PROGRAM ABSTRACTS FOR DEMONSTRATIONS & TEACHING

Three programs for evaluation of human-computer interaction

ROBERT M. SCHUMACHER, DENNIS DECOSTE,
DARYL DUGAN, and PAUL M. GRUNZKE
*University of Illinois at Urbana-Champaign
Champaign, Illinois*

Software has been developed for a human-computer interaction (HCI) course offered by the Psychology Department of the University of Illinois at Urbana-Champaign. In the course, students design experiments and evaluate specific areas of interest in HCI. Three programs used to support these studies are described here. With slight modification, these programs can also be used in HCI research.

Knowledge Representation Program

Description. Knowledge Representation is a program designed to evaluate the hypothesis that people with different levels of expertise in operating systems have different mental representations of those systems (see, e.g., Adelson, 1984; Anderson, 1982; Kay & Black, 1984). This program builds the required data matrices for input into commercially available multidimensional scaling programs needed for analysis of the data. The Knowledge Representation program uses 18 commands that a hypothetical text editor might possess to evaluate differences in knowledge representation of users. The program randomly orders and displays the 153 ($(18 \times 17)/2$) pairwise comparisons required. Subjects respond by entering similarity judgments.

Input. First, subjects input a unique three-character identifier, such as their own initials. Next, they answer a short questionnaire that experimenters use to properly assign the subjects to one of four groups: novice users, casual users, nontechnical users, or technical users of computing machinery (see Card, Moran, & Newell, 1983). Subjects respond to each of the randomly ordered 153 pairwise comparisons by rating each pair on a 7-point similarity scale. Finally, subjects rate each of the 18 commands on five unidimensional scales (e.g., How destructive is this command?). Command order is randomized for each question. Ratings on these unidimensional questions can be used in multiple regression equations as

predictors of the multidimensional scaling output dimensions. For example, if ratings on the destructiveness scale account for a high amount of variance in the multidimensional scaling solution, then one can argue that destructiveness is one of the dimensions used by subjects in their pairwise ratings.

Output. For each subject, the program builds a data file containing answers to the questionnaire on the first line, followed by an 18×18 square matrix of the ratings produced by the subjects. The order of the responses within the matrix is uniform across all subjects. A second 5×18 matrix is created that saves subjects' responses to the unidimensional questions for use in the multiple regression. Again, the order of the responses is uniform across subjects.

Limitations. No reliability analysis is afforded in this program, but such an analysis can easily be added. Intrinsicly, the number of commands available for comparison is limited (to 18), because the number of pairwise comparisons increases dramatically as the number of commands increases. The program can be modified to allow three command comparisons.

Analogies Program

Description. The Analogies program addresses the issue of whether the use of analogies will facilitate the learning of a new system. Well-known examples of analogies used in learning include the use of hydraulic principles to assist in the learning of electrical circuit principles (Gentner & Gentner, 1983) or the use of a filing card analogy to assist in the learning of computer text editors (Rumelhart & Norman, 1981). This program allows evaluation of the subjects' abilities to traverse a complex menu system after receiving various training regimens.

A study using this program might have three conditions: no training; training with a schematic of the menu system that subjects memorize; and training with a schematic and a metaphor that liken the menu system to a multi-story department store with different departments, sub-departments, and elevators for access (see Billingsley, 1982). Since this program supports network-like menu structures, great flexibility is provided.

Input. The Analogies program preselects various routes through the menu for the subjects to traverse. These routes are determined by a data file, which can be easily edited. Subjects must enter simple commands on the keyboard to direct themselves through the menu system and to arrive at requested nodes within the menu structure.

Output. The Analogies program produces a file of the commands, the latencies to issue the commands, and the

---

The authors contributed equally to this paper; the ordering of authors was determined randomly. We are indebted to Arthur F. Kramer for helpful comments. Address correspondence to Robert M. Schumacher, Department of Psychology, 603 E. Daniel St., Champaign, IL 61820.

path the subjects traverse to arrive at the various locations in the menu system.

Limitations. The main limitation of the Analogies program is that the subjects' task is simply to move through a menu structure. Thus, complex tasks that use menus in a secondary role are not currently supported. This menu program can be modified for more complex tasks by using the program as a supporting routine.

