Int. J. Man-Machine Studies (1984) **21**, 49–63

Speech-controlled text-editing: effects of input modality and of command structure

D. L. MORRISON†, T. R. G. GREEN, A. C. SHAW‡ AND S. J. PAYNE

MRC/ESRC Social and Applied Psychology Unit, Department of Psychology, The University, Sheffield S10 2TN, U.K.

Performance measures and satisfaction ratings were obtained from skilled typists and from non-typists using two different designs of editor, one requiring more commands but simpler ("short transactions"), the other needing fewer commands but more complex ("long transactions"). Each subject used the same editor in two versions, one with all input from the keyboard, the other with spoken commands but typed parameter strings. The results indicate that short transactions were preferred, although they were not always most error-free. Speech input was consistently rated lower than keyboard by typists; non-typists initially preferred speech but swung to preferring keyboard. Although the dislike of speech may have been due to the limited hardware, subjects' comments suggested that switching modality during a command was inherently disruptive.

Young child: "Mummy, why are you talking to that machine? You know it can't understand you."

Mother: "Sshh, keep quiet or it'll hear you!"

(From a conversation during the experiment reported below.)

Introduction

Speech recognizers are becoming effective. What are they good for? This article addresses one possible area of use, text-editing. Meryowitz & van Dam (1982), in their authoritative review, say that "voice input devices... may prove to be the text devices of the future" (p. 325). It is not yet possible to supply all input by speech, using cheap equipment, and we therefore developed a system in which commands could be spoken and literal text typed. We compared this system with a structurally identical one in which all input was typed.

At the same time we also compared editing systems in which the transaction cycle (number of actions required before the system responded) was short or long, by comparing two editor designs. With a short transaction design, some commands take few parameters but achieve rather little; in a long transaction design, the corresponding commands take more parameters but achieve rather more. We shall discuss the notion in more detail below. Choosing between commands with few parameters or many is a perennial problem for the designers of interactive computer systems, and at present little empirical evidence has been offered. Both schemes have advantages. If commands take few parameters, each command is conceptually simple; also the transaction cycle

† *Present address:* Department of Applied Psychology, UWIST, Cardiff, Wales.
‡ *Present address:* SYSTIME Ltd, Leeds, U.K.

0020-7373/84/070049+15$03.00/0

is short and knowledge of results comes quickly. If commands take many parameters, fewer distinct command names need be learned, and a given task can be performed with fewer commands.

The interactive computer system chosen for investigation was a simple line-oriented text-editor. Line-oriented editors are not as popular as screen editors, but were chosen as the experimental tool because the transaction cycle length could be easily manipulated. Many other interactive systems closely resemble line-oriented editors; for instance, simple data-base systems contain commands such as "search for the string TARGET", whose structure is a command name ("search for the string...") and a string, just like a typical command in a line-oriented text-editor.

The questions we address are therefore the following.

Are short transaction lengths better or worse than long?
Is it better to speak commands (and type parameters) or to type all input?
Is there an interaction between input modality and optimum transaction cycle length?

Before discussing the experimental design we shall look briefly at the background of command language studies and of speech recognition.

COMMAND LANGUAGE STUDIES

By the length of the transaction cycle, we mean the number of user actions performed before the system generates observable knowledge of results. We do not propose to quantify the measure precisely. The number of actions depends closely on the grain of analysis and, in a detailed study of editing, Card, English & Burr (1978) were unable to choose a grain size that was clearly correct. Of course, the transaction length often varies between commands in any one system, as well as varying in aggregate between systems. In the Unix "*ed*" editor, a well-known line-oriented text-editor, a typical short transaction is

```
p
```

which prints the text on the current line. A medium length transaction is

```
s/cat/dog/p
```

meaning "change 'cat' to 'dog' and print the results". Longer still is

```
/cat/s/Name/Apellation/p
```

meaning "find the next line containing the string 'cat' and on that line change 'Name' to 'Apellation', and print the result".

