Some Thoughts on the Interface Between User Interface Management Systems and Application Software

J. Sibert, R. Belliardi, and A. Kamran

The George Washington University, Department of Electrical Engineering and Computer Science, Washington, DC, USA

INTRODUCTION

This paper summarizes the keynote presentation in the topical area of the relationship between user interface management systems (UIMS) and application software. We were asked to prepare this presentation because of our implementation experiences in a UIMS environment. Accordingly we will focus on our experiences during our implementation effort. But first, we will begin by establishing a working definition of a user interface management system. We will then discuss some desirable capabilities of UIMSs in general, and, finally, the interface to application software in the context of a sample application.

Working Definition of a User Interface Management System (UIMS)

Before we can discuss our main topic, we must begin by accepting a working definition of a UIMS. We feel that a UIMS represents a higher level of abstraction than, for example, a graphics subroutine package. The end user of an application does not see the UIMS, but only the user interface. The UIMS manages all aspects of this user interface. By this, we mean not only the interactive dialog between the user and the computer, but also all of the output, including application specific output. This last assumption is somewhat radical, but we consider output, including semantic output, to be inseparable, from the user's point of view, from the interface. Therefore, a true user interface management system must manage the output as well as the interactive dialog. Another aspect of a UIMS is that it should be possible to change the user's view of the application without changing any application code. In fact, the user interface should be reconfigurable and variable without resort to coding at all. Finally, we feel that a user interface management system is an integrated software tool. This implies integration between the dialog specification tools and the run time support environment. A library of subroutines cannot be a UIMS.

Some Desirable Characteristics of a UIMS

As a result of our experience with user interface management systems (Kamran and Feldman (82)) and in the design and implementation of user interfaces, (Sibert (83)) as well as the experience of others as reported in the literature, we feel that a good UIMS should exhibit certain characteristics. The
characteristics are summarized in Table 1 with special emphasis given to those for which the interface with application software is an important issue.

A UIMS should support a sufficiently robust set of data types to allow the specification of at least the following generic operations: (1) The user should be able to specify a position either absolutely or relatively, and (2) He should be able to specify quantities either as integers or real numbers and be able to specify a character string.

The user interface management system should also support a variety of selection options including objects from a display and selecting items from a menu. These selection types become particularly troublesome in two situations. The first occurs when it is necessary to select multiple items from the same set without knowing in advance how many items must be selected. The second is when the contents and size of the set are not known until run time. We call this run time enumeration. In terms of data types, the only issue which involves application software in a significant way is that of run time enumeration of selection items.

A UIMS should be capable of providing immediate feedback at all three levels of the dialog: lexical, syntactic, and semantic. Of these, only the semantic level implies a direct connection with the application software. However, it is important to note that under certain circumstances all three levels of feedback can be represented simultaneously. For example, consider disposing of a document in the Apple Lisa user desktop environment. The document is selected and an icon representing the document is dragged to a garbage can. The action of putting the icon in the garbage can represents lexical feedback by constantly showing a cursor location representing mouse movements, syntactic feedback by changing an object's position on the screen, and semantic feedback by inserting the object in the garbage can indicating that the object has been deleted.

A good UIMS should be able to support a wide variety of interaction styles, depending upon physical device availability and characteristics of the user such as relative level of sophistication for the task, left or right handedness, etc. This, of course, is one of the major motivations for developing the UIMS concept in the first place. The UIMS should be able to produce context-sensitive messages. Specific examples of these messages are error or help messages. Good user interface design principles tell us that an error message or a help message that is not context-sensitive will frequently give either too much or too little information to the user, only furthering his confusion. Obviously, for messages to be context-sensitive, the UIMS must communicate in some way with the application software. We also feel that the UIMS should support the simultaneous availability of input devices. This implies that it should be possible to configure a user interface in such a way that the user can at his option use a mouse graphics tablet, keyboard, or programmed function key to generate the same semantically and syntactically correct command.