CHI'85 Panel Discussion

Identifying and Designing Toward New User Expectations in a Prototype Text-Editor

Robert Mack
User Interface Institute
IBM Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, New York 10598

Introduction

Learning to use a text-editor can be difficult for novice users: extensive instruction is typically required and much trial-and-error (e.g., Mack, Lewis & Carroll, 1983; Seybold, 1979). How can we design an editor interface that requires much less training and minimizes user difficulties? This paper discusses the design and initial evaluation of an editor prototype which tries (a) to get novices started doing meaningful work relatively quickly (e.g., in a half hour) with no explicit step-by-step instruction, and (b) minimize serious problems as novices master basic text-editing operations. The approach taken to achieve these goals was to try to better accommodate empirically identified expectations on the part of novices about how editing operations should work, and avoid problematical design features. Two lines of research contributed to identifying expectations and problems, and hence specifying a more intuitive interface design.

Identifying Text-Editor Interface Problems

The first line of research involved "think aloud" studies of commercially available word processors. Computer naive office temporaries were asked to learn basic word processing skills using self-study manuals. We observed many problems both with the instructional materials and with the computer interface (see Lewis & Mack, 1982; Mack, Lewis & Carroll, 1983). One notable problem was that novices generalized from typewriting, and were not readily able to understand novel text-editing operations or possibilities.

For example, novices did not readily understand why familiar operations like Carrier Return modify text rather than simply move the typing point (see also Douglas & Moran, 1983). They also did not readily understand how reflowing of text that accompanies operations like inserting or deleting or reformatting was managed. And they did not readily understand such abstract concepts as blank areas of a document window where text entry is not permitted (so-called non-typing areas) or how familiar objects like blank lines are often represented by embedded (sometimes invisible) formatting symbols.

The second line of research tried to more directly probe possible novice expectations by staging demonstrations of text-editing activities and asking novices to (a) predict how to accomplish the goal behind the activity, and (b) describe what they thought they saw after the demonstration (see Mack, 1984). These observations point beyond expectations about typewriting, and towards a general expectation we might describe as *actions are simple*: i.e., a tendency to assume that an action is associated with one outcome. Operations that seemingly involve more than one outcome are assumed to require more actions than is typically the case. For example, there is evidence that participants analyzed multiple effects of operations like inserting or deleting into component actions like "make space" followed by typing, in the case of inserting, or erasing and "getting rid of gaps", in the case of deleting.

Designing the Editor Prototype

These empirical observations do not in themselves provide clear-cut functional specifications for a text-editing interface. However, they provide a basis for analyzing existing editor implementations and how these might be modified to more closely approximate novice expectations. For example, the observation that participants did not readily understand embedded format symbols, or the association of these symbols with familiar typewriter-like operations like Carrier Return led to the challenge to eliminate such symbols and find more concrete and direct ways for specifying and manipulating text objects (like new blank lines), and the properties of these text objects.

Similarly, the observation that automatic reflowing of text as it accompanies various activities like deleting or inserting or reformatting conflicts with the expectation that actions are simple and need to be explicitly managed led to the challenge to implement reflowing of text in its various guises as an explicitly managed operation, at least for novices.

The attempt to solve these problems led to the design of an editor prototype in which text-editing objects and actions refer to more familiar and concrete elements of the paper office, and users have more explicit control over operations involving reflowing. For example there are function keys that refer explicitly to concrete text objects like Word, Line, Paragraph and Page. These functions can be used in combination with actions like Delete, Insert, Format and Adjust to specify the scope of these editing and formatting actions. Inserting a line, for example, involves pressing the Line function followed by Insert. Similarly, to delete a line, one uses Line and Delete. Lines which contain gaps due to deletions, or paragraphs whose lines are rendered uneven from revision changes, can be explicitly reflowed using an Adjust key. One version of an Insert function allows users to make space and then type (vs. initiate an insert mode). These changes were intended to give users more explicit control over editing changes, in some cases matching what novices expect (e.g., "make space" to insert or "get rid of gaps" following erasures). Familiar typewriter operations like Carrier Return, Space and Backspace work similarly to their typewriter counterparts. And the prototype eliminated unfamiliar features like embedded formatting symbols or non-typing areas (within the margins of the document).

