1989

A window-based user interface :

Gyutae Baek
Lehigh University

A Window-Based User Interface

with the psychological aspect

for A Knowledge-Based System

by

Gyutae Baek

A Thesis

Presented to the Graduate Committee

of Lehigh University

in Candidacy for the Degree of

Master of Science

in

Computer Science

Lehigh University

1989

CERTIFICATE OF APPROVAL

This thesis is accepted and approved in partial fulfillment of the requirements for the degree of Master of Science in Computer Science.

May 16, 1989
Date

Howard J. Hilaman
Professor in Charge

Lawrence J. Varnerin
Chairman of Department

ABSTRACT

A good user interface is vital to application programs and users, and very hard to design. The design of a good user interface for knowledge-based systems on the SUN workstation is a challenge. The user interface should be designed from a psychological perspective in order to make the user dialogue easy and efficient. The window-based user interface on a powerful workstation is an effective method to explain the system's reasoning and knowledge for those knowledge-based systems which have the poor interface capabilities.

Table of Contents

CHAPTER 1

INTRODUCTION

The complexity of today's software has made better communication techniques necessary. For large-scale knowledge-based systems, program complexity makes it more difficult to convey meaningful information to the users. These programs normally have little or no instructional capabilities. Therefore, a knowledge-based system should be utilized in conjunction with a User Interface Management System(UIMS) which will make the communication between the user and the application program effective and easy.

The user interacts directly with the user interface, not with the system which manages all aspects of the UIMS. A user interface can serve several kinds of users, including implements, novices, and experts. To interact efficiently with the user, the user interface should be designed with psychological aspect according to the different users by using a good UIMS. In recent years, a number of UIMS have used windows as the basis for advanced and good user interfaces. Efficient workstations with bitmapped screens provide the new technological base for window-based and graphic user interfaces. The window-based UIMS on a powerful workstation should support a variety of selection options including objects from a display and items from menus through multiple windows using a mouse. The window-based UIMS with these desirable abilities provide many opportunities for exposing the knowledge-based system's reasoning and the knowledge captured by the user. This means that a well-structured window-based user interface in knowledge-based system allows a user to browse easily through the knowledge base and view reasoning processes through windows.

The window-based user interface between the user and BFI knowledge-based system is implemented on a SUN workstation under the operating system Unix. The basic concept of the UIMS and psychological aspects in UIMS is described in chapter2,3. Also, this thesis presents the model of the window-based user interface for a knowledge-based system on the workstation.

CHAPTER 2

USER INTERFACE MANAGEMENT SYSTEM

2.1 HISTORY

Recent research has been directed at the development of better tools to support the design, implementation, specification and evaluation of human-computer interfaces. Such tools have been variously named "user interface management systems", "dialogue management systems", and "abstract interaction handlers". User interface management systems build upon concepts developed in earlier user interface specification and prototyping systems. The state-of-the-art is now at the point where user interface management system(UIMS) packages are becoming commercially available. This set of software tools will have a significant impact on the structure of applications in the future. Its advantages include faster and more reliable application domains, and easily modifiable user interfaces.

2.2 DEFINITION OF USER INTERFACE MANAGEMENT SYSTEM

The computer program is divided into two parts according to role. One part is the role of communication. In this part, the user interface of the computer program displays output and accepts input from the user. The rest of the computer program is the role of task, in this case of the application. The User Interface Management System is a set of tools to support the design, management, implementation, maintenance, and evaluation of the interface between the user

and the application. Such a system is necessary, due to the innate complexity of user-computer interaction.

A simple model of a interface between the user and the application is shown Fig. 1. The application can not converse with the user directly without the user interface knowing about it. In this simple model, the user interface can communicate with only two components by the useful tools of UIMS. On the one side there is a user who conducts a dialogue with the user interface by sitting at a computer. On the other side there is the application program which the user interface draws upon to perform task and manage data. The convenience and ease of dialogue between the user and the application depends on both the UIMS facility and user interface efficiency.

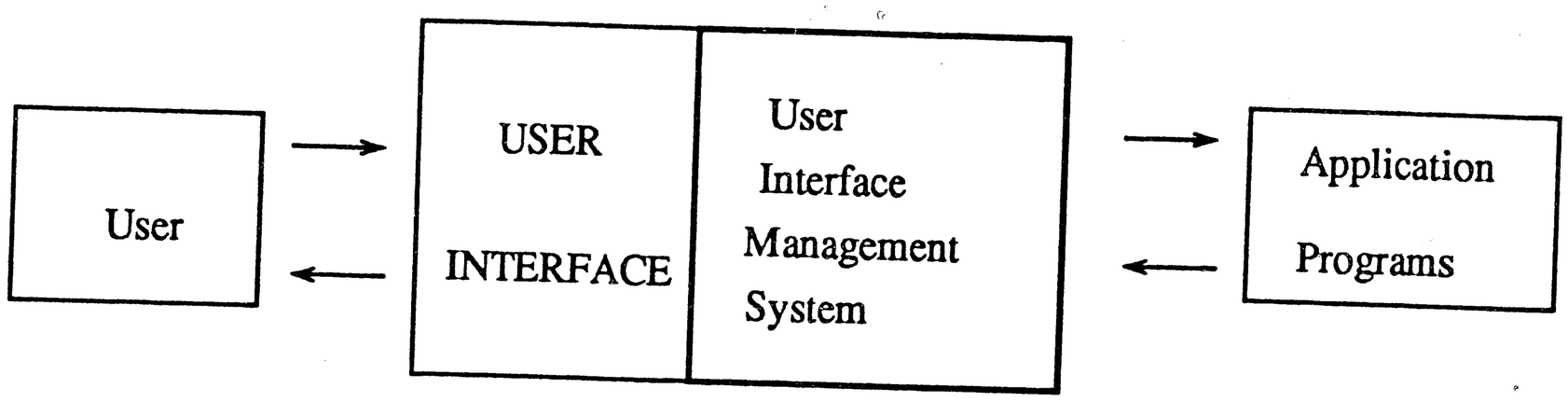

Fig. 1 A simple model of a interface batween the user and the application programs

In brief, a UIMS can be thought of as the mediator system between the user and the application. We can divide the UIMS into three components:

(1) Interaction method library

There are many methods to input a certain type of data by using input devices such as menus, buttons, typing, and sliders, etc. This library contains a collection of interaction methods.

(2) Control component

This component controls the sequence of events and interaction methods.

(3) Analysis component

The analysis component supports the evaluate of the user interface after it has been created. If the interaction with the user is graphical in nature, the UIMS can make use of a graphics system. In this case, such a graphics system would be an integral part of the UIMS. The author believes that a UIMS represents a higher level of abstraction than a graphics subroutine package.

