particular they exhibit both the statement-as-list element and adjacent-as-sibling misconceptions in similar, ambiguous, situations. When confronted with a control structure that initially has a single statement action part students tend to select that statement, whether they mean to add a sibling in a newly created block or they want to add a sibling to the parent control structure itself.

At least two solutions are suggested by these data. One would be to adopt the combined version. Subjects using the combined version performed well on the tasks requiring the addition of a new begin-end block. They also did well on the tasks requiring a new placeholder near an existing optional placeholder. They initially did poorly on the tasks requiring a new statement after the parent of an adjacent statement. However the initial time differences were, after all, gone by the second such task.

The other option would be to adopt the tree walk version, but always include begin-end blocks, even where they are optional. When the student chooses a control structure from the construct menu the block syntax would always appear by default. The tree walk version avoids the optional placeholder errors, and avoids needless early confusion on the adjacency tasks. Including the begin-end blocks by default would define away the remaining ambiguity. Though we did not explicitly code the subjects' verbal responses they often did comment aloud when the environment seemed to them to do the "wrong" thing. Especially early on in the experiment, they seemed to be confused by the automatic appearance/disappearance of the block syntax.

Some might argue that the combined version is preferable because it can instruct students about a particularly troublesome aspect of Pascal's concrete syntax. However we prefer adding the blocks by default. Doing so may offend some Pascal purists, and some may continue to believe that early learning about the details of concrete syntax is particularly important.[15] However a basic rationale for structure editing in the first instance is that the student's initial exposure to computing should be both simple and graceful, thus allowing early attention to be directed towards more fundamental issues of programming method and software engineering. The present study does suggest that it is possible to fine tune structure editing environments to better accomplish that aim.

References

[1] R. Chandhok, D. Garlan, D. Goldenson, P. Miller and M. Tucker, "Programming Environments based on Structure Editing: The GNOME approach," Proceedings of the 1985 National Computer Conference, IFIPS Press, July 1985.

[2] R. Chandhok, D. Garlan, P. Miller, J. Pane and M. Tucker, Karel GENIE, Santa Barbara, California: Kinko's Service Corporation Academic Courseware Exchange, 1987.

[3] J.P. Chin, K.L. Norman and B. Shneiderman, "Subjective User Evaluation of CF Pascal Programming Tools," under review, 1988.

[4] Dennis R. Goldenson, "Teaching Introductory Programming Methods Using Structure Editing: Some Empirical Results," under review 1988.

[5] L.R. Neal, "Cognition-Sensitive Design and User Modeling for Syntax-Directed Editors," Proceedings of the 1987 Conference on Human Factors in Computing Systems and Graphics Interface, Toronto, 1987.

[6] S.P. Reiss, "Graphical Program Development with PECAN Program Development Systems, " Proceedings of the Software Engineering Symposium on Practical Software development Environments, ACM-SIGSOFT/SIGPLAN, April 1984.

[7] C. Scheftic and D. Goldenson, "Teaching Programming Method and Problem Solving: The Role of Programming Environments Based on Structure Editors," Proceedings of the 1986 National Educational Computing Conference, AFIPS, San Diego, 1986.

[8] T. Teitelbaum and T, Reps, "The Cornell Program Synthesizer: A Syntax Directed Programming Environment," Communications of the ACM 24(9), 1981.

[15] To accommodate such concerns, and to allow later explicit attention to Pascal list syntax, it is possible to allow the environmental defaults to be altered as preferences by the teacher or more advanced student.

COMPUTER AIDS FOR VISION AND EMPLOYMENT (CAVE)

DOUGLAS GRIFFITH, HODGE DOSS, DAVID WINFREE

The percentage of blind and visually impaired (VI) adults of working age who are gainfully employed is approximately 33%. The confluence of two factors, the changing nature of the world of work and developments in microcomputer technology, offer the potential of greatly reducing the magnitude of this problem. In the information age a strong majority of jobs can be classified as information processing jobs. Given suitable adaptations, the blind and VI can perform such jobs using microcomputers.

There are three generic types of outputs for the blind: synthesized speech, braille (both hardcopy and paperless), and for individuals with sufficient vision (and most legally blind individuals have some usable vision), large print displays. In addition to these generic types of outputs, the different types of software for controlling access to these outputs are of critical importance.

