These lines are a beacon for a sort in an array, particularly if they occur inside program loops.

HOW ARE BEACONS USED?

According to Brooks, there are two stages in program comprehension: hypothesis and verification.

Hypothesis is conceptually driven. The comprehender generates an overall hypothesis about the program's function from skimpy sources, such as the program name or a brief description. This high-level hypothesis, combined with the programmer's knowledge and expectations, leads to the formation of subsidiary hypothesis about what major structures and operations ought to appear in the program.

Verification is both data-driven and conceptually driven. Now working with the program text, the comprehender tries to find support for the hypothesis. Verification is not done by studying the program line-by-line, but by looking for beacons. The role of beacons, then, is to link the conceptually driven hypothesis formation stage to the data of the program text.

WHAT WE ARE STUDYING

We are collecting empirical evidence of conceptually driven processing in program comprehension. In particular, we are interested in the existence and use of beacons.

OUR CURRENT WORK

Wiedenbeck has done two preliminary experiments, using a memorization and recall technique. The results support the existence and use of beacons in program comprehension.

EXPERIMENT ONE

Hypothesis: Beacons, or key features, exist as a focus for program comprehension. In a Shellsort program the beacon was assumed to be the swap lines, and we were interested in whether they were recalled better than other parts of the program.

Subjects: 12 novice and 12 experienced Pascal programmers.

Method: Subjects memorized the 23-line Shellsort program for three minutes and at the same time tried to figure out its function. They then recalled it verbatim for four minutes and also wrote down what they believed to be its function.

Results:

1. Program function: novices 67 percent correct; experienced 100 percent correct.
2. Lines recalled: Experts recalled significantly more than novices, an average of 11.5 vs. 6.5 lines. Overall, beacons were not recalled better than non-beacons. However, beacons were recalled significantly better by experts than were non-beacons e.g., 77.8 percent recall of beacons vs. 47.5 percent recall of non-beacons.

Beacons were not recalled better by novices e.g., 13.8 percent recall of beacons vs. 30.4 percent recall of non-beacons.

EXPERIMENT TWO

Hypothesis: Same as experiment one.

Subjects: 12 experienced Pascal programmers.

Method: A different group of subjects studied the Shellsort program for comprehension for four minutes but did not memorize it. Afterwards, seven index cards, each containing one program line, were given to the subjects. On each card they had to fill in the lines following the line given. They were told to complete the cards in order of most sure to least sure of correct recall. Correctness of recall and order of recall were recorded. One card contained the first line of the swap, which was assumed to be the beacon (`temp = a[j];`). The other cards contained random non-beacon lines.

Results:

1. Correctness of recall: Subjects recalled the beacon lines correctly 88 percent of the time. They recalled non-beacon lines correctly only 4-29 percent of the time.
2. Order of recall: The card containing the beacon line was recalled earliest in the recall period.

The experiments are preliminary. However, they suggest that: 1) beacons exist; 2) they are a focus for program comprehension for experienced programmers; 3) they do not seem to be a focus in comprehension for novices; 4) experienced programmers focus on beacons in different kinds of comprehensive tasks (i.e., when they must memorize a program and also when they are simply studying it for understanding).

REFERENCES

- [1] Brooks, R. Towards a theory of computer program comprehension. *International Journal of Man-Machine Studies*, Vol. 18, 1983, pp. 543-554.

TEXT EDITING PERFORMANCE AS A FUNCTION OF TRAINING WITH COMMAND TERMS OF DIFFERING LENGTHS AND FREQUENCIES

VIRGINIA A.L. GUNTHER, DANIEL J. BURNS, DAVID G. PAYNE

State University of New York at Binghamton

There has been considerable controversy recently concerning the characteristics of computer command terms that affect the ease with which naive users acquire and learn to use these terms. Researchers have examined command words in terms of their "naturalness," semantic relation to the command function, specificity and/or obliqueness (as measured by users,

designers and/or experimenters), to name a few. (For illustrative research examples and a review of this literature, see Grudin and Barnard, 1984, Landauer, Galotti, and Hartwell, 1983, and Landauer and Galotti, 1984.) On the positive side, this research has helped bring into focus the need to consider how selection of command terms affects users' performance. On the negative side, however, there seems to be little agreement as to which, if any, of these factors really affects performance levels. This research was conducted to add to the empirical data base and to examine whether the findings from basic research on human memory can be used to predict performance in a real-world text editing task.

A primary goal of the present research was to assess the extent to which a variable that affects human memory performance (normative frequency of occurrence of the target items) will similarly affect performance when naive users learn to use a text editor. A second goal was to demonstrate that there are important boundary conditions for extrapolating results obtained with one set of command terms to other, seemingly similar, sets of commands. For example, numerous text editors allow the user to enter either the entire command term or some abbreviation thereof. There have been several attempts to determine which abbreviation rules produce the "best," or easiest to use, abbreviated command terms (e.g., Streeter, Ackroff, & Taylor, 1983). Unfortunately, however, many extant systems still use only one abbreviation rule, namely that the first letter of the command may be used in place of the entire term. Thus, a question that can be asked is whether the specific whole-word terms that produce the best user performance will also produce the best performance when the user is allowed to use the single letter abbreviated commands.

METHOD

Subjects, Design and Procedure. Subjects were introductory psychology students who participated in four sessions spread over two consecutive days. In each session subjects were given hardcopy versions of eight paragraphs, each containing eight errors noted by standard copyediting marks. Subjects were instructed to use the text editor to correct the errors in each paragraph, store that paragraph and then retrieve the next paragraph. The same paragraphs were used in each of the four between-subjects conditions. What varied across conditions was the type of command terms employed. Half the subjects used whole words while the others used the first letters of the corresponding whole word commands. Subjects who received the single letters were not explicitly informed of the correspondence of these letters to the root words from which they were derived. For half the subjects the words corresponded to high frequency words and for the remaining subjects they corresponded to low frequency words. The design was thus a 2 (Length of Command: words vs. letters) x 2 (Frequency: High vs. Low) x 4 (Session: 1-4) mixed-factor design.