2.3 PREFERABLE FEATURES IN USER INTERFACE

There are four preferable features in a UIMS. It is a difficult task to accomplish all these features. However, a structured framework in a UIMS can help to confront each of the requirements separately. It will not only help in designing new systems, but also in tailoring the user interface of existing systems to the user's needs.

(1) multimedia multimodal communication

Typical current systems use only alphanumeric text as the medium of communication, and

most graphics systems with multiple input devices allow only one of them to be used at a time. It would be more natural to give the user the freedom of choosing any devices he prefers, e.g. pointing and voice concurrently.

(2) continuous communication

In the conventional style of dialogue(action-reaction cycle), the user become bored if the response times of the computer is longer than a second. Even more, if the delay of the answer is non-determinably changing, the user will become frustrated with the repeated waiting and checking while the operation is being completed.

(3) interleaving concurrent tasks

Instead of formulating every action as a discrete command, the user should be able to deviate and return again. For example in a CAD system, it would be useful to make data base queries during the input phase, to help design decisions.

(4) free-formed commands

The difficulty of learning available commands of computer systems often hinders their acceptability. A program asking the user everything separately, or guiding him through modes by changing menus, is only a partial solution. Better system would have natural-like language understanding, and possibly even adapting to the user's gestures.

2.4 Advantages of using UIMS

The UIMS is used by four humans. These are (1) the designer of UIMS and (2) the designer of user interface and (3) the application programmer, and(4) the user. When the designer of UI uses the UIMS, there are many advantages:

. The overall quality of the user interface should be higher because it will be easier to change the problems discovered through testing.

. Applications can be quickly and economically maintained.

. The user interface will have more consistency within the applications.

. Designs can be rapidly prototype and implemented.

. Application programmers can easily and quickly use and modify the user interface.

2.5 Various Approaches to UIMS Structure

(1) Unstructured Approach

Lack of programming methodology leads to a single module without any structure. The I/O devices are called directly by the application, and their calls are spread over the program. This makes the program hard to understand and it has poor portability. This kind of system is seldom made today.

(2) User Interface Management with Internal Control

Separating the direct handling of I/O devices in a module of "logical device" subroutines provides good portability of the application programs. This is the most common approach in current interactive systems. Internal control means that the management of communication devices is handled by an application program. So this is "the machine's point of view". A single application reserves the workstation, and the user is controlled by the machine who makes requests.

(3) User Interface Management with External Control

If an application task is thoroughly analyzed, it can be subdivided into natural subtasks that

are relatively independent. Programming these subtasks as non-interactive modules callable from a master command brings us to the external control approach. Every subtask can be initiated directly, without navigating through a sequence of submodes. If it can be guaranteed that all the subtasks are small enough, then the response to commands is practically instantaneous, and the user has the feeling of continuous communication. In several senses this is "the user's point of view". The user can command the application, and not just answer interrogations. The problem with external control User Interface Management is that most existing applications are designed with internal control and the division into subtasks and tailoring for external control may be difficult.

(4) Window-based User Interface Management

One user, at one workstation, deals with different independent applications. In many cases, such a technique presents itself to the user by a number of windows on a screen, where each window is associated with a different application process running. The user selects the one process with which he wants to communicate by moving the cursor into that window with the mouse. The next section explains the window-based UIMS in detail.

2.6 Window-based User Interface Management System

2.6.1 General Description

In today's computer workstations it is realized that a good user interface management system should be placed as near to the user as possible. Also, with the increasing use of graphics

workstations in an variety of applications, the need to manage the screen in an effective manner has become more important than ever. Software supporting screen layout is called a window manager, and the individual areas are known as windows. For each process there exists a dedicated window on the screen.

Using a mouse, a window can be activated and can reserve other input devices for its process. For the activated process the window is transparent, in the sense that it has virtually direct control of all I/O devices associated with that window. The user interface management system must have facilities for multiplexing real devices for its process.

2.6.2 Role of the Window-based UIMS

A window-based UIMS is a system service that provide for the creation, deletion, and modification of windows. The window-based UIMS's prime functions are resource management, protection and providing an interface to the user. A major function is ensuring that application output does not stray outside the window boundary. Also it provides efficient tools to the interface designer and allows a user to communicate very easily and reliably with an application.

The simple model of a window-based interface between the user and the application is as shown in Figure 2. The application calls various graphical primitives of the window which are displayed on the screen. A single mouse has to be multiplexed between a number of different applications.

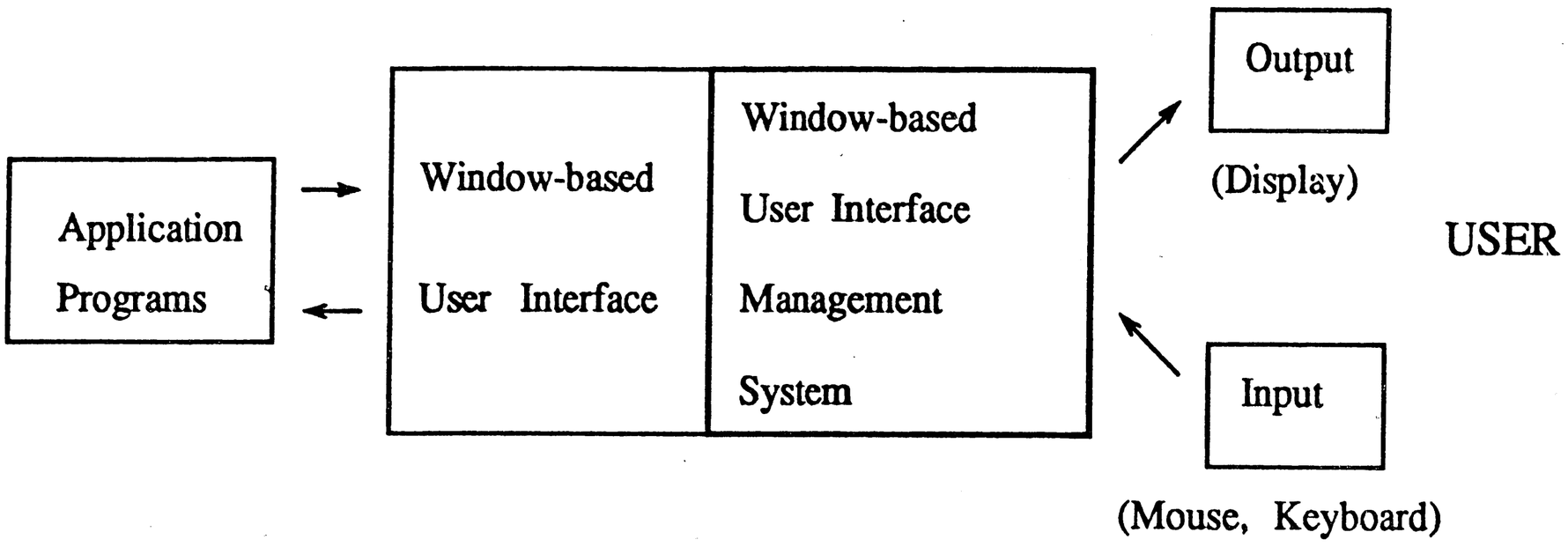

Fig. 2 A simple model of window-based interface between the user and the application programs

2.6.3 Properties of a Window-based UIMS

A window-based UIMS has the following characteristics.

