Issues Pertaining to the Design of a User Interface Management System

A. Kamran

Institute of Information Science and Technology, Department of Electrical Engineering and Computer Science, School of Engineering and Applied Science, The George Washington University, Washington, DC, USA

INTRODUCTION

We reported at the GIIT Workshop our experiences in designing and implementing a prototype Interaction Handling Package called the Abstract Interaction Handler (AIH). In keeping with the terminology that has evolved since that Workshop, this paper will refer to the AIH as a User Interface Management System (UIMS). This paper reports our experiences and realizations beyond the Seattle Workshop [GIIT83]. We have been proceeding in two directions.

1) We have been implementing Interactive programs on top of our Interaction Handling package for the purpose of testing the usefulness of our model of a UIMS as a tool for designing good interaction.

Building tools for Interaction Dialog design as an application of the Interaction Language has shed light on:

(a) The suitability and power of the Interaction Language (as defined in [KAMR83])

(b) The functional separation of the application part and the interaction part of an interactive program.

2) Through experience with our UIMS, we have been constantly involved in addressing the larger issues -- those of defining and understanding more clearly, the domain of Interaction.

In addressing these issues, we have a better understanding of the interface between the Interaction Handling part (UIMS) and the set of semantic modules (application). Also, through the implementation of Interaction Techniques [KAMR83], we understand more clearly the interface between the graphics package and the Interaction Handler.

1. Problems with Basing the UIMS - Application Interface on the Linguistic Analysis Model

A grouping of actions in terms of the task types (data types) of the Interaction Language (IL) results in the expression of a command. A combination of one or more commands realize a user-level task. For instance, a user's desire to rearrange a room in a room layout application will be accomplished through a number of tasks of moving furniture around. These tasks will be mapped onto commands in the Interaction Language. Let us consider a sample command: "MOVE OBJECT FROM TO." This command can be represented in the current Augmented Transition Network (ATN) model of the IL as illustrated in Figure 1.

If we make the separation between the UIMS and the application based directly on the linguistic (semantic - syntactic - lexical) analysis model [POLB82], giving only semantic control to the application modules, then the whole process represented
Figure 1. Representation of the command "MOVE OBJECT FROM TO" in current ATN model of the IL.

in Figure 1 would be in the domain of the UIMS. Only at the completion of the command (which the UIMS should be able to recognize) would the control be given to the appropriate application module for the purpose of realizing the specified command.

In interactive programming, a number of factors have to be considered in applying the linguistic analysis model as the basis of separation of functionality. One of the main factors to be considered is feedback. In most cases, immediate feedback is desired at the lexical, syntactic and semantic levels while the command is still being formulated. Let's consider the example command of Figure 1. It is desirable that the user receive immediate feedback, even as the command is being parsed. Assume that the user is using a keyboard to type in the command. Each letter of the alphabet is echoed as the user combines these lexemes to form the first word of the vocabulary "MOVE." The parse for the recognition of this word is part of syntax. If, for instance the user types in a word that is not in the vocabulary, it is desirable that the Interactive System give immediate feedback rather than wait for the whole command to be typed in. At least it should have the capability to handle immediate feedback. At times, a user might be able to avoid continuous syntactic feedback in the sense mentioned above. For instance, if the user is using a keyboard to type in the example command of Figure 1, the system can work in a mode where syntactic parsing begins only when the user indicates a completion of the command specification (e.g., by pressing the RETURN key). Having continuous feedback is generally dependent on the interaction device available as well as the style of interaction. Thus, there are two ways of analyzing the specification of a command like "MOVE OBJECT" as illustrated in Figure 2.

Figure 2. Discrete feedback model and continuous feedback model of the command "MOVE OBJECT."