Technical Report Documentation Page

1. Report No.	2. Government Accession No.	3. Recipient's Catalog No.
UMTRI-85-27		

4. Title and Subtitle	5. Report Date
THE WIZARD OF OZ: A TOOL FOR RAPID DEVELOPMENT OF USER INTERFACES	June 1985
	6. Performing Organization Code
7. Author(s)	8. Performing Organization Report No.
Paul Green and Lisa Wei-Haas	UMTRI-85-27
9. Performing Organization Name and Address	10. Work Unit No.
University of Michigan Transportation Research Institute Ann Arbor, MI 48109-2150 U.S.A.	11. Contract or Grant No. None
	13. Type of Report and Period Covered
12. Sponsoring Agency Name and Address	Final
University of Michigan Transportation Research Institute Ann Arbor, MI 48109-2150 U.S.A.	14. Sponsoring Agency Code

15. Supplementary Notes

16. Abstract

The Wizard of Oz technique is an efficient way to examine user interaction with computers and facilitate rapid iterative development of dialog wording and logic. The technique requires two machines linked together, one for the user and one for the experimenter. The experimenter (the "Wizard"), pretending to be a computer, responds to user queries either directly or by pressing function keys to which common messages have been assigned. The software automatically records the dialog and its timing. This report provides a detailed description of the initial implementation of the Oz paradigm using microcomputers, and a summary of previous applications on other machines. Implementation guidelines, emphasizing potential pitfalls and enhancements not currently included in the literature, are also presented.

17. Key Words	18. Distribution Statement
User Interfaces, human-computer interaction, rapid prototyping, human factors, ergonomics	

19. Security Classif. (of this report)	20. Security Classif. (of this page)	21. No. of Pages	22. Price
Unclassified	Unclassified	18	

TABLE OF CONTENTS

ACKNOWLEDGMENTS

We would like to thank Stacy Reifeis, Marijean Price, and Kara Heinrichs for their reviews of drafts of this report and Paul Ziots for his help with debugging early versions of the software. We would also like to thank Elizabeth Zoltan-Ford for her meticulous explanation of the origins of the Oz technique. Finally, we would like to thank Marilyn Mantei, who at a seminar, brought the Oz technique to Paul's attention.

ORIGIN OF THE IDEA

The Wizard of Oz paradigm is based upon the L.F.Baum (1900) story of that title. In the story, the Wizard produced astonishing images of himself by manipulating a set of controls while hidden behind a curtain. Witnesses of these apparitions initially believed them to be the Wizard himself. The Wizard of Oz paradigm discussed here provides much the same scenario, in that images of the software prototype are presented to the user by a "wizard" (the experimenter) from behind the scenes. The unknowing user believes that a fully functional application system exists.

The formalization of the paradigm occurred in Chapanis's communications laboratory at Hopkins. While several different explanations of its origin have been offered, the most likely one comes from Gerald Weeks, as described in a letter from Zoltan-Ford (1984). The idea was conceived but not fully implemented in 1975 during development of Michael Kelly's Ph.D. dissertation (Kelly and Chapanis, 1977), when Weeks was a research scientist at Hopkins. At that time Oz was referred to as the "experimenter in the loop technique."

As with any discovery, what is important is not who was first, but whose report led to other events. Within Chapanis's group, the first use of the paradigm was for a comparison of voice and keyboard natural-language inputs, using Randy Ford's CHECKBOOK program (Zoltan, Weeks, and Ford, 1982). The first public presentation of the idea had actually occurred a year earlier in Gould's description of a "listening typewriter" study done at IBM (Gould, Conti, Hovanyecz, 1981). While not the first report of its use, Chapanis's lucid explanation of its merits at a symposium (Chapanis, 1982) was instrumental in drawing attention to the technique. (In Chapanis's case, the methodological details actually appeared in a proceedings appendix, submitted after the meeting had ended.)