(1) It controls the resources associated with only one display space.

(2) It manages a set of drawing spaces from one or more application concurrently.

(3) Only one application at a time can write into a drawing space. The application is said to own the drawing space and the associated window.

(4) A single application can own several windows concurrently.

(5) It is responsible for ensuring that the owner application does not draw outside the window boundary.

(6) Windows go out of existence when the application that owns them is halted.

2.6.4 Controlling the Window-based UIMS

There are two methods by which a user can control the window-based UIMS:

(1) Indirectly via an application (the application program interface)

(2) Directly via a special layout task (the user interface)

Some of the functions expected to be included in the application program interface are:

- creating and destroying windows
- redrawing images in windows
- providing titles for windows
- requesting a sampling input from the mouse, keyboard, or function button

Some of the functions expected to be included in the user interface are:

- resizing and repositioning a window
- changing a window display priority
- reassigning an input device from one application to another

CHAPTER 3

PSYCHOLOGICAL CONCEPT FOR THE HUMAN-COMPUTER INTERACTION

When the interaction between a man and a computer is well designed, using the computer is so natural that a user does not step back. By identifying the psychological processes used in performing a human-computer interaction, it would be easier to develop a good user interface management system. This chapter will describe the basic psychological concepts for human-computer interaction.

3.1 Human Information Processing

The goal is to convey a version of the existing psychological science base in a form suitable for analyzing human-computer interaction. The human mind is like an information processing system. A description of how the human mind processes information is useful in making the model of the information processing system. The human mind can be divided into three processes: the perceptual process, the cognitive process, and the active process. It is possible to organize the description of the psychological science base around these processes.

[The Model Human Processor]

Stuart K. Card, Thomas P. Moran and Allen Newell described characteristics of the human and developed the model as an information processor relevant to human-computer interaction.

The model gives an integrated description of the psychological knowledge about human performance. The Model Human Processor consists of three interacting subsystems mentioned above. The Model Human Processor can be described by a set of memories and processors together with a set of principles. The next sections describe the three processes in this model.

3.1.1 The Perceptual System

When interacting with computers, users often receive and send information in the medium of written language. The characteristics and limitations of the human information processing system will be relevant for maximizing the operating efficiency of the user interface management system. There are several sub-processes which are often distinguished in discussions of this complex process.

(1) The user must first perceive the visual patterns of the words through sensors. On the basis of this visual information the user must decode the meanings of the individual words present. According to current theories of this process, the meaning of words are stored in a kind of internal buffer memory called the lexicon.

(2) Users must relate the meanings of the individual words in a sentence. The meaning of a sentence will depend not only on the individual words but on the way in which they are combined.

(3) The last step involved is the combination of the meanings of the sentences in order to reach an understanding of the process as a whole.

Ease in understanding the written material will depend on the characteristics of the process

and familiarity with the concepts involved. Icons and menus, and buttons aid in helping the user understand the system more quickly. Colorful windows make a user to rapidly distinguish among several categories of data.

3.1.2 The Cognitive System

The cognitive system receives symbolically coded information from the sensory images stored in its working memory and uses previously stored information in long term memory to decide how to respond. Working memory holds the intermediate products of thinking and the representations produced by the perceptual system. Working memory is where all mental operations obtain their operands and leave their output for the motor system. Long term memory holds the user's mass of available knowledge which is accessed associatively from the contents of working memory. A general representation of the information processing model is presented in Fig.3

3.1.3 The Active System

The Active system carries out the response. For computer users, there are two important sets of effectors. They are the head-eye movement and hand-arm-finger movement. Among the many input devices, the mouse is the fast device for picking and positioning.