Abbreviations Program

Description. HCI has a literature devoted to determining optimal abbreviations for use in computing machinery (Ehrenreich, 1985; Grudin & Barnard, 1985; Sebrechts, Furstenberg, & Shelton, 1986). The Abbreviations program is a very flexible program that is controlled by two ASCII files prepared by the experimenter. The experimenter must define the rules and the commands he/she wishes to use. There is no restriction on the number of sets of abbreviations or the number of commands (to be abbreviated) in each set. It is possible to conduct an experiment in which both the number of commands used in each set and the commands themselves are different. The Abbreviations program tests subjects' abilities to encode commands as abbreviations; however, with changes to the instructions and the rules, it can also be used to test decoding. The recall functions of the program are used to test differences in subjects' abilities to recall commands abbreviated under the various rules and may provide evidence as to which abbreviation is most effective. Each phase of the program is timed, and keystamped output is provided for the encoding/decoding phase. A sample set of instructions and four sets of rules and commands are provided in the Abbreviations program.

Input. Excluding user responses to system prompts, the input consists of two prepared ASCII files. The first file consists of a set of instructions, and the second file contains the abbreviation rules and the commands associated with each rule.

Subjects are first presented with the instructions and are allowed to recall them as needed. Next, subjects are presented an abbreviation rule followed by a set of commands to abbreviate. At the completion of each set, subjects input as many of the abbreviations as possible. When all sets of abbreviations have been completed, subjects are prompted to recall and enter as many as possible of the abbreviations used during the experiment.

Output. Two files are produced for each subject. The first file contains the abbreviations produced by the subject (paired with the expected abbreviation) and the latency of the subject to produce the abbreviation in question. The second file contains the times, accurate to hundredths of a second, required for each correct keystroke of an abbreviation produced by a subject. This file also contains data from the abbreviation recall portion of the program. Each recall attempt is paired with the time required to produce the recall and is coded as either a match or a mismatch. Other descriptive statistics are also provided.

Limitations. The internal documentation contains steps needed to modify the output to facilitate interfacing with statistical programs. Small systems, with less than 256K of memory, will limit the number of abbreviations that can be efficiently processed in each set of abbreviations.

Common Program Characteristics

All of these programs are written in Turbo Pascal by Borland International. These programs have run successfully on IBM and IBM-compatible machines with one 5¼-in. disk drive, DOS 2.0 or higher, and 256K of RAM. All subjects have been able to complete the experimental conditions associated with each study in one or two 1-h sessions. The programs have external data files as needed and are well documented for ease of modification.

Program Uses

An adequate program is a necessary, but not a sufficient, condition to ensure successful study results. These three programs can be used, as written, to support a laboratory associated with an HCI class, but with slight modifications, the programs are sufficient for use in an HCI research effort.

Program Availability

The programs are available on request by writing to Robert Schumacher. Please send a blank 5¼-in. DSDD formatted diskette and \$2 (or \$5 without a diskette). Both the source code and the compiled program will be sent.

REFERENCES

- ADELSON, B. (1984). When novices surpass experts: The difficulty of a task may increase with expertise. *Journal of Experimental Psychology: Learning, Memory, & Cognition*, **10**, 422-433.
- ANDERSON, J. R. (1982). Acquisition of cognitive skill. *Psychological Review*, **89**, 369-406.
- BILLINGSLEY, P. A. (1982). Navigation through hierarchical menu structures: Does it help to have a map? *Proceedings of the Human Factors Society*, (pp. 103-107). Santa Monica: CA: Human Factors Society.
- CARD, S. K., MORAN, T. P., & NEWELL, A. P. (1983). *The psychology of human computer interaction*. Hillsdale, NJ: Erlbaum.
- EHRENREICH, S. (1985). Computer abbreviations: Evidence and synthesis. *Human Factors*, **27**, 143-155.
- GENTNER, D., & GENTNER, D. R. (1983). Flowing waters or teeming crowds: Mental models of electricity. In D. Gentner & A. L. Stevens (Eds.), *Mental models* (pp. 99-129). Hillsdale, NJ: Erlbaum.
- GRUDIN, J., & BARNARD, P. (1985). When does an abbreviation become a word? and related questions. In L. Borman & B. Curtis (Eds.), *Human factors in computing systems: Sigma chi proceedings* (pp. 121-125). Santa Monica, CA: Human Factors Society.
- KAY, D. S., & BLACK, J. B. (1984). Changes in knowledge representations of computer systems with experience. *Proceedings of the Human Factors Society* (pp. 963-967). Santa Monica: CA: Human Factors Society.
- RUMELHART, D. E., & NORMAN, D. A. (1981). Analogical processes in learning. In J. R. Anderson (Ed.), *Cognitive skills and their acquisition*. Hillsdale, NJ: Erlbaum.
- SEBRECHTS, M. M., FURSTENBERG, C. T., & SHELTON, R. M. (1986). Remembering computer command names: Effects of subject generation versus experimenter imposition. *Behavior Research Methods, Instruments, & Computers*, **18**, 129-134.