corporation, or Digitalk, Inc.

Cited references

1. *Systems Application Architecture Common User Access Guide to User Interface Design*, SC34-4289, IBM Corporation (October 1991); available through IBM branch offices.
2. *Systems Application Architecture Common User Access Advanced Interface Design Reference*, SC34-4290, IBM Corporation (October 1991); available through IBM branch offices.
3. D. Norman, *The Psychology of Everyday Things*, Basic Books, New York (1988).

4. I. Graham, *Object Oriented Methods*, Addison-Wesley Publishing Co., Wokingham, England (1991).
5. R. E. Berry, "The Designer's Model of the CUA Workplace," *IBM Systems Journal* **31**, No. 3, 429–458 (1992, this issue).

Accepted for publication March 17, 1992.

Richard E. Berry *IBM Personal Systems Programming, 11400 Burnet Road, Austin, Texas 78758.* Mr. Perry is a Senior Technical Staff Member in the object technology area in IBM's Personal Systems Programming group in Austin, Texas. He joined IBM in 1968 in the Albuquerque, New Mexico, branch office where he performed a variety of programming maintenance and systems engineering duties. Since moving to programming development in 1971, he has held various technical and management positions including: lead programmer and chief designer of the user programming facility for the IBM 3650 Retail Store System; programming development manager for the Retail Store System; lead architect for the IBM 5520 Administrative System Files Processing, and architecture manager for the IBM 5520 Externals Design. Throughout his career Mr. Berry has concentrated on defining product function and user interfaces. In 1982 he was appointed lead architect of IBM's User Interface Architecture which became the Common User Access (CUA) component of Systems Application Architecture (SAA) in 1987. He was one of the codesigners of the Workplace Model.

Cliff J. Reeves *IBM Personal Systems Programming, 11400 Burnet Road, Austin, Texas 78758.* Mr. Reeves is the manager of the object technology business area in IBM's Personal Systems Programming group in Austin, Texas. He was formerly the manager of IBM's Common User Access team in Cary, North Carolina. Mr. Reeves joined IBM in England in 1971 and has held a variety of programming and management positions in the areas of IBM's Cross System Product, Data Processing (DP) Professional Products, and Decision Support Products. He was the manager of user interface strategy for Office Vision/2 and also managed IBM's business and technical relationships with EASEL Corporation and Digitalk, Inc. In 1989 he was appointed manager of CUA and directed the development of the CUA interface for 1991. Mr. Reeves is currently responsible for IBM's business relationship with Taligent and for future use and IBM marketing of Taligent products.

Reprint Order No. G321-5480.