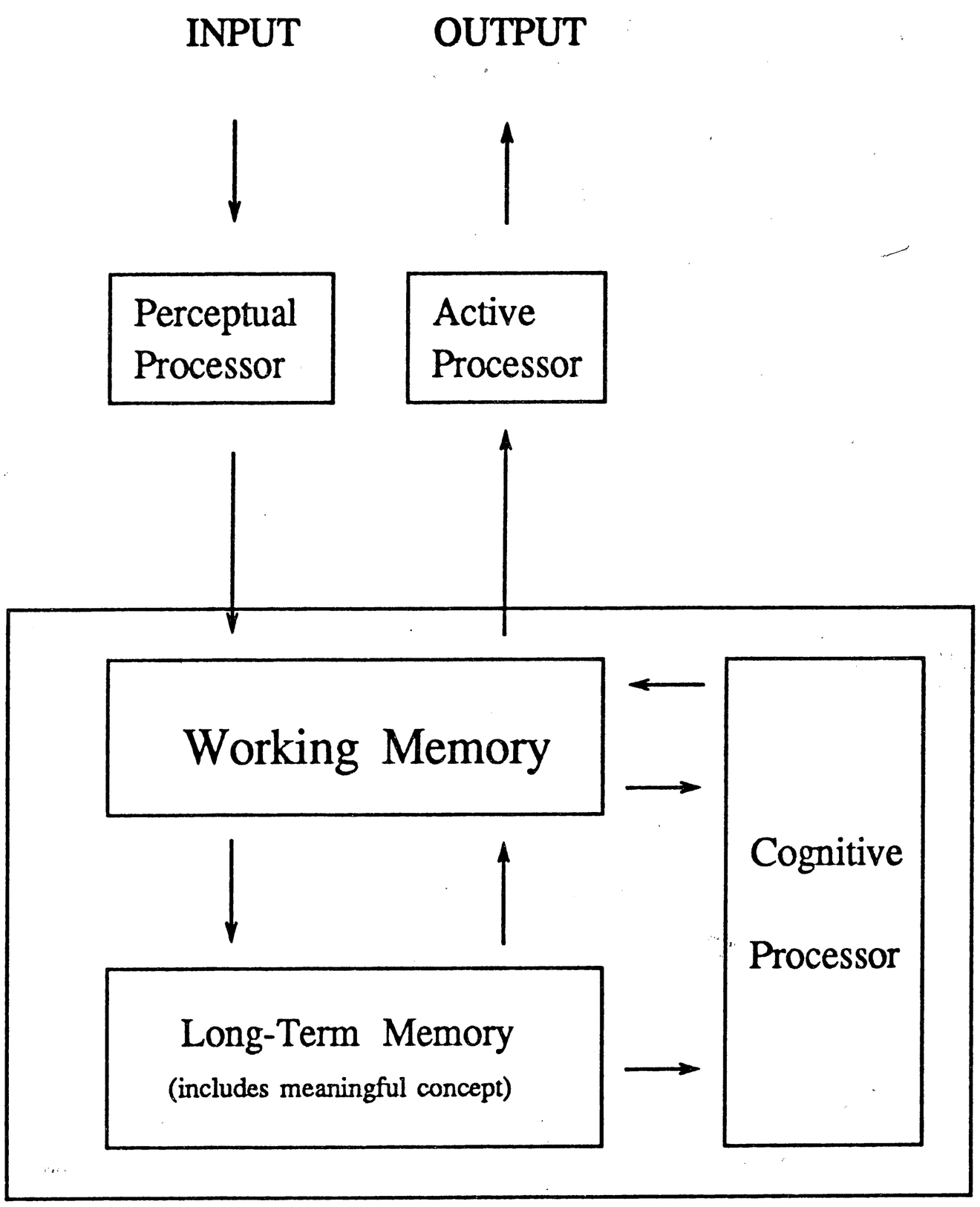

Fig. 3 The components of the information processing system

3.2 The Window as an external memory

The analysis of the effect of the human information processor on window use begins with the proposition:

. A fundamental constraint on the user's cognitive performance arises from limitations of working memory.

The ability to do mental arithmetic is limited largely by difficulties in keeping track of the intermediate products and keeping one's place. The limitation of the number of mental things that can be kept track of is a strong constraint on human cognitive capabilities.

The display of a computer provides the possibility of giving the user an external memory that is an extension of the user's own memory. The computer display is not only an external memory, it is also the communication medium between the user and the computer. The full power of the display as an external memory and communications medium and the cooperative interaction between these are only realized when the display supports independent, but related, objects of memory and communication. It is for this reason that the windowing technique has become the harbinger of improvement in human-computer interaction.

3.3 Application of Psychological Concept to the UI's Design

3.3.1 Principle

This section rests on a view of how psychological aspects can be applied to system design.

The framework for an application is the structure and performance of the user interface. The structural components of the UIMS are the computer, the user, and the task. The performance of User Interface is determined by its structural variables(computer, user, task) on the basis of psychological aspect.

Task + User + Computer System ------> Performance of User Interface

The basic performance variables are concerned with learning, quality, time, robustness, error, and so on.

. Learning : How does the performance improve over time.

. Quality : How well are the tasks done.

. Robustness : How does performance adapt to unexpected conditions or to new tasks.

. Time : How long does it take the user to do a task with the system.

. Error : What errors are made, how consequential are they.

3.3.2 Computer Variables

The user interface aspects of computers vary in the following variables: architecture, communication style, input devices, etc. The SUN which the BFI system is implemented on uses the UNIX operating system and SunWindows. SunWindows is an attempt to bring windows to a unix system. The SunWindows system supports three layers:

(1) the pixrect layer providing low level pixel manipulation function

(2) the window layer providing a hierarchy of overlapping windows

(3) the SunTools layer providing for the interactive creation of applications and associated windows.

The SunTools layer requires that applications conform to a particular style of control structure, and supplies many of the functions of a User Interface Management System. In the second window layer, SunWindows supports subwindowing through its hierarchical structure of windows. Windows are constrained to lie within the bounds of the screen or the parent window in the hierarchy.

Overall the SunWindow system provides a rich and perhaps complex set of function. The Window-based user interface of the BFI system is implemented using the SunWindows.

3.3.3 User Variable

Users vary widely in many ways; for instance, in experience and motivation. User characteristics can be further subdivided as below:

(1) Ability(cognitive).

Cognitive ability has implications for the ease of initial use of the window manager and the degree to which it supports combined operations.

(2) Motivation

Poorly motivated users place great demands on the window manager to ensure that it does not need much effort to interact effectively with it.

(3) Experience

User's experiences have implications for the selection such as the nature of help, error messages, the complexity of the facility provided, and so on.

An extensible User Interface is necessary to cope with the requirements of a wide range of

users. If the characteristics of the user are known, the default of the User Interface should be matched to their needs and preferences. The strategy of a User Interface for the BFI system is to build a solid theoretical and empirical characterization of the expert user before attending to novice and casual users.

3.3.4 Task Variables

The task is an application program which is operating on a User Interface Management System via a user interface. The use of windows by a user depends heavily on the tasks which the user is trying to accomplish. It is impossible to accomplish an analysis of the window and displav design without consideration of the task for which the windows are used.

CHAPTER 4

OBJECT-ORIENTED APPROACH OF WINDOW-BASED USER INTERFACE

4.1 Object-Oriented Programming

Object-oriented programming is a style of programming that is based on directly representing physical objects and mental concepts in the machine. The goal is to make the machine cognitively aware of the physical world and able to reason about it using mental representations. Object-oriented programming centers around several major concepts:abstract data types, classes, and type hierarchies. An abstract data type is a model that encompasses a type and an associated set of operations. These operations are defined for and characterize the behavior of the underlying type. A class definition describes the behavior of the underlying abstract data type by defining the interface to all the operations that can be performed on the underlying type.

An object is a variable declared to be of a specific class. Such an object encapsulates a state by containing a copy of all the fields of data that are defined in the class definition. Actions may be performed on such an object by invoking methods defined in the class definition. The process of invoking a method is called sending a message to the object. Each class variable or object represents an instance of the class.

Object-oriented programming has the following characteristics.

- Object-oriented programming is extensible because the programmer can create new types

and may be endowed with specific properties and whose behavior is characterized in a class definition.

- Object-oriented programming permits sharing knowledge between related groups of objects because of the inheritance hierarchy. Subclasses can build the knowledge of previous classes by adding more instance variables and methods.
- Object-oriented programs are well-suited for parallelism. Because the knowledge in object-oriented programs is localized, each object containing its own local knowledge and expertise, different processors can work on different objects at the same time.
- Object-oriented programming has great diversity. It is possible to write object-oriented programs in nearly any language.

4.2 Prowindows

4.2.1 General Description

ProWindows, although not a complete object-oriented programming language, is a window-based user interface tool kit designed from an objected perspective. That is, ProWindows is an object-oriented programming package which enables programmers using Prolog to quickly and easily create window-based user interfaces for their Prolog application programs. ProWindows uses the notion of named objects and a scheme for passing messages between objects to create these interfaces. High-level messages are passed between ProWindows and Prolog. ProWindows handles the low level tasks required by the Prolog application program.