The first appearance of the "Wizard of Oz" name in print was in Jeff Kelley's thesis (Kelley, 1983a, 1983b, 1984a). It is thought the name was coined in response to a question at a graduate seminar at Hopkins (Chapanis, 1984; Kelley, 1984b). "What happens if the subject

sees the experimenter" (behind the "curtain" in an adjacent room acting as the computer)? Kelley answered, "Well, that's just like what happened to Dorothy in the Wizard of Oz." And so the name stuck.

PREVIOUS APPLICATIONS

Four applications of the Wizard of Oz technique exist in the open literature. Each application is somewhat different, with each contributing a new idea about how the methodology might be applied.

In the initial application, Gould, Conti, and Hovanyecz (1981, 1982, 1983) examined the use of a large-vocabulary "speech-driven typewriter." Since such a machine was beyond the state of the art, Gould simulated one. Subjects dictated into a microphone and a skilled typist listening from another room immediately entered what was said into a terminal connected to an IBM mainframe. Words in the vocabulary were echoed to a terminal before the subject. Echoing was on a word-by-word basis to enhance the illusion of a "listening typewriter." Basic editing and correction capabilities were also provided. For example, saying "nuts" would erase the last word, "nuts 5" the last five.

Zoltan, Weeks, and Ford examined the effect of input mode on user interaction with a natural-language checking account manager running on an IBM mainframe. In response to either typed or spoken input, output appeared on a CRT. A clever twist to this experiment was inclusion of a practice condition where subjects "trained" the computer to recognize their voice by reading standardized phrases (e.g., "Now is the time for all good men to come to the aid of their country"). Unknown to the subject, the voice recognizer was actually an experimenter in another room, typing in what the subject said into a computer.

Kelley (1983a, 1983b, 1984a) used the Oz method to develop a natural-language interface for a computerized calendar. Early on, the Oz paradigm, running on a mainframe computer, was used to simulate the interface. The experimenter, behind the scenes, recoded queries and entered them into a real data base. From information obtained in these interactions, a language processor was derived. Later, the Oz technique supplemented this interface, with the wizard (a "co-processor") handling inputs that the program was not able to process. With subsequent versions of the language processor, involvement of the invisible wizard diminished, until the program ran without intervention. The notion of testing the strength of the software by phasing out the wizard is one of the contributions of Kelley's research.

Wixon, Whiteside, Good, and Jones (1983) (see also Good, Whiteside, Wixon, and Jones, 1984) used the Oz technique to define an interface for an electronic mail system. This study represents the transition of Oz from a purely research tool to one that is also useful in software development. Participants were tested in both the lab (using a superminicomputer) and in a shopping center store (using a minicomputer). Any commands the subject entered on the computer that were not in the command set were intercepted by an unseen experimenter and translated into acceptable system commands. Using this "interactive system," an expanded set of common commands was generated.

Thus, previous applications have focused on natural language interfaces. For one-of-a-kind tests it often does not make sense to develop special computer hardware and software when flexible liveware (people acting as wizards) with fully functional natural language interfaces are readily available.

A second focus of previous applications has been on special-purpose Oz software designed to run on large computers. While that is fine for research, designers need software to run on small machines to foster iterative development.

THIS IMPLEMENTATION

This implementation of the Wizard of Oz is unique, in that it relies upon microcomputers, is much simpler than its predecessors, and makes extensive use of user-definable function keys for message presentation. The experimenter, acting as the "wizard," sits at a microcomputer that is connected to the subject's machine. To create a message for the subject, the experimenter either types it in directly or presses a function key to which a multiple-character message has been assigned. Only after a return key is encountered is the message sent to the subject's machine. The subject sees the cursor reveal the message as a smooth stream. However, keystrokes entered by the subject appear on the experimenter's screen as they are typed, allowing the experimenter to anticipate replies. Each character presented, along with the time between returns, is automatically recorded by the Oz software running on the experimenter's computer.

To implement Oz studies, the experimenter needs an IBM Personal Computer (PC) with at least 64K of memory (128K is preferred), one disk drive, an RS-232 serial interface, display and a display driver, and either a second PC or a dumb terminal for the subject. (Currently the Ann Arbor Terminals 431E, Ambassador, and Zenith Z-19 are supported. Other terminals may work, depending upon how they handle destructive backspace.) The two machines are linked via a standard null modem cable.

Required software includes Advanced BASIC (version 1.1), PC-DOS (version 1.1 or later), and the Oz software written in interpretive BASIC. (There are actually two versions of the Oz software, one for the experimenter's microcomputer and a terminal emulator for the subject's microcomputer.)

Also required is a keyboard enhancer/macro processor. Software used to date includes PROKEY (RoseSoft, 1983), KEYSWAPPER (Vertex Systems, 1984), and NEWKEY (Bell, 1984). This software allows the experimenter to assign character strings to keys (e.g., F1 = "Enter choice"). Thus, lengthy messages can be generated with a single keystroke. This makes the "system" response times brief and helps preserve the illusion of communicating with a computer. (Direct entry

is primarily for on-the-spot creation of messages to subjects for unanticipated errors.)

The initial application at The University of Michigan involved 38 students (mostly seniors) enrolled in Industrial and Operations Engineering 491--Human Factors in Computer Systems (Green, 1984). Two-thirds of the students were in Industrial Engineering; almost all of the remainder were studying Computer Science or Electrical Engineering. The industrial engineers had completed one human factors course and generally one computer course. The other students had completed many computer courses but had no previous human factors training.

The class was divided into ten interdisciplinary teams. Each team was asked to develop a user interface for a home computer banking program. The software would allow customers to determine the current balance in checking and savings accounts, transfer funds between them, pay bills, stop payment on checks, and perform security-related functions. The interface was to be easily learned and used by people of all levels of computer experience.

Each group was required to produce a simulated interface, several reports, a user's manual, and a short videotape of the interface in operation. To be included in one report were predictions of learning and performance times of novice and expert users. Performance predictions could be based on Oz simulations or derived from the Model Human Processor (Card, Moran, and Newell, 1983).

As part of the development process, it was strongly suggested that students use the Wizard of Oz software. Computer science students initially leaned towards exercising their programming skills rather than using the unfamiliar Oz software. Examples of coding problems helped change their minds. (For instance, "$4.00," "300," "2,100," and those values correctly and incorrectly spelled out as words, could all be acceptable entries for money.) Nonetheless, three of the ten groups chose not to use the existing Oz package. One wrote a program in PASCAL for the Apple Lisa, one team decided to write application-specific software, and the last group modified the program by placing all messages in a disk file instead of using keyboard-enhancing software. Both the PASCAL and the application-specific programs required an

extraordinary amount of effort to write. Message retrieval in the file-based system (retrieved as needed) was particularly slow.

The general plan was for groups to develop an initial message set and decision rules, test one or two users, revise the messages and rules, test additional users, and so forth. Typically, four to six users were tested as part of the development process. The result of such an iterative procedure is enhanced usability (Gould and Lewis, 1983).

To further the reader's understanding of how this software might be used, a sample experimenter's start-up screen (Figure 1), a listing of pre-defined keys (Table 1), an experimenter's mid-session screen (Figure 2), and an output file listing for an abbreviated hypothetical session (Figure 3) are provided.