Evaluating the Editor Prototype

The editor prototype is being designed and evaluated iteratively. To date, two evaluations have been carried out. In both evaluations participants were given a half day to accomplish as many of seven letter typing tasks as possible, given no step-by-step instruction, but only a set of reference cards. The cards briefly define word processing, how to turn the computer on, the two general ways to use the word processor (select items from menus, press function keys) and provide general definitions of basic functions grouped according to typical text-editing tasks. Participants were encouraged to think aloud as they worked, and the experimenter occasionally asked questions about what the participants were thinking about.

In the first evaluation, six computer-naive office temporaries required about an hour to complete their first job which consisted of creating, typing, printing and finishing a simple one page memo (time to respond to experimenter questions is factored out of all times). Examination of the first two hours indicated that participants completed an average of four of the seven letter typing tasks, attempting in the process about 383 subtasks (i.e., goal-related tasks like create a new document, delete a word, adjust a paragraph, print a document, etc.). Participants experienced problems in about 20 percent of these subtasks (i.e., two or more attempts were needed to accomplish the goal, or they failed completely).

Based on problems identified in this evaluation, modifications and a second evaluation were undertaken. In this case, six participants were able to get started doing meaningful work within a half hour, again, with no explicit (step-by-step) instruction and, indeed, a reduced set of reference cards. In the first two hours, participants were able to accomplish four typing tasks, attempting a total of 486 subtasks for which about 20 percent led to problems (i.e., two or more attempts to accomplish a subtask).

Overall, both versions of the prototype avoided many of the problems observed in commercially available text-editors, involving familiar typewriter-like operations like Carrier Return or unfamiliar features like formatting symbols or non-typing areas. While overall performance was comparable for the two evaluations an examination of performance on, and problems with, specific types of operations indicate both cases of improvements in the second prototype version, relative to the first, but also cases where problems remain in finding intuitive implementations of more advanced text-editing operations. In particular, while participants do not seem to find it unusual that they need to explicitly manage reflowing, and did better doing so for the second iteration, key aspects of how reflowing operations are implemented still create problems for these users. These problems are being solved by further iterative evaluation and modification.

Key Design Approaches

Three design approaches are illustrated in this prototype research. First the prototyping effort has been driven by extensive qualitative information about problems novices having using editors, and insight into the causes of those problems. This included an adaptation of the verbal protocol technique to directly probe new user expectations.

Second, these expectations have driven an analytical process of identifying an interface design. to better accommodate these expectations. Think aloud protocols and evidence for expectations do not necessarily provide direct or unique specifications for a user interface, and other considerations entered into the prototype specifications. But novice expectations pointed towards two key interface design requirements: (a) user control over reflowing and (b) more concrete representation of text objects involved in operations.

Third, the prototype itself is being designed iteratively through user testing and modification aimed at achieving key behavioral goals of getting novices started doing meaningful work relatively quickly, and minimizing serious problems as they progress further into the editor's capabilities.

References

Douglas, S. and Moran, T. Learning text editor semantics by analogy. In *Proceedings CHI'83 Human Factors in Computing Systems* (Boston, December 12-15, 1983). ACM, New York.

Lewis, C. and Mack, R. Learning to use a text-editor: Evidence from "Thinking Aloud" protocols. *Proceedings of Human Factors in Computing Systems Conference,* National Bureau of Standards, Gaithersburg, Maryland, 1982.

Mack, R., Lewis, C. and Carroll, J. Learning to use word processors: Problems and prospects. *ACM Transactions in Office Information Systems*, 1(3), 1983, 254-271.

Mack, R. *Understanding text editing: Evidence from predictions and descriptions given by computer-naive people.* IBM Research Report RC 10333, IBM Thomas J. Watson Research Center, Yorktown Heights, New York, 1984.

Seybold, J. Training and support: Shifting the responsibility. *Seybold Report on Word Processing*. 1981, 4.