RESULTS

A variety of performance measures related to speed and accuracy were obtained from each session. We also obtained data from subjects regarding their experience with computers, word processors and typing. Because of space limitations we will only discuss the variable of the total time to complete each session. This variable has the nice features that it (a) has considerable real world relevance; (b) is related to ease of learning during the early sessions; and (c) is related to real world task completion rate (following practice) during the later sessions. The results obtained with this variable are consistent with the other dependent variables and may be summarized quite simply.

First (not surprisingly), all four conditions showed dramatic and significant improvements across the four sessions. Secondly, there was no main effect of the length of the command term, nor was there a main effect of the frequency variable. More importantly, however, there was a significant crossover interaction of length and frequency. For the whole-word conditions, high frequency words produced better performance than did low frequency words, exactly as would be predicted based on the human memory literature. However, the opposite pattern was obtained with the single-letter command conditions, where the initial letters of the low frequency items produced better performance than the initial letters of the high frequency items.

CONCLUSIONS

There are four main conclusions that may be drawn from this experiment. First, there are indeed factors that affect the ease with which naive users learn different type of commands. Secondly, the findings obtained in basic research in cognitive psychology do predict performance in real world tasks and thus may be useful to system designers. Thirdly, extreme caution must be taken in extrapolating results obtained with one set of commands to other, related terms. In the present experiment the whole word condition that produced the best performance produced the worst performance when the items were reduced to single-letter abbreviations. Finally, further research is needed to assess the extent to which learning one type of command term (e.g., high frequency words) will affect performance when the user is allowed to switch to another "easier" type of command (e.g., single letters).

REFERENCES

- Grudin, J., & Barnard, P. (1984). The cognitive demands of learning and using command names for text editing. *Human Factors*, 26, 407-422.
- Landauer, T. K., Galotti, K. M., & Hartwell, S. (1983). Natural command names and initial learning: A study of text editing terms. *Communications of the ACM*, 26, 495-503.
- Landauer, T. K., & Galotti, K. M. (1984). What makes

a difference when? Comments on Grudin and Barnard. *Human Factors*, 26, 423-429.

Streeter, L. A., Ackroff, J. M., & Taylor, G. A. (1983). On abbreviating command names. *Bell Systems Technical Journal*, 62, 1807-1826.

A LONGITUDINAL STUDY OF AUTHORIZING USING NOTECARDS

MELISSA L. MONTY, THOMAS P. MORAN
Xerox Palo Alto Research Center

Authoring is a general term that includes the tasks of collecting and organizing notes and ideas, documenting sources, and building information structures to produce a report, article, or book. The Xerox NoteCards system was designed to assist in the authoring process and to develop models of authoring. We employed a history graduate student to use the NoteCards system to write a research paper and studied him closely through observations, interviews, videotapes of his working sessions, and archived versions of his working files. Over a period of seven months he collected and organized his notes and ideas and wrote an outline and first draft of his paper using NoteCards.

Organization of notes and other elements of the task. We examined the elements and phases of his task and the way he organized his information. In addition, we analyzed how this organization changed over time in relation to the stages of his work. His NoteFile contained notes, a Bibliography FileBox, a collection of Unfiled Cards, a "Topic Browser," and, later, the outline and drafts of his paper. Each element (NoteCard or FileBox or Browser) in the NoteFile appears as an icon showing the title of the note which, when selected with the mouse, opens a window on the screen to display the contents of the note. NoteCards can be multiply classified in the NoteFile. From the early stages of his notetaking his notes were organized hierarchically by topics. The "Topic Browser" is a graph of his topic hierarchy which he used as an outline of the topics he was planning to include in his paper. Since each icon in the Browser gives direct access to each topic FileBox, he used this "Topic Browser" for filing and retrieving the notes stored in the topic hierarchy. The Bibliography FileBox contains a collection of the bibliographic Sources he used. Each Source is linked to all of the notes collected from that Source.

The information in his NoteFile is organized at least three ways. It is grouped by source because each note is linked to the source it came from. Second, the notes are organized into a hierarchy of topic FileBoxes. Third, some of the notes were organized by the structure of the paper (by means

of "footnote" links from various points in the text of the paper).

Comparing organizational structures. The figure above shows one analysis technique we used to compare the different organizational structures. Each number refers to one note. (They are actually sequence numbers giving the relative time of creation.) On the left of the figure, the notes are grouped according to the bibliographic sources from which they were taken. The center of the figure shows those same notes organized by topics using the NoteCards FileBox mechanism. On the right, the same notes are shown in the order in which they were referenced in the paper. The only notes plotted in this figure are those referenced in the first two sections of the paper. In all, 75 notes were referenced in the paper out of more than 200 notes that were written and filed. (The 75 that were footnoted came from 34 of the 88 sources used. They were filed in 21 of the 39 topic FileBoxes in the topic hierarchy.)

Before he began writing his paper, he reorganized parts of his topic hierarchy so it would better match the way he planned to organize the paper. The three "Time" headings in the figure refer to three adjacent work sessions which were selected to illustrate a period when he reorganized his topic hierarchy. The figure shows notes from two sibling FileBoxes (Time 1) moved together into one FileBox (Time 2) and then subdivided into three FileBoxes (Time 3). In this case, the