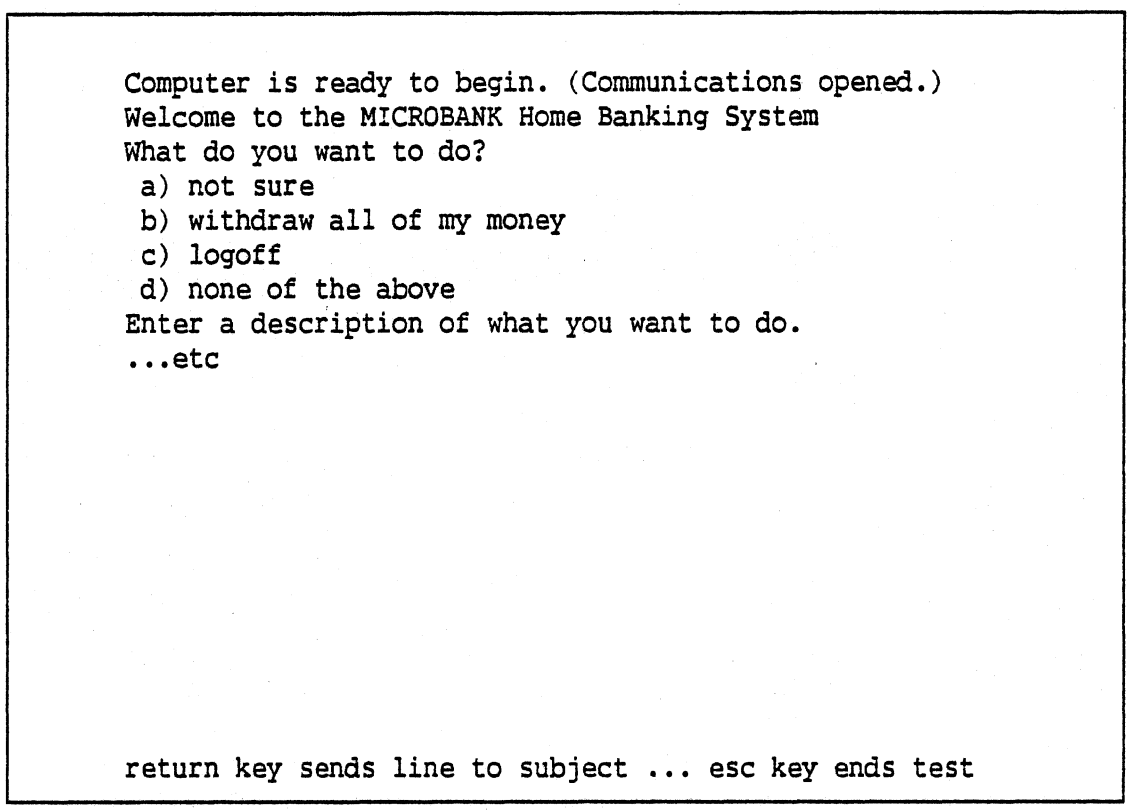
```
WIZARD OF OZ COMMUNICATIONS PROGRAM FOR THE IBM PC/XT
written by Paul Green (with help from Paul Ziots)
University of Michigan (Ann Arbor)
version of 9/4/84

This program will record communications between your IBM PC or XT  and
a  test  subject's terminal.  All characters typed along with the time
between carriage returns are recorded in a disk file.

--- Files are ----------------------------------------------------
C:\comm
        .   <DIR>         ..  <DIR> PC-TALK .EXE     PC-TALK .KEY
PCTKREM .MRG       PC-TALK .DEF     PC-TALK .DIR     OZ-E    .BAS
OZ-S    .BAS       OZ-OUT1          TALK64  .BAT     TALK128 .BAT
PC-TALK .BAS       OZ-INSTR         OZ-HUMOR
  5091328 Bytes free

------------------------------------------------------------------



Enter the name of the file where the data goes. demo

Enter the experimenter(s)' initials. pg

Enter the subject's name or initials. Guess Who

Is the subject using an IBM PC/XT? (y or n)? y

To end the test, press the escape key on the IBM keyboard.

Press any key to start recording.
```

Figure 1. Sample experimenter's start-up screen.

TABLE 1

HYPOTHETICAL LISTING OF PRE-DEFINED KEYS

Key combination	Definition
F1	Welcome to the MICROBANK Home Banking System
F2	What do you want to do?
Shift/F2	a) not sure b) withdraw all of my money c) logoff d) none of the above
F3	And now for something completely different-



Notes:

1. The message "Computer ... opened.)" is from the Oz software. A similar message appears on the subject's screen. ("Computer is ready.")

2. The last (25th) line "return key ... test" appears only on the experimenters' screen and serves as a reminder. It does not scroll.