Because of the need to dynamically create, manipulate, and destroy visual objects, the development of the user interactive graphic interfaces is most naturally undertaken using an object-oriented programming point of view. From this perspective, objects(such as windows) are treated as abstract entities interacting with Prolog and other objects through the mechanism of message passing.

Prowindows maintains information concerning the state of the windows on the screen, the state of text and graphic entities contained within these windows, and the state of a user's interaction with these entities. These entities are called objects. This state information is accessed through messages. The messages provide a uniform means of changing the state of an object, and for requesting that an object perform an action.

4.2.2 Object

In Prowindows, objects are fully characterized by the tasks they can perform. The objects of Prowindows are grouped together into classes of similar objects, with each particular object being seen as an instance of its class. This structure allows for the creation of new objects having the same behavior as other members of a particular class. Classes are objects in their own right, with their own sets of behaviors. When an object is created, its class is specified. The built-in classes of Prowindows are roughly divided into the categories described below:

- Kernel : classes that describe objects, messages, and other classes
- Data Types: classes that describe abstract data types, including points, dimensions , collections of objects, and lists.
- Windows : classes that provide access to most of the facilities of the window system.
- Dialog : Classes that allow the user to directly communicate with an application program

by using various kinds of menus, buttons, and keyboard access.

. Text : classes that provide simple text manipulation tools for loading, editing, and saving text.

. Graphics : classes that describe both primitive graphical objects(lines, boxes, circles, text, bitmaps,etc) and compounds of primitive graphical objects. The tasks that an object can perform are called the object's behaviors. When an object is sent a message, the object then invokes the behavior, and is performed.

4.2.3 Message

The main predicates in Prowindows are the following:

. new : create new objects

. send : set values or cause actions

. get : retrieve a description or value from an object

. get-ref : retrieve the object reference of a value from another object

. object/1 : determine whether an object currently exist

. object/2 : unify an object's description with a Prolog term

An object is sent a message using the Prowindows predicates send/2 or send/3 in order to request that the object exhibit a behavior.

. send(object, behavior)

. send(object, behavior, value)

Each message contains certain attribute values:

. Object : The object attribute of a message specifies the object that is to receive the message.

. Name : The name attribute of a message defines which behavior the message is to invoke.

. Value : The value attribute of a message provides the value or values needed by the object to carry out the requested behavior.

4.3 Object-oriented and Window-based User Interface

The window-based user interface of the BFI system is written in ProWindows which is designed from an object-oriented perspective. Automatically, the window-based user interface of the BFI system should be operated in the object-oriented method which is well-suited to a KBS. The Figure 3 shows the outline of the object-oriented approach of the window-based user interface in the BFI system.

The KBS sends messages to invoke a particular ProWindows object during the inference process. The messages are encapsulated in the ProWindows through the window-based user interface which invokes window objects on the basis of SunWindow. The format of the window(size, location, color, content, and so on) is decided by the messages in the KBS. In the object-oriented model, the user communicates with the KBS and applications through the display, which is represented in the windows on the screen. Each window object has its visual representation and functional role. Thus, the knowledge-based system can abstract the window displays with the window-based user interface as objects. The window objects provide a means of information and category of various kinds of windows.

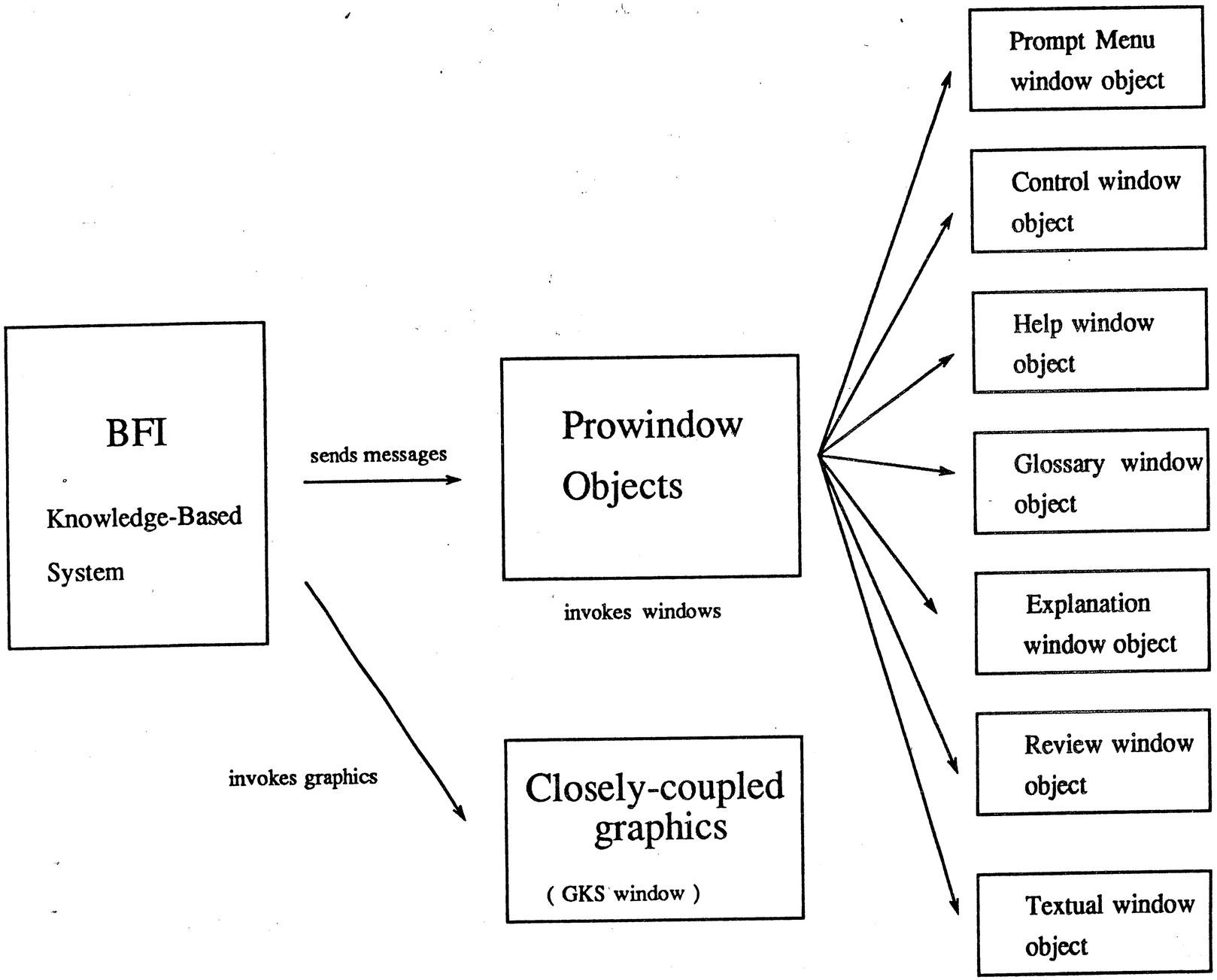

Fig. 4 Outline of the Object-oriented and Window-based User Interface for the BFI Knowledge-based System

CHAPTER 5

APPLICATION of a WINDOW-BASED USER INTERFACE to a KBS

5.1 Role of Window-Based user interface in a KBS

The communication between a user and a knowledge-based system can be divided into two channels. There is an implicit communication channel and an explicit communication channel.

