TOWARDS UNIFORM USER INTERFACES IN AN OFFICE AUTOMATION SYSTEM

Ronald L. Danielson

Department of Electrical Engineering and Computer Science
University of Santa Clara, California

ABSTRACT

The facilities provided by an office automation system must be efficiently usable by a wide variety of different user groups, each of which has definable characteristics with respect to skills, which should be exploited by, and expectations, which should be satisfied by, the user-system interface. There are three basic dialog formats which may be combined to implement this interaction: menu, command, and softkey. An examination of the respective characteristics of dialog formats and user classes leads to the conclusion that a softkey interface best satisfies the needs of the widest range of users and thus should form the basis for most of the interaction. Examples are given of how uniformity of interaction of existing office utility programs with menu and command interfaces may be improved by modifying those programs to use softkeys. The modifications may be done in a way which allows all classes of users to shift from the existing interface to the new interface without retraining.

RÉSUMÉ

Les ressources offertes par un système de bureautique doivent pouvoir être utilisées de façon efficace par une variété de classes d'utilisateurs différentes qui ont des caractéristiques identifiables du point de vue des compétences, que l'interface utilisateur-système doit exploiter, et des attentes auxquelles il doit satisfaire. Il existe trois modes de dialogue de base que l'on peut combiner pour mettre en oeuvre l'interaction: menu, commande et touche personnalisable. Suite à une étude des caractéristiques respectives des modes de dialogue et des classes d'utilisateurs on conclut qu'une interface à touche personnalisable répond le mieux aux besoins de la plus grande gamme d'utilisateurs et qu'elle devrait donc servir de base à la plus grande partie du dialogue.

On donne des exemples sur la façon d'améliorer l'uniformité de dialogue des programmes utilitaires de bureau existants, munis d'interfaces de mode menu et de commande, en les convertissant à une interface à touche personnalisable. Les modifications peuvent être faites de façon à permettre à toutes les classes d'utilisateurs de passer de l'interface existante à une nouvelle interface sans qu'un recyclage soit nécessaire.

1. Office Automation and User Interfaces

It is generally agreed that development of office information systems (OIS's) has the potential to greatly increase the productivity of office workers at all levels [1, 2, 3]. It is also agreed [2, 4] that to realize this potential, an OIS must be integrated:

- at the program level, to insure efficient computer processing;
- at the functional level, so that all necessary office activities may be performed; and
- at the user interface level, for without a smooth, functional interface an OIS may not be used at all.

Hayes, et. al. [5], gives a striking example of the typical frustrations of the human-machine interface, and a discussion of research activities aimed at improving that interface. This paper, by contrast, is concerned with improving the uniformity of the user interface in current OIS's, through software techniques which are well-understood and using hardware which is commonly and inexpensively available.

Each user (or user community) can be characterized with respect to skills and expectations, which should be exploited and satisfied (respectively) by the interface. The interface designer must match these characteristics with the characteristics of the particular mode of interaction, within the hardware/software context in which the implementation occurs.

The end goal is to make the user/system interface as uniform as possible across the whole range of functions offered by the OIS. This provides the user with a familiar environment during all interaction with the system, making it easier to learn new functions, and to shift between use of different functions. Schneiderman [6] summarizes a number of design considerations for interactive systems.

One aspect of an OIS which is somewhat unique is the wide range of functions provided by (what the user should view as) a single system. This means that even a user who spends a lot of time on the system may use some of the functions only rarely. It is thus important to emphasize that each user learns different aspects of a system at different rates [7], and goes through several phases while learning any particular aspect [8]. This means that the interface must accommodate changes in characteristics for a single user across different OIS functions, as well as for different users within each function, to be effective.

2. Characterizing Users

Since it is obviously impossible to characterize individual users, one is first faced with the problem of identifying classes of users and then characterizing typical members of each class. De Blasis [9] suggests a categorization based on the user's role in the office: manager, principal, secretary, clerical, etc. The problem with such a classification is the wide differences in abilities which can exist between two individuals who fill the same role in different offices.