It can readily be observed that only the experienced user is happy with long transactions; less-experienced users would prefer to find the required line, verify it, and only then make the change. The reticence of inexperience would be still more marked with such a command as

```
/cat/,/dog/m/fauna/
```

"find the next line containing 'cat' and the next line containing 'dog', and move the block bounded by those lines to follow the next line containing 'fauna' ". Although that command can be presented quite simply as "Move everything between A and B up to C", novices seem to prefer the sequence "Mark A as the beginning. Mark B as

the end. Mark C as the destination. Move as marked". The latter form gives knowledge of results at every step, but demands more commands and a greater number of different commands.

The majority of behavioural studies of command language systems have dealt with text-editing. Some have dealt with entire systems (Roberts & Moran, 1983) and many have investigated the details of user input actions. Fewer have investigated the construction of the command language, although a study by Ledgard, Whiteside, Singer & Seymour (1980) has shown very effectively that changes to the command syntax can produce massive performance effects, and studies by Barnard, Hammond, Morton, Long & Clark (1981) have shown that the choice of command names and command syntax can be important. Elaborate analytical efforts have been applied to the mapping between tasks and actions by Moran (1981), and an equally elaborate analysis of the concept of mental workload in command systems has been made by Treu (1975). However, we know of no evidence directly bearing on the question of transaction cycle lengths; see, for instance, the review by Embley & Nagy (1981).

Though evidence may be lacking, opinion is not. Tesler (1981) presents an extensive polemic against modes, tarring all possible interfaces with the same brush, whether the "mode" was accessing part of one file while editing another, or using such editing "modes" as Insert and Replace. In the SmallTalk interface some effort has been made to unite insertion, deletion, and replacement into one mode. The implication is that a prefixed command followed by one or more parameters creates a transient mode, and is as such deplorable: modes lock the user out from other activities, in particular from browsing to collect the information required to complete the command ("user preemption"). The preferred style minimizes transaction length by using postfixed commands with a minimum number of parameter strings. While Tesler's particular example, in which a user has to escape the editor completely in order to discover the name of a file to be accessed, carries conviction, the argument fails when pushed to the extreme. In the experiment reported here, users never had to browse for information not present on the screen, and there is no clear *a priori* argument for minimizing transaction lengths at the cost of increasing the number and variety of commands.

SPEECH RECOGNITION

Significant research efforts in speech recognition have led to greatly improved performance, but have tended to say little about how the advanced technology might sensibly be used. This is particularly true of isolated-utterance recognizers, which are at present much less expensive than continuous-speech recognizers but which perform a far more limited job. "Hands-busy" tasks, where a person has to stop one activity, put something down, and make a keyboard entry, are one obvious candidate: Welch (1980) reports that in industrial applications no clear advantage can be demonstrated with present technology. Another candidate is the control of devices for disabled and handicapped people.

One previous investigation of speech-controlled text-editing has been reported, a futuristic approach by Gould, Conti & Hovanyecz (1983), who wondered whether keyboard input could be dispensed with altogether. Gould *et al.* ingeniously simulated a "listening typewriter": users faced a display screen and spoke into a microphone to edit the document displayed. In the next room a skilled typist translated their commands

into keystrokes. Gould *et al.* reported that users composed letters more slowly than with a traditional dictating machine, but quite liked the system.

Ingenious though it may be, the "listening typewriter" simulated very advanced technology indeed. More importantly, the task of Gould *et al.* required compositon, rather than editing existing text.

In the present article, we shall postulate that existing technology can be refined to make cheap, accurate speech recognizers available with a vocabulary of about 100 isolated utterances. It would then be possible to edit text by using spoken commands. The text itself would still have to be typed, however, since the vocabulary would normally be far greater than 100. Thus, in order to insert the word *example* at the point where the cursor is in a document, it would be necessary to speak the command "insert" and then to type the word "example". We shall call this speech-plus-keyboard. The question then is how it compares with the conventional keyboard-only method of text-editing.

INTERACTIONS

One view is that speech-plus-keyboard would be quite ineffectual; since typing is fast and accurate, and the typist's hands are already at the keyboard, there is nothing to be gained.