. Explicit Channel : A bitmapped screen with multiple windows, menus, a mouse(pointing device) in the powerful workstation widens the explicit channel between the user and the KBS. These technologies are necessary but by no means guarantee a good window-based user interface. Exploiting these technologies to benefit the user requires an understanding of a workstation and the psychological principle, including nature of user's ability.

. Implicit Channel : When communication is based on shared knowledge structures, it is not necessary to exchange all information between the user and the KBS explicitly.

An effective window-based user interface on the explicit channel is more than just creating an attractive display. The window-based user interface should have a considerable body of knowledge about the user as a nice communication process. The complexity of today's software has made better communication techniques a necessity. And the use of the complex knowledge-based systems today is limited severely by the communication bottleneck in the narrow channel without good user interfaces between users and programs. A good user interface is definitely

necessary in a knowledge-based system. A knowledge-based system is more useful if the user interface supports rich and elaborate interactive graphical facilities which is using the window-based UIMS on the powerful workstations with, a bitmapped screen. Therefore, the window-based user interface on the workstation is very effective method for exposing the KBS's reasoning and knowledge captured to the user.

5.2 Implementation of the Window-based UI on the BFI KBS

5.2.1 The System Architecture

The BFI system is developed and implemented on the SUN workstation in 'C', 'Prolog', and 'ProWindows' under the operating system UNIX, providing a set of modules and a communication layer for the combined interfaces. Fig.5 shows a model supporting the window-based user interface, in combination with the declarative programming Prolog and ProWindows. ProWindows provides the user with a set of tools for textual and graphical as well as direct manipulative input and output. It provides a concise and safe communication and synchronization based on UNIX and tailored for Sun workstations with bitmapped displays, and the available windowing primitives.

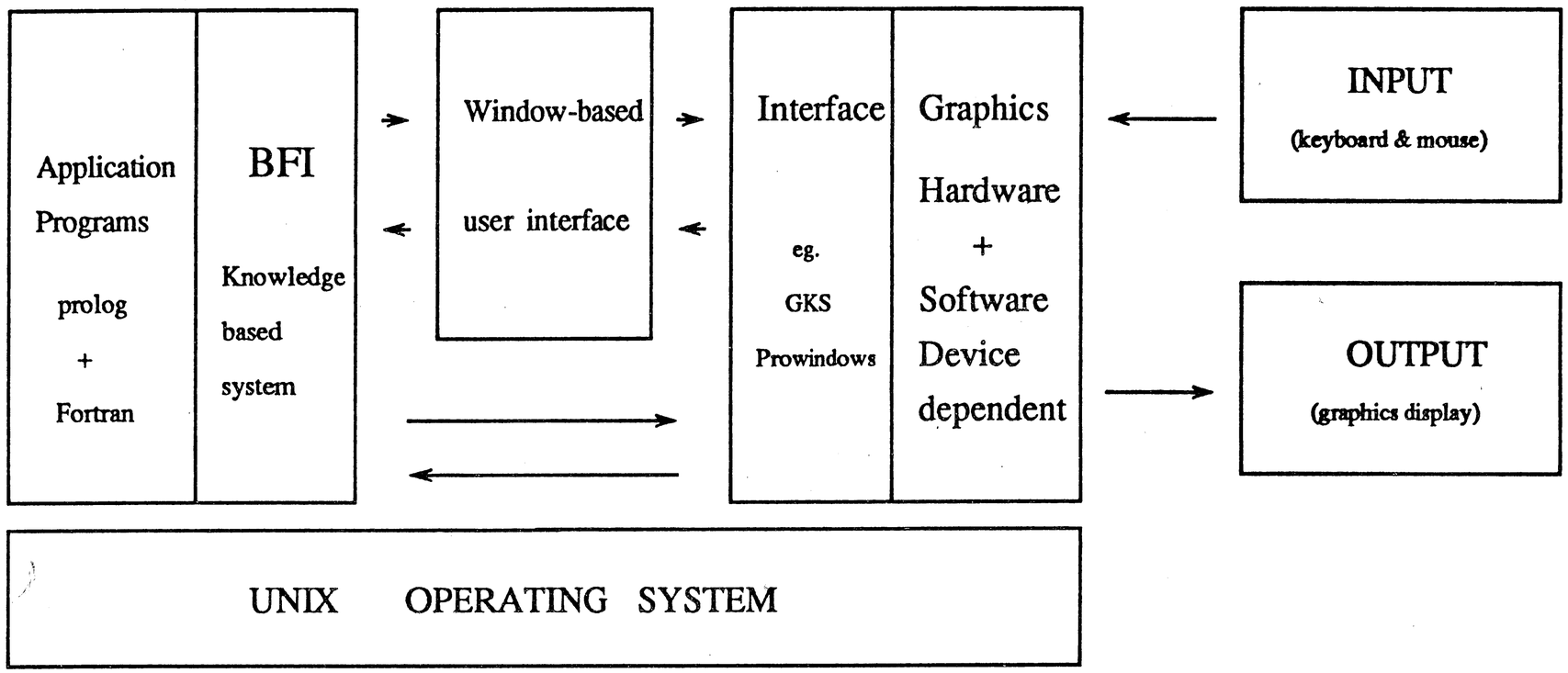

Fig. 5 Overall System Architecture for Window-based User Interface

5.2.2 Design Criteria

The design is based on the following ideas.

(1) Using an Existent Environment

The application is running on the Unix operating system and communicates with the user through the window-based UIMS which is implemented by ProWindows and SunTools. The use of a standard at the operating system level as well as in the graphical interface call allows for both independent development and relatively fast adaptation to the hardware. Therefore, it is not necessary to re-implement low level graphic elements which are available in the workstation's software.

(2) Data Separation

The logic part of the knowledge-based system does not have to know about nor manage both the interface programs and the screen oriented parts of the interaction, and vice versa. Applications do not have to know about the layout of tests, but should be able to initiate or destroy a process for display. Therefore, only the interface program which is capable of displaying messages has to read and display them on the screen.

(3) Communication Layer

Processes running in parallel can be independent from each other or cooperate with each other. Since there are at least two processes running (the application and the process for the interface programs), a bi-directional communication facility is necessary and available. This allows independent use of modules for different tasks using independent processes(windows).

(4) Code Splitting

The application program(Prolog) declares the logic of the expert system and other applications, whereas the sequential graphical and textual input and output is handled and

managed by the window-based UIMS which is implemented in C and ProWindows. A major problem in the implementation of interactive software is the internal communication, synchronization, and compatibility of the different programs. A C-Prolog binding mechanism based on Unix pipes forms the basis of the direct manipulative interface developments. The application and interaction modules can be plugged into the communication layer.

5.2.3 Advantages of the structure

There are several advantages for the system design in figure 5.

(1) The programmer can use the advantages of the direct manipulative interfaces without understanding the underlying window-based UIMS because of the data separation, code splitting, and the communication layer.

(2) Interfaces are adaptable to users' needs. AI programmers can easily design the user interfaces between applications and the UIMS to match the users' requirements.

(3) Users and the programmer can simply understand and modify the user interface without knowing the complex UIMS.

(4) Applications can receive the correct input. Because the user interface is at a higher level than the UIMS and GKS, it makes the user interface stable and reliable.

(5) Code splitting and data separation make it possible for the programmer to divide his problem and implement each part in the most suitable programming environment.

(6) Layered software structures are possible.

5.3 The Description of the Windows in the BFI System

5.3.1 Control of windows

The knowledge-based system in a window-based user interface can overlap the windows on the screen to expand the domain of its display. But the screen of the computer is a resource with very definite constraints. If the windows are numerous and the screen size is small, the user may become confused and spend most of his time attempting to searching for the windows. This situation is almost similar to the paging algorithms for the virtual memory algorithms of operating systems. If the number of pages in the program's working set is not greater than the number of pages actually available for the program to run, the program will run efficiently. But, in the reverse case, the program will spend most of its time reading and writing pages back and forth from main memory to the secondary memory device.

In the normal case, the user has the option to remove windows or keep them open. When the user destroy unnecessary windows not to be confused, he can remember the content of the destroyed windows by trading his own memory load for the windows as an external memory. It can unfortunately occur that the user destroyes the necessary windows. In the application of the BFI system, the KBS automatically controls the windows by sending the message to the window-based user interface in the following way.

(1) Destroy the useless windows

(2) Change the windows to icons which will be used later

(3) Rearrange the windows during the inference

The user does not need to exert any effort in removing or repositioning windows without destroying the necessary windows. This allows the user to concentrate on his own work.

5.3.2 Windows Type

The window-based UIMS provides various primitives callable in the application via the window-based user interface. Typical primitives in the window-based UIMS are the followings:

. text output

. graphical output

. menus and buttons as selectable regions

. other activatable regions for selection and control

. sound, etc

Applications can use these primitives to communicate friendly with users. A window may contain various primitives to show users compressed information.

Applications invoke several kinds of windows by sending the messages to the window-based user interface. The type of window is decided by the application or knowledge-based system. We can classify the windows into the seven groups according to the role in the BFI system.

(1) prompt menu windows

(2) help window

(3) glossary window

(4) explanation window

(5) review window

(6) graphical window

(7) error window

(8) control window

A window display is divided into several sections which have various functions: input and output, and control. Each section is a group with the same function that is composed of various primitives to satisfy effectively the users' requirements. An example of the window-based user interface is presented in figure 6.

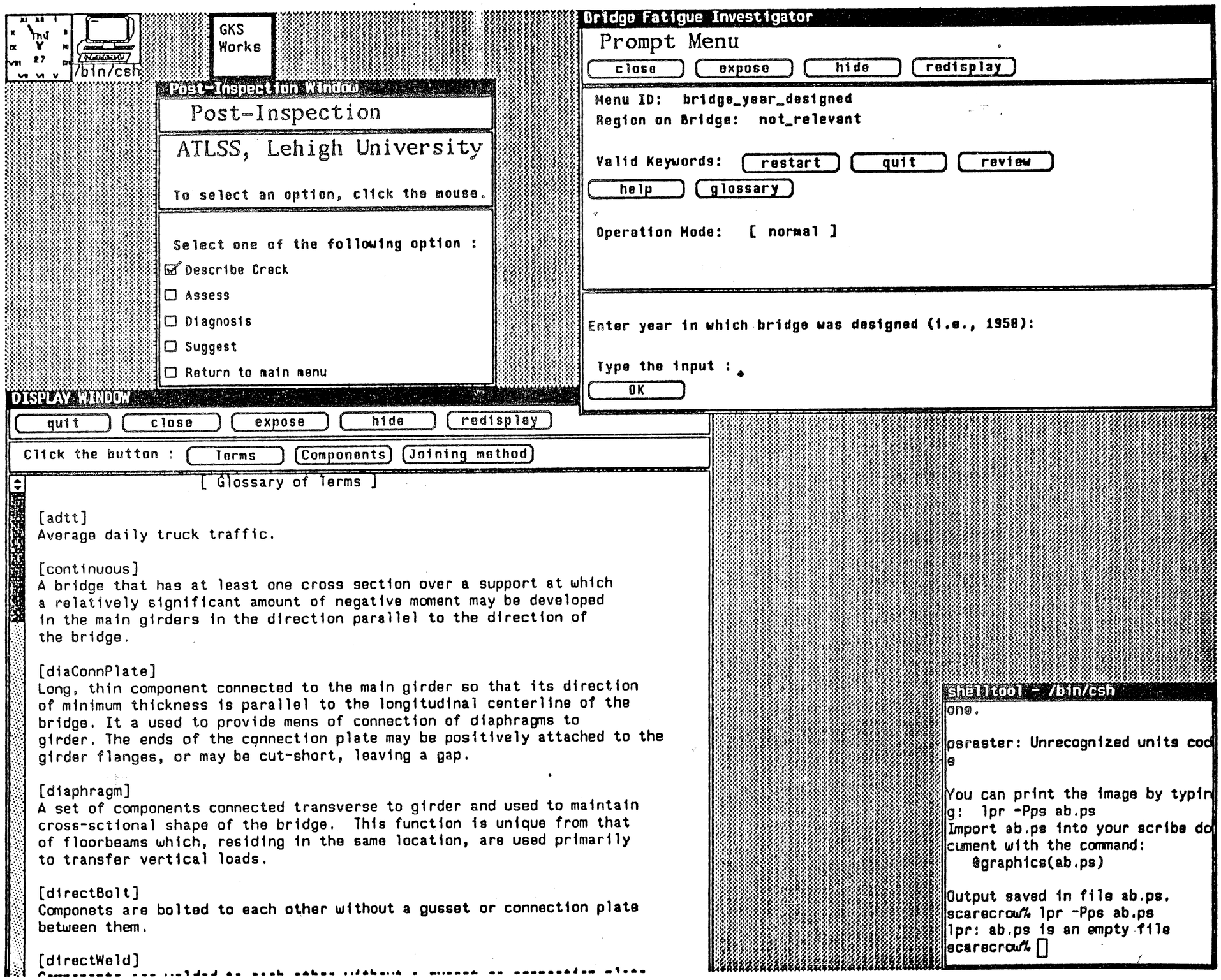

Fig. 6 An example of the Window-based User Interface in BFI system

5.4 Contributions of the Window-based User Interface

The effectual window-based user interface can contribute to the users by implementing it in applications on the high- performance bitmapped workstation.