Cuff [10] discusses the characteristics of "casual users," classifying individuals as casual or not based on considerations such as frequency of use, level of skill, and familiarity with computer concepts. A categorization based on these more general considerations would be applicable to any office, and would permit a single user to fit into different categories depending on the office function being performed.

Figure 1 summarizes the characteristics of four different user classes, determined by their knowledge of computers (naive or sophisticated) and how often they use the system (frequently or occasionally). Members of each category are described in terms of their:

- initial knowledge of the system and ability to learn during a session,
- frequency of error and level of correction needed,
- desired pace of interaction,
- need for context, and
- ability to handle complexity.

Naive and occasional (N-O) users need as much handholding as the system can provide. They don't know what the system expects of them, nor have they experience on which to base guesses. They expect to make errors, but need a "safety net" to help them recover when they do. A minimum complexity, slow-paced dialog is desired. Since they are only occasional users, they are unwilling to invest much time (certainly less than an hour) in training, and thus will have only a general idea of what the system can or will do. An example of a prototypical N-O user would be a personnel manager

	Naive	Sophisticated
Occasional	little initial knowledge of system, little learning during session; uses HELP often	little initial knowledge of system, good learning during session; relates to other systems known; infrequent use of HELP
	expects to make errors; detailed description of error and possible corrections needed	expects to make errors; summarized description of error and possible corrections helpful
	prefers slow-paced interaction; not troubled by slow output	wants quick response once input is made; faster output desired, especially as session runs on
	needs explicit context to be able to decide on action to perform	explicit context reduces errors; can function without it
	needs minimum complexity of interaction	doesn't know system well enough to use complex dialog capability
Frequent	good initial knowledge of system; infrequent use of HELP	good initial knowledge of system; almost no use of HELP
	surprised at errors; summarized description of error needed, no description of possible corrections necessary	surprised at errors; usually no description of error needed
	wants fast interaction, fast output	demands rapid interaction
	no explicit context needed	no explicit context needed
	prefers to minimize complexity	complexity no problem

Matrix of User Characteristics
Figure 1

using a filing system to retrieve and print a document.

Sophisticated and occasional (S-O) users, by contrast, are able to learn quite a bit about a system during a session, although they forget much of that knowledge between sessions. Typically, they relate this system's actions to those of similar systems with which they are familiar. They also expect to make errors, but recover with much less assistance. A complex dialog would not be a problem, but they usually will never learn the system well enough to use the complexity. A typical S-O user might be a programmer trying to schedule a meeting using an on-line calendar system.

Naive and frequent (N-F) users are very familiar with the system, probably through some formal training period as well as constant practice. Despite their familiarity, they prefer to minimize the complexity of the dialog by, for example, restricting the set of system functions they will employ [11]. They want rapid interaction and minimal system prompts, since they probably don't have to read the prompts to decide what to do next (they have "internalized" the dialog context). A secretary doing text processing via a familiar editor is an example of an N-F user.

The sophisticated and frequent (S-F) user needs little help from the system, demanding as fast and functional an interface as possible; excessive prompting by the system is a definite hindrance to the user/system dialog. They don't expect to make errors, but when errors occur a simple notification, without explanation, is usually enough to allow correction. Most S-F users will have trained themselves through adventurous trial-and-error experimentation. A programmer using her favorite editor would fall into the S-F category.

One characteristic which is common to members of all four categories is poor typing ability. Even many people who must type frequently do not type well, and typing difficulties are compounded if special character keys must be used.

3. Characterizing Interface Modes

As with user characteristics, there are a number of different ways to categorize modes of interaction. For example, Martin [12] lists 23 different interface techniques for alphanumeric displays alone. This is much too fine a classification for our purposes. In addition, we have explicitly excluded expensive or unusual media (voice output, speech recognition, eye motion, and pointing devices such as light pen or mouse) from consideration, even though they may significantly enhance the interface.

Therefore, we will consider three broad classes of interface. Each of the three classes will use text display as a common output medium, and are thus differentiated by the type of input. The classes are: menu, command, and softkey.