Figure 2. Experimenter's screen from a hypothetical session (shown mid-session).

```
Wizard of Oz dialog of 09-05-1984 11:39:13

..program written by Paul Green (with the help of Paul Ziots)
..U of Michigan, UMTRI-Human Factors, Ann Arbor, MI 48109-2150

Experimenter(s)=pg  Subject=Guess Who

listing of file=demo

-------------------------------------------------------------

 12 E: Welcome to the MICROBANK Home Banking System.
  2 E: What do you want to do?
  1 E:  a) not sure
  1 E:  b) withdraw all of my money
  1 E:  c) logoff
  1 E:  d) none of the above
 15 S: d
 17 E: Enter a description of what you want to do.
 12 S: This is a stickup.
 25 E: "Stickup" is not a legal operation.
 21 E: ... Please come to the bank for assistance.
 12 E: This sounds like a Monty Python routine.
  2 E: And now for something completely different-
 16 E: Enter a description of what you want to do.
 22 S: I want to tell a terminal joke.
 11 E: "terminal joke" - what's that?
 33 S: enough
 10 E: logging off ... Have a nice day!
  1 ^[

ended at 00-05-1984  11:46:30
```

Notes:

1. Each line contains in columns 1-3 the time in seconds from the return of the previous line (or the start signal for the first line) until the return for the line in question was encountered, in column 5 who originated the line (the experimenter (E) or the subject (S)), and in columns 8 and beyond what was entered.

2. The times for the first few lines are short (1-2 seconds) because they were function key entries (F1, F2, shift/F2).

Figure 3. Output file listing for a hypothetical session.

What kinds of facilities are needed?

In the application at Michigan, students conducted user tests either in the laboratory or at the Engineering Library. At the library, experimenters signed out connecting cables and linked any two PC's together from among the rows of machines present. Because many computers were in use at a time, it was often not apparent to subjects that they were conversing with the experimenters at another computer in the same room.

Most of the laboratory studies were performed at the University of Michigan Transportation Research Institute at night or on weekends. Cables were strewn down the hallway between various offices and a nearby conference room. Despite their impromptu nature, both arrangements worked successfully.

What kind of instructional materials are needed?

Experimenters were provided with a four-page handout supplemented by instructions scribbled on the blackboard describing the software. There were no manuals or demonstration videotapes. Additional instructional materials could have overcome several minor difficulties. Nonetheless, the ability of students to learn and utilize the software rapidly with minimal instruction testifies to its simplicity. Most questions concerned the keyboard enhancing software, not the Oz package.

Are changes to the communications process needed?

Students reported there was an intermittent communications bug (causing the first character of a session to be misinterpreted), and the 300 baud rate seemed sluggish at times. They were not major problems. Upgrading the baud rate to 1200 may require implementing an XON/XOFF protocol and possibly compiling the code to avoid communications buffer overflows.

One common problem was concurrent entry of data by the experimenter and subject. For example, if the experimenter typed "ENTER SELECTION" and the subject typed "pay from checking," "EpNaTy EfR rSoEmL chECeTckIOinNg" would appear on both screens. These conflicts clearly

detracted from the credibility of the simulation. Enabling the keys on the subject's machine to beep or click often helped, but only under very quiet conditions. Future versions of Oz should buffer lines to prevent these collisions.

The return function frequently presented problems for both experimenters and test subjects. When defining keys for the interface, experimenters had the option of embedding a return at the end of the message. If excluded from the definition, the experimenter had to remember to type it after pressing the defined key. Theoretically, omission of the return in the key definition was to increase efficiency, as messages and menus could be combined as needed. However, this inconsistency promoted delays and experimenter errors. Explicitly showing where returns are needed on the experimenter's listing of defined keys may alleviate this problem. Switching the cursor from a blinking underscore to a blinking or steady block may also help, as could giving the experimenter a second monitor showing exactly what the subject is seeing.

In spite of these difficulties, the ease of use of the Oz technique and its "magical" effectiveness created considerable enthusiasm in both subjects and experimenters. Subjects commented that at times they forgot they were not in communication with a real banking system.

Which keyboard enhancer should be used?

Associated with each enhancer were unique problems. Of the three packages examined thus far, NEWKEY is the cheapest (available on local computer bulletin boards), least powerful, and least well documented; PROKEY is at the other end of the scale on all dimensions, and KEYSWAPPER lies somewhere in between. KEYSWAPPER (not copy-protected) was made available for trial use by students. Some groups obtained their own copies of PROKEY (copy-protected). NEWKEY was not available when the course was taught. Insufficient memory for key definitions and the controlling software in the 64K machine was a problem encountered by groups using the PROKEY software, even for this moderately sized interface. This problem was circumvented by either deleting comments from the Oz program or using a 128K machine.

The nesting of key definitions is a feature in all three enhancers, and potentially an advantage since it can conserve large amounts of memory. However, this function must be used with some caution, since accidental use of a previously defined key in a message string might produce nonsense. For example, if the letter "f" was defined as "Found your account," and the F1 key as "Press ENTER to confirm," then pressing the F1 key would create the message, "Press ENTER to conFound your accountirm.."

It is much easier to assign multiple character messages and menus to a single key using the PROKEY than the other software. NEWKEY has the disadvantage to making it easy to accidentally redefine one key in the process of defining another. In one demonstration, the "a" key was inadvertently redefined as the adjacent "s" key. When a subsequent message was typed, a very elusive pseudo-typographical error appeared. With KEYSWAPPER, many students inadvertently loaded the Dvorak keyboard. Subsequent typing caused a panic as many believed they had "broken" the computer.

Ideally, future versions of Oz should include an integral keyboard enhancer. Definitions should be constructed by using a variety of editors and reading them into memory prior to a test. This could increase the response time of the software, but probably not very much. Incorporating the feature implies dismissing the capability to define keys on the fly. Experience has shown that this is rarely done, and when it occurs, the spontaneous definitions are often wrong.

How many experimenters are needed to conduct a test?

Short response times were essential to maintaining the illusion of a working system. Those were fostered by practice walkthroughs prior to testing subjects. Although Oz requires only one experimenter to run a test, several groups discovered it was better to have two "wizards." With one person to aid in the referencing of the code sheet, the "system" response time improved. Moreover, the additional "wizard" was able to note difficulties as they unfolded (e.g., instructions scrolling off the screen after repeated errors, mid-response delays, etc.). The ability to annotate dialogs via the keyboard as they occur may be a valuable feature to add to the Oz software.

It was also beneficial to place a third experimenter unobtrusively near the subject to note the subject's comments and problems, and, in this study, to act as the bank "HELP" phone. In a more formal laboratory test, this observer would watch the subject using a one-way mirror or video system and a real "HELP" phone would be provided.

What additional features are needed?

A great deal more could be done with the time data. Splitting the between-return key times into thinking time (from the ending of one line until typing recommences) and typing time would be useful. Also convenient would be a routine to sort the messages according to their associated thought times. Clearly, the messages with the longest times are the ones most likely to need revision.

In demonstrations given since the course, people have suggested that tab keys and protected fields (for form-filling), mice, joysticks, speech input, speech output, pull-down and pop-up menus, and windows be supported to facilitate the evaluation of state-of-the-art hardware and software. (The current Oz simulation is basically a "glass teletype.")

What procedures should be followed in conducting tests?

People on several teams commented that formalization of the materials given to subjects (copies of their own phone bills and statements, written tasks to perform, etc.) made the experiment much more realistic. Others noted that it was wise to keep the explanations to subjects of the banking problem at a minimum, since this detracted from the legitimacy of the test.

CONCLUSIONS

While the previous discussion identifies many weaknesses of the current implementation, this quick and inexpensive tool has proven to be extremely useful. Teams of students enrolled in a three-credit course who had never before designed a user interface developed reasonable protocols for one, in one month, at the end of a semester, when there were many competing demands from other courses. In addition to its development, manuals and reports for the prototype were written, and a videotape created, all in that month!

Clearly, the Wizard of Oz tool greatly expedited interface development and emphasized the iterative nature of the process. Virtually no programming code must be written for the interface prototype; hence, dialogs can be developed and modified without having to worry about potentially tiresome code changes or a break in a daily testing schedule.

Moreover, this preliminary set of studies emphasized the simplicity of learning to use the Oz methodology. Students with minimal experience with the tool and little instruction were able to collect useful data almost immediately.

All too often, work is not fun. Subjects enjoyed participating in Oz studies and so too did the experimenters.

Finally, the materials needed to conduct a similar user interface study are minimal, as access to two machines (one an IBM Personal Computer) is no longer a major limitation. With the previously mentioned enhancements added to the Oz software package, the Wizard of Oz paradigm provides a compelling methodology for developing user interfaces.

REFERENCES

Baum, L.F., The Wizard of Oz, (originally written in 1900), New York: Holt, Rinehart, and Winston, 1982.

Bell, F.A., NEWKEY: A Keyboard Enhancer for the IBM Personal Computer, version 1.1, 1984, (available from Frank A. Bell, 20950 Smallwood, Birmingham, Michigan, 48010).

Card, S.K., Moran, T.P., Newell, A., The Psychology of Human-Computer Interaction, Hillsdale, New Jersey: Lawrence Erlbaum Associates, Inc., 1983.

Chapanis, A., "Man/Computer Research at Johns Hopkins," Information Technology and Psychology: Prospects for the Future, Proceedings of the 3rd Houston Symposium, pp. 238-249 in R.A.Kasschau, R. Lachman, K.R. Laughery, eds., New York: Praeger Publishers, 1982.

Chapanis, A., personal communication, 1984.

Good, M.D., Whiteside, J.A., Wixon, D.R., and Jones, S., "Building a User-Derived Interface," Communications of the ACM, October 1984, 27(10), 1032-1043.

Gould, J.D., Conti, J., and Hovanyecz, T., "Composing Letters on a Simulated Listening Typewriter," Proceedings of the Human Factors Society-25th Annual Meeting, Santa Monica, CA: The Human Factors Society, 1981.

Gould, J.D., Conti, J., and Hovanyecz, T., "Composing Letters on a Simulated Listening Typewriter," Proceedings of the Conference on Human Factors and Computer Systems, Gaithersburg, MD, Association for Computing Machinery, 1982.

Gould, J.D., Conti, J., and Hovanyecz, T., "Composing Letters with a Simulated Listening Typewriter," Communications of the ACM, April 1983, 26(4), pp. 295-308.

Gould, J.D. and Lewis, C., "Designing for Usability---Key Principles and What Designers Think," Proceedings of the CHI'83 Conference on Human Factors in Computing Systems, New York: Association for Computing Machinery, Inc., 1983a, pp. 50-53.

Green, P., "Teaching a Course on Human Factors and Computer Systems," IEEE Computer Graphics and Applications, December 1984, 4(12), 43-47.

Kelley, J.F., "An Empirical Methodology for Writing User-Friendly Natural Language Computer Applications," Proceedings of the CHI'83 Conference on Human Factors in Computing Systems, New York: Association for Computing Machinery, Inc., 1983a, pp. 193-196.

Kelley, J.F., "Natural Language and Computers: Six Empirical Steps for Writing and Easy-to-Use Application," unpublished Ph.D. dissertation, Baltimore, MD: The Johns Hopkins University, 1983b.

Kelley, J.F., "An Iterative Design Methodology for User-Friendly Natural Language Office Information Applications," ACM Transactions on Office Information Systems, March 1984a, 2(1), pp. 26-41.

Kelley, J.F., personal communication, 1984b.

Pew, R., "Lecture 22: How to Study User-Computer Systems," in Pew, R. and Green, P., (eds.), Human Factors Engineering Short Course Notes (25th edition), Ann Arbor, Michigan: The University of Michigan Chrysler Center for Continuing Engineering Education, 1984.

RoseSoft, Inc., Prokey 3.0 User's Guide, Seattle, Washington: RoseSoft, Inc., 1983.

Vertex Systems, KEYSWAPPER Version 1.4 User Manual, Los Angeles, California: Vertex Systems, 1984.

Wixon, D., Whiteside, J., Good, M., Jones, S., "Building a User-Defined Interface," Proceedings of the CHI'83 Conference on Human Factors in Computing Systems, New York: Association for Computer Machinery, Inc., 1983, pp. 24-27.

Zoltan-Ford, E., personal communication, 1984.