(1) Efficiency

A window-based user interface maximizes the efficiency of the interface between users and applications. Because it provides various primitives on the screen and multiple windows, the user interface allows the user to observe and easily control the behavior of the application.

(2) Simplicity and Reliability

The window-based user interface uses a graphical tool of an UIMS and provides an effective method for displaying and viewing. Therefore, window-based user interface makes the conversations between user and complex applications more simple and successful.

(3) Apprehension

Windows have a function similar to the external memory and make the users recognize quickly the meaning of the content on the display by shortening the access time into internal memory. The users may quickly understand the knowledge of the KBS and operate and control rapidly the application through the multiple windows.

CHAPTER 6

SUMMARY

Efficient and friendly user interfaces are important for both the users and the programmers. Well-structured user interfaces make it possible for the programmer to modify the environment to be suitable for any type of user. Good user interfaces maximize expressiveness, understanding, and efficiency of the interface between users and the application.

In order to design a good user interface, the designer should consider the psychological factor which has three components: user, task, and system. In the case of the BFI system, a window-based user interface is developed and implemented on the popular and powerful SUN Workstation which provides high performance bitmapping. Window-based user interfaces provide a powerful means for displaying knowledge and reasoning in large knowledge-based systems. Experts and novices merely have to use a pointing device, such as a mouse, to open up the realm of a particular domain.

The window-based user interface that was implemented in the BFI system was tested on various users. The tests show that a window-based user interface is a useful means for communicating with the users and also explaining and manipulating the applications.

REFERENCES

[1] Andrew Monk, "Fundamentals of Human-Computer Interaction", Academic Press, 1984

[2] F. Klix and H. Wandke, " Man-Computer Interaction Research ", North-Holland, 1986

[3] Stuart K. Card and Thomas P. Moran, and Allen Newell, " The psychology of Human-Computer Interaction ", Lawrence Erlbaum Associates, Inc., 1983

[4] F. R. A. Hopgood and D. A. Duce and E.V.C. Fielding and K. Robinson, and A. S. Williams, " Methodology of Window Management", Spring-Verlag, 1986

[5] Gunther E. Pfaff, " User Interface Management System ", Spring-verlag, 1985

[6] Brad A. Myers, "Creating User Interfaces by Demonstration", Perspectives in Computing,1988

[7] M.D. Harrison and A.F. Monk, " People and Computer(designing for Usability)", Cambridge University Press, 1986

[8] M.D. Harrison and A.F. Monk, " People and Computer(designing the interface)", Cambridge University Press, 1985

[9] Yannis Vassiliou, "Human Factors and Interactive Computer System", Ablex Publishing Corporation, 1982

[10] Bruce Christie, "Human Factors of the User-System Interaction", North-Holland, 1985

APPENDIX A

What is the BFI?

Essential to ensuring the safety of bridges is the detection and assessment of potential and actual structural distress. Fatigue has been observed to be a major cause of structural distress in steel bridges. Looking for the resulting cracks and deciding what to do about them are important tasks.

The Bridge Fatigue Investigator(BFI) is a knowledge-based expert system that addresses a major infrastructure problem: the maintenance of America's immense inventory of existing bridges. The specific purpose of BFI is to assist a bridge engineer in inspecting for fatigue damage in steel girder bridges, and evaluating such structures for their susceptibility to fatigue and fracture problems. BFI can be used in two situations: pre-inspection and post-inspection.

. Pre-inspection : The major goal of the pre-inspectiion usage of BFI is to bring such expert knowledge to bear in the challenge of the inspection for cracks. Using the input data of the bridge by the inspector, along with information contained in its knowledge base, BFI identifies those connection details on the given bridge which are most susceptible to fatigue distress.

. Post-inspection : If a crack is observed during the inspection, BFI prompts the user for a description of the location, orientation, and size of the crack. BFI also provides advice in diagnosing its cause, assessing its seriousness, and suggesting what ought to be done about it. BFI performs the assessment by reasoning about the crack propagation, both qualitatively and quantitatively.

APPENDIX B

How to use the window-based user interface

The application can invoke and control many kinds of windows by sending messages to the window-based user interface. The application calls the procedures in the window-based user interface in order to open the necessary windows. The arity number and content of argument are different according to the window type. The below shows the procedure names with the arity number and window name invoked.

. window_main / 3 : prompt menu window

. window_help / 3 : help window

. window_glossary / 3 : glossary window

. window_review / 4 : review window

. show_window1 / 4 : explanation window

. window_error / 1 : error window

. welcome_window / 0, window_main3 / 3, window_query /2 , etc : control window

. etc

For example, the message for the prompt menu window has three arguments by using square brackets at the application.

. The first arity contains the calling window object and title and window position, and window size.

. The second and third arities contains the various primitives according to the application's requirements.

The application can open, control, and destroy the invoked window during the inference.

Below is an example of the message calling the prompt menu window of the BFI system.

```
window_main([ @object,' Bridge Fatigue Investigator',' Prompt Menu',
                                point(540,0),size(600,185) ],

        [    [ label(' Menu ID:    '),below, []   ],
             [ label(Menu), right, []   ],
             [ label(' Region on Bridge:   '), below,  []   ],
             [ label( Mom ), right, [] ],
             [ label(CC),below, []    ],
             [ spe_label,_,C],
             [ label(KK),below, []   ],
             [ spe2_button,State,K ],
             [ label(TT), below, []   ],
             [ spe_label, right, T ],
             [ label('  Operation  Mode:       ',below, [] ],
             [ label(Key), right,  []   ]
    ],
    [
         [ label(QQ),below, [ ]   ],
         [ label('  '), below, []   ],
         [ opt_label,Roption,R ],
         [label(RR),below, []   ]
    ]).
```

VITA

The author, son of ByungYul Baek and KyungSook Park, was born in Seoul, Korea March 3, 1959. He received a Bachelor of Engineering degree in electrical engineering from YonSei University in 1985. He worked at GoldStar Tele-Electrical Co., Ltd. as engineer. He studied at Lehigh University for his Master of Science degree in computer science. He is working the VFC INC./LU Ben Franklin project under professor Hillman. His research area is Expert System for the site characterization.