3.1. Menu Interface Considerations

The term "menu" refers to a display screen which provides a context to aid the user with input. Menus either allow the user to select different actions ("branching" menus) or ask for information to be input ("data entry" menus). Branching menus sometimes contain one or two fields for data entry.

Usually, menus are connected as a tree. In general, a user moves down in the tree, selecting choices on branching menus and inserting information on data entry menus, until a desired action is accomplished. Data entry menus are usually followed in sequence, with no explicit user choice allowed.

Movement back up the tree after completion is not so straightforward. Typically, the user must back out menu-by-menu, until a menu is reached which allows following the next desired branch of the tree. This is relatively slow, and a user often desires to move as rapidly as possible to a known menu elsewhere in the tree.

As a minimum, the interface should facilitate such movement by providing capability from any menu to return to the immediately preceding menu, return to the last menu which offered a choice of branches, or return to the main menu for the function.

Displaying menus on the terminal screen can be a very slow process, particularly at low transmission bandwidths, which in turn drastically limits the pace of the dialog. Limiting the number of selections offered on any particular branching menu reduces the impact of this restriction, as does use of short, concise selection descriptions. This limitation of choices also agrees with the concept of memory chunking, and allows the user to determine the possible choices much more rapidly.

The number of keypresses needed to make a selection should be minimized (i.e., lower case rather than capital letters) to reduce the need for typing skills.

3.2. Command Interface Considerations

Command-driven interfaces require the user to input a string of characters representing an action to be performed. This approach typically allows a faster rate of interaction than the menu technique, due to the significant reduction in output from the system. However, a correspondingly greater burden is placed on the user to be aware of possible actions. This burden can be reduced by providing a context, in the form of a prompt to solicit input of a command. The pace of the dialog is slowed according to the length of the prompt.

In order to minimize memorization, the total number of commands within a function should be kept as small as possible. Frequently, one command word may invoke any of several related actions, depending on parameters entered by the user in response to prompts. Thus, the user enters a single command (a function to be performed, for example) and is prompted for the object of the command.

This eases the memorization problem for new or occasional users. Frequent users can enter the command string and required parameters all at once, which reduces or eliminates the need for prompting, and speeds up the interaction.

To make commands easier to remember, the command string should be a natural language word which has the same meaning TO THE TYPICAL USER as the corresponding command action. Words which have a meaning as computer jargon, but don't have the same meaning in everyday communication, should be avoided. In addition, the same command should be used for similar actions in different functions.

It is also necessary that the interface accept abbreviations as well as full commands, since typing long command strings quickly becomes

tedious. Abbreviations should have a uniform length (e.g., all three characters). If the system specified that any unique prefix of the command is an adequate abbreviation, the user must know all commands to determine what is unique. This is a major problem for occasional users. Abbreviations should also be formed in a uniform manner.

3.3. Softkey Interface Considerations

Softkeys are special keys whose function can be varied under program control, and which are typically grouped together in a separate area of the keyboard. The number of softkeys available can vary between about eight and 20.

They represent an intermediate approach between menu and command interfaces. The meaning currently attached to each of the keys may be displayed on the screen, providing a context in which to make a choice, and selections are indicated by a single keypress. Displaying the valid softkeys only requires writing a single line, which allows very rapid interaction. However, the limited number of keys sacrifices some of the flexibility of the command interface for greater ease of use.

Meaningful titles, which reflect the actions performed, should be chosen for the softkeys, similar to command string selection. Keys which are active within a given function are typically grouped together into "softkey sets," whose functions are displayed on the terminal screen. Only the keys within a set which represent meaningful actions in the current context should be displayed.

Multiple keysets within a function represent one means of circumventing the limited number of softkeys. The user simply scrolls through the available keysets until softkeys implementing the desired action are displayed, and then proceeds normally.

Another approach would be to use prompts and parameters to expand the capabilities of each key, as described for commands. The rate of interaction using such prompting may be increased by prompting for more than one input parameter with a single output. Note that this approach again places a premium on typing ability.