A wide range of products is on the market to aid the blind or VI computer user. Unfortunately, there is little in the way of performance data to assist the potential user in assembling a particular configuration. Moreover, in spite of this wide range of products, computer usage by the blind and VI is quite low.

The reasons underlying this low utilization rate provide interesting grounds for speculation. There appears to be a general lack of awareness regarding both the availability of the technology and its. Beyond this lack of awareness, however, are the basic issues of human-computer interaction when the user is blind or VI. Computer usage is much more difficult for the blind or VI user. Not only must the specially adaptive hardware and software be mastered, but the lack of ready visual access creates other difficulties which provide sources of frustration. Although there are many cases of the blind and VI using computers effectively, these individuals tend to be exceptional people with technological leanings. Computer access will have to he simplified greatly if a large percentage of blind and VI users are going to be able to use the technology effectively.

The purpose of the Computer Aids for Vision and Employment (CAVE) Program is to conduct basic research into the use of computers by the blind and VI to develop a technology base for creating training programs and product improvements that will increase the access of computer technology to the blind and VI.

The CAVE Program has adopted the research approach advocated by Card, Moran, and Newell (1983). The GOMS (Goals, Operators, Methods, and Selection Rules) model is appropriate at least two levels of analysis. At a superordinate level, it provides the basic paradigm for the research, the researcher's mental model of the project. Here the goals are generic tasks such as word processing, making business plans, etc., and include, in addition to the standard operators, operators that are specific to the adaptive hardware and software.

At this level, different methods can he defined by the different types of adaptive equipment; e.g., synthesized speech, braille, large print software. In turn, the time it takes for a given individual to complete the task successfully would provide the basis for defining a selection rule regarding which method to employ. At a more subtle level, methods can define strategies for accomplishing particular tasks.

Of course, GOMS was proposed originally as the user's mental model of the task at hand. In our research it has become apparent that a GOMS approach provides an especially fruitful means of analyzing the interactions of a blind or VI user with a computer. First of all, it should be appreciated that due to the need to operate the specially adaptive hardware/software in addition to the applications software, the mental model of the blind/VI user is, of necessity, more complex than the mental model of a sighted user. Thus, if the user's model is incorrect applications program commands can be mistakenly executed in place of screen access commands (i.e., commands to the speech synthesizer to read designated portions of the screen). Such misconceptions can have disastrous consequences. Secondly, the lack of, or impoverished, visual access to the information on the monitor can easily cause the user to have an inaccurate mental model with respect to either the content of or the location in the task. For the blind user, a task that would be simple for a sighted user, such as making a spelling correction with the aid of a spell checker, can result in the execution of a chain of erroneous operators which have severely damaging effects on a document due to the user's faulty model of his actions and the task.

A log program is employed to record and time all the user's keystrokes. Eventually, the data from these programs will be used to develop detailed models of individual users' performance on different tasks. There are, however, a number of problems in doing research with blind/VI computer users. A principal problem concerns the large amount of training involved. Although some of our clients have computer skills and are generous enough to donate some of their time to our program, their experience is typically limited to their own systems; whenever a new configuration needs to be considered additional training is required. Typically, however, blind/VI individuals with computer skills are employed and are hard pressed to volunteer time to the program. We are attempting to collect data from these individuals by means of a telephone survey. Most volunteers to the program need to be trained.

In most cases, new laboratory participants require training on keyboard skills. Although practically all the volunteers to our program have been trained to touch type, unless the individual has been working with computers, the ability to type with even a modicum of speed is extremely rare. To address this problem, we have developed a typing tutor for the blind/VI user.

Given adequate keyboard skills, and given synthesized speech as a means of screen access, the next problem is the gaining of proficiency in understanding synthesized speech. It is our considered opinion that the magnitude of the synthesized speech understanding problem has been greatly underestimated in the literature (e.g., Schwaub, Nusbaum, & Pisoni, 1985). The results of Story and Kuyk (1988) of recognition rates ranging from 44% to 66% for first-time users coincide much more with our experience. Even high quality synthesized speech causes difficulties for some individuals. To address this synthesized speech understanding problem, we have developed some simple programs that present either words or sentences and require the user to echo back (via the keyboard) the content of the synthesized output.