An alternative view is that speech-plus-keyboard would be preferred, at least by poor typists, because it requires less typing. For unskilled typists every keystroke is a separate penance, so that any possibility to reduce typing should be advantageous.

It is also possible that speech-plus-keyboard might have certain advantages for skilled typists as well as unskilled: notably, it achieves a classic separation of function by modality. Since commands are spoken and literal text is typed, a high degree of psychological organization is imposed. In particular, speech-plus-keyboard solves the problem of ambiguity, where the same string may be either a command or a literal, depending on the context. Thimbleby (1982) discusses the difficulties caused by character-level ambiguity in typical word-processing and text-editing systems. Obviously, if the commands are spoken and the text typed, no possibility of ambiguity can arise. It is a well-established finding that, all else being equal, well-organized response sets are easier to learn and allow faster, more accurate performance.

An experiment by Payne, Sime & Green (1983) offers some support to this argument. They compared the learning and use of extremely small command languages in which the degree of overlap between the symbols used for commands and those used for parameters was varied. When the command symbols were perceptually separated from the parameter symbols, by the simple means of putting one set in upper case and the other in lower case, performance improved substantially.

Supposing, therefore, that the preceding argument is correct, then we can predict an interaction between input modality and editor design. The editor with a short transaction length will gain less from having its commands put into a different modality, since it is already easier to use, than the editor with a long transaction length.

To assess these interactions, we compared text-editing by skilled and unskilled typists using short-transaction and long-transaction editors, and using keyboard-only input and speech-plus-keyboard. Because both systems are alike in requiring keyboard entry of running text, there would be little to be learned from using a task like that of Gould

et al., which emphasized text composition; instead, the experimental task required correction of previously-prepared documents.

Method

DESIGN

Two styles of text-editor were compared, the MC and FC styles described below, which could each be used in a keyboard-only mode or in a speech mode (more accurately, speech-plus-keyboard). Both expert typists and novice typists took part, and each subject used one style of editor consistently throughout the experiment, but alternated between speech and keyboard modes. The design therefore has three factors, each of two levels: style of editor × typing experience × input mode, with repeated measures over input mode.

Structured interviews at the start and end of the experiment were used to reveal subjective impressions and preferences. During the experiment all interactions with the computer were logged.

APPARATUS

Subjects sat at the keyboard of a Terak 8510 console, on which was displayed the document to be edited. When using the speech-driven editors, a microphone was attached to the movable arm of an adjustable table-lamp, replacing the bulb housing, so that it could be swung into any position the subject desired. Before using the microphone the subject pressed a foot-operated pneumatic switch; with the switch in its resting state, the microphone was disconnected and the subject could talk to herself freely. The microphone was part of the Heuristics H2000 SpeechLink speech recognizer, mounted in an Apple II computer. Identifications from the Apple were transmitted to the Terak for action. When a sound could not be identified, the Apple notified the Terak and the word PARDON? was displayed.

The accuracy of speech recognition was improved by using an adaptive algorithm to compensate for changes in diction during the task. During preliminary investigations, this algorithm greatly improved recognition rates for the SpeechLink (Green, Payne, Morrison & Shaw, 1982), although it should be noted that the vocabulary and task were different and that direct comparisons with the present task should therefore be avoided.

The Terak was also attached to a third computer which performed the actual text editing, using the Unix "*ed*" editor. *ed* was invisible to the subjects, and all their commands were translated into *ed* commands automatically. The text for editing was held in this third computer but could be transmitted to the Terak so that the subject could see its current state.

A reference list of commands was available to subjects. It was pasted to the table and covered by a hinged flap so that the experimenter could count the number of times subjects referred to it.