4. Matching User and Interface Characteristics

It should be clear that no single interface satisfies the requirements of all four categories of users. Menus provide excellent context to aid N-0 users, but the resulting slow display speed makes them impracticable for S-F users. Commands give the user almost complete control over the pace of the dialog, but require accurate, fast typing ability to obtain a high rate of interaction, and the amount of memorization required is intimidating for occasional users. Softkeys provide less context than menus, and allow less user control than commands, but they do yield a high rate of interaction and require an absolute minimum of typing to effect program control.

One might be tempted to specify three different interfaces: menus for N-0 users, softkeys for N-F and S-0 users, and commands for S-F users. However, such an arrangement, coupled with varying frequencies of use of the different functions within an OIS, would require users to shift back and forth between interface types as they changed functions.

If a preponderance of the anticipated users fell into one of the classes, the corresponding interface would be the obvious choice. If that is not the case, a softkey interface is the best compromise for all users, being nearly ideally suited to N-F and S-0 users, and more than acceptable to the other two classes.

Choosing softkeys as the principal mode of user interaction does not rule out the use of other modes, or softkeys in conjunction with other modes, for some applications.

For example, a function selected by softkey will frequently require several parameter values before execution. Prompting for these data is very tedious at slow transmission speeds. Use of a data entry menu is usually faster and clearer for input in a structured, unvarying manner.

5. Implementing Softkey Interfaces

If the OIS is being implemented as a new system, designing the softkey interface is no problem. However, many existing OIS's, particularly those running on a shared central computer, have been developed in a piecemeal fashion with different user interfaces. In this case, the new interface must be implemented so as to be minimally disrupting to the existing user community.

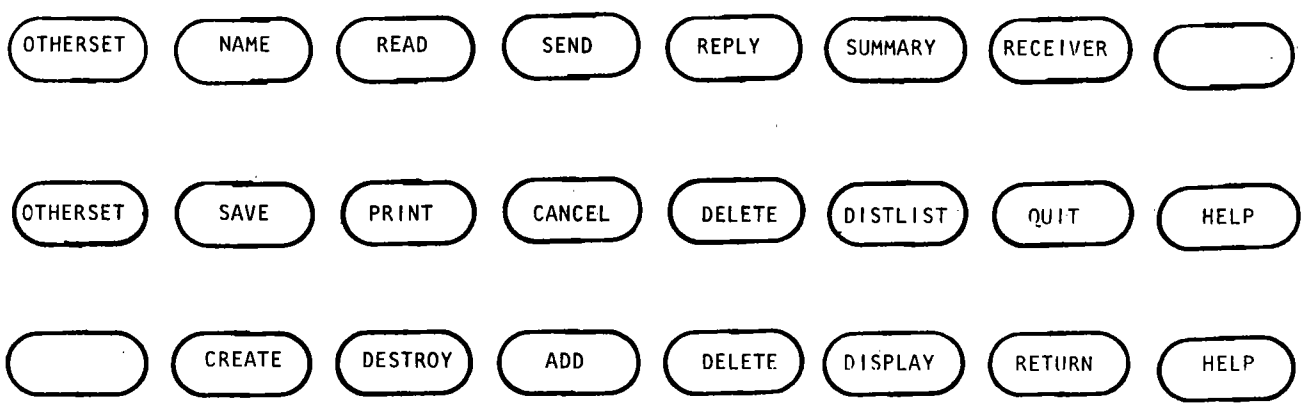
For command interfaces, the existing commands serve as the basis for the new softkey labels. The commands must be grouped into keysets containing related functions. The command string (or perhaps the abbreviation) would be the label for the corresponding softkey. This provides an immediate association with the previous

command, resulting in an almost effortless conversion. Note that this is not the time to correct inconsistencies in abbreviations or command mnemonics.

For example, Figure 2 contains the command list for a hypothetical electronic mail system. The original commands formed two groups, one for typical mail functions and a smaller group for maintenance of distribution lists. Assuming eight softkeys, the resulting three keysets are shown in Figure 3. Note that two additional keys had to be included: one to toggle between keysets in the main mail routine, and another to return to the main routine from the distribution list function.

```
CANCEL            = Erase message already sent
DELETE            = Erase message received
DISTRIBUTION LIST = Begin distribution list functions
HELP              = Access help material
NAME              = Input this user's mail name
PRINT             = Make a hardcopy of message
QUIT              = Leave the message program
READ              = Read incoming messages
RECEIVERS         = Access directory of valid recipients
REPLY             = Respond to message received
SAVE              = Copy message to disk file
SEND              = Send message
SUMMARY           = See synopsis of messages received
ADD               = Add name to existing distribution list
CREATE            = Create new distribution list
DELETE            = Delete name from existing distribution list
DESTROY           = Destroy existing distribution list
DISPLAY           = View names on existing distribution list
```

Hypothetical Message System Commands
Figure 2



Softkeys for Message System
Figure 3

The transition from menus to softkeys is somewhat more involved. For branching menus, the ideal choice is to have each selection correspond to one softkey. This may require that menus be broken up into more than one set of keys, depending on the number available. Labels for the keys should be chosen from keywords in the selection description.

A gradual transition might be desirable, in which case the softkeys which will be available should be emphasized by, for example, capitalizing the keywords with which the keys will be labeled. Intermenu movement could be implemented via softkeys, as well.

6. Summary

An integrated user interface design is important to the successful use of the various functions of an OIS. Considering the diversity of users an OIS must satisfy, and the different levels of familiarity any single user will have with different functions, an interface based on softkeys is the best compromise between speed of interaction (to satisfy experienced, demanding users) and support and guidance (needed by inexperienced, hesitant users). Existing menu and command interfaces can be modified to use softkeys, and greatly increase uniformity across separately developed functions, with a minimum of disruption of the existing user community.

7. Acknowledgements

The author would like to thank L. Hurtado-Sanchez for his comments on several topics in this paper, particularly user classification and characteristics. He would also like to thank R. Feirstein, R. Horowitz, M. Moy and A. Mueller for discussions about practical user interfaces.

8. References

1. Engelbart, D. C., "Towards integrated, evolutionary office automation systems," Proceedings of the 1978 Joint Engineering Management Conference (October, 1978) pp. 63 - 68.

2. Tsichritzis, D. C. and F. H. Lochovsky, "Office information systems: challenge for the 80's," Proceedings of the IEEE, 68:9 (September, 1980) pp. 1054 - 1059.

3. Rhodes, W. L., Jr., "How to boost your office productivity," Infosystems (August, 1980) pp. 38 - 42.

4. Ellis, C. A. and G. J. Nutt, "Office information systems and computer science," ACM Computing Surveys, 12:1 (March, 1980) pp. 27 - 60.

5. Hayes, P., E. Ball, and R. Reddy, "Breaking the man-machine communications barrier," Computer, 14:3 (March, 1980) pp. 10 - 30.

6. Schneiderman, B., "Human factors experiments in designing interactive systems," Computer, 12:12 (December, 1979) pp. 9 - 19.

7. Nickerson, R. S., "Man-computer interaction: a challenge for human factors research," IEEE Transactions on Man-Machine Systems, MMS-10 (December, 1969) pp. 164 - 180.

8. Bennett, J. L., "The user interface in interactive systems," in Annual Review of Information Science and Technology, Vol. 7, C. A. Cuadra, Ed., American Society for Information Science, Washington, D. C. (1972).

9. De Blasis, J. A., "An office automation perspective to information systems management," Proceedings of the Twelfth International Conference on System Sciences, Vol. II (January, 1979) pp. 40 - 49.

10. Cuff, R. N., "On casual users," International Journal of Man-Machine Studies, 12:2 (February, 1980) pp. 163 - 187.

11. Wimmer, K. E., "Research on human interface considerations for interactive text generation," Proceedings of the Fourth International Conference on Computer Communications: Evolutions in Computer Communications (September, 1978) pp. 727 - 732.

12. Martin, J., Design of Man-computer Dialogues, Prentice-Hall, Englewood Cliffs, New Jersey (1973).