Of course, the clients next need to be provided additional training on the specialized hardware and software, as well as with the particular applications programs they are going to be using. Given this considerable overhead in training, it is not surprising that having adequate sample sizes presents a problem. The problem involves more than numbers, however. Typically, each client presents such individual difficulties that one is hesitant about making generalizations across subjects. It would appear that some hybrid of clinical and experimental approaches is called for here.

A problem of particular concern to the blind user community, particularly to the blind user who cannot make use of large print software, is the problem of icon-based interfaces. The difficulties presented by icon-based interfaces should be obvious. Although the problem appears to be tractable, the question is whether there is sufficient interest to address this problem. We are currently just beginning to look into the problems presented to the blind by icon-based interfaces.

REFERENCES

Card, S., Moran, P., & Newell, A. (1983). **The Psychology Of Human-Computer Interaction.** Hillsdale, NJ; Lawrence Erlbaum Associates.

Schwaub, E.C., Nusbaum, H.C., & Pisoni, D.B. (1985). *Some Effects Of Training On The Perception Of Synthesized Speech.* **Human Factors** 27 395-408.

Story, S.M., & Kuyk, T. K. (1988). *First Time Recognition Of Synthesized Speech: A Comparison Of Three Systems.* **The Journal of Visual Impairment and Blindness**, 82, 28-29.

Note: The CAVE Program is supported by the Department of Labor of the State of Michigan as well as by internal funds of the Environmental Research Institute of Michigan.

ARTICULATING THE EXPERIENCE OF TRANSPARENCY: AN EXAMPLE OF FIELD RESEARCH TECHNIQUES

KAREN A. HOLTZBLATT, SANDY JONES, MICHAEL GOOD

Summary

Over the past two years, our field research with users has indicated that elements of an application design can disrupt users' work. Understanding how applications disrupt users' work has helped us to articulate the meaning of interface transparency. Interface transparency and related concepts have previously been explored from theoretical perspectives, but have not been grounded in user data.

The relationship between the user's work and interface transparency is a key element of our understanding. Disruptive systems distract users from their task. Systems can disrupt users by fragmenting the task into elements which do not match the user's view of the task. Insufficient functionality and awkward interface mechanisms for a particular task also disrupt users. We need to understand users' work in much richer detail than we do now in order to build systems that assist them with that work.

Interpretive Field Research

Interpretive field research on human-computer interaction (Whiteside et al., 1986) is a process that allows the researcher to collect and interpret users' responses to software as they are engaged in the use of that software. The researcher is present during the use of the software in the context of the user's actual work setting and work task. The researcher can both observe and question the user about his or her experience with the software and its impact on their work as it unfolds. Data collected in these contextual interviews includes:

- Users' ongoing experience with the product,
- The nature of the users' work,
- The impact of the product on the users' work,
- The meaning of usability for the user,
- Directions for future products,
- Specific problems and strengths of specific products,
- Interaction of specific products with other products, and
- Interaction of the product with the users' environment.

We do not assume that something is experienced as disruptive to work because it seems that way to us. We check out our interpretations with the user to establish a shared understanding of user experience which grounds our interpretations. Similarly, if we see a user doing something effortlessly, seemingly without awareness, we check this out. With the user as co-researcher, we can track aspects of usability to elements of the system implementation.

Development of Usability Concepts

A concept is "A general understanding derived from particular instances or occurrences" (Webster's).

We want to develop a set of concepts that describe usability as it is understood and lived by users. We want to derive concepts which are useful for guiding the design of new systems and interfaces.

We are building a set of descriptive concepts that crystallize user experience. These concepts will provide designers with a new way of "seeing" user's work and usability, with specific implications forsystem design. We are not trying to build a cause-and-effect model of usability. For instance, we are not trying to predict that x units of transparency will automatically produce y units of usability.

Building concepts from interpretive field research is an inclusive process. As we interview each person, new instances modify and expand the concepts in our understanding of usability.

Interviews reveal instances of the phenomenon. The instances are used to build concepts. The concepts form a framework for communicating user experience to designers. Designers use the concepts to design the next system.

Transparency: An Example of a Usability Concept

Transparency is not a new concept. Rutkowski (1982) proposes that transparency is the ideal relationship between user and tool, with the tool seeming to disappear. Winograd and Flores (1986) relate this aspect of computer