THE EDITORS

Four major tasks were required during the experiment: deletion of lines; insertion of lines; alteration of letter-strings within a line; and "cut-and-paste", or relocation of

a group of lines. Two line-oriented editors were specially constructed, in which these four tasks were achieved in different ways. The MC (Many Commands) editor provided many simple commands, such as RAISE to move the cursor one line upwards. A few commands required a parameter string, such as LOCATE ABC which moved the cursor downwards to the next line containing the string ABC. The FC (Few Commands) editor provided "larger" commands, most of which required at least one parameter string.

The difference was most noticeable for the cut-and-paste task. Using the MC editor the sequence of commands was to move the cursor to the start of a chunk to be relocated, and to mark it with the command BEGINNING; then to move the cursor to the end of the chunk and mark it with the command FINISH; then to move the cursor to the destination, and use the command TRANSFER to perform the text relocation. The entire task could be performed without once using a command that required a parameter string. Moreover, until the final command TRANSFER was given the beginning and end markers could be repositioned freely. In the FC editor only one command was given, but it required three parameters identifying the start, finish and destination: TRANSFER ABC GHJ XYZ transferred a block, starting with a line containing ABC and ending with a line containing GHJ, to the position immediately after a line containing XYZ. In each case, the line used was the first line found, searching downwards, which contained the target string.

In the speech versions, the commands were spoken into a microphone and the parameter strings were typed at the keyboard. In the keyboard-only versions, the commands were issued at the keyboard using specially-programmed function keys. The RETURN key was used to end keyboard input for both editors, and a function key was used as a separator between strings where required in the FC editor. None of the ordinary text characters was used for any special purpose, so that there were no problems of command ambiguity.

The full command sets for the two editors are given in Table 1. The names of the commands were chosen to describe the effect of each command while being acoustically distinguishable by the speech recognizer. For instance, the name RUBBER was chosen because "delete" and "rubout" were mis-recognized as "locate" too easily. The commands were identically named in the speech and the keyboard-only versions of each editor.

PROCEDURE

The experiment was designed around four 1-h sessions: an introductory training session, two "structured editing" sessions, and a "free editing" session.

The introductory training was intended to be through a tutorial document, appropriate to whichever style of editor the subject was to use; but the social situation was uncomfortable to subjects and so the experimenter talked them through the tutorial documents. Half the subjects were trained on the speech version first. Like many other speech recognizers, the SpeechLink must be "trained" to recognize each speaker, and so, besides learning to use an editor, subjects also practised using the microphone, reading the command words, and then testing to see whether the words were correctly recognized. It was necessary to give most subjects some advice on how to achieve reliable recognition, the usual problem being that they spoke rather timidly into the microphone at first. The first structured interview was given at the end of this session.

TABLE 1
Command sets for the two editors

Command	Parameters (if any)	Effect
(a) *MC editor*		
Top		Move pointer to first line of text
Bottom		Move pointer to last line of text
Raise		Move pointer up 1 line
Lower		Move pointer down 1 line
Locate	target string	Move pointer to next line that contains the target
Rubber		Delete current line
New	insertion string	Inserts new line above current line
Modify	target string, new string	Changes target to new string
Beginning		Mark current line as start of block to be moved
Finish		Mark current line as end of block to be moved
Transfer		Move marked block to line above current line
(b) *FC editor*		
Top		Move pointer to the first line of text
Bottom		Move pointer to the last line of text
Rubber	target string	Deletes next line that contains the target
New	target string, insertion string	Inserts new line of text below next line
Modify	target string, new string	Changes target to new string
Transfer	target string A, target string B, target string C	Move the block between A and B inclusive to C

During the "structured editing" sessions subjects corrected a document on the screen, working from a sheet of required corrections. The order in which to make corrections was indicated and the nature of the operation was implied, although not stated outright (Fig. 1). Two such documents, T1 and T2, were corrected in each session. Half of the subjects used the speech mode in the first session and the keyboard mode in the second, half used the reverse.

In the fourth session, "free editing", subjects corrected a document on the screen working only from a paper copy of the document in its desired final form, so that they had to determine the editing operations for themselves. Two documents, T3 and T4, were corrected. Half of the subjects worked in the order T3 by speech, T4 by keyboard, T3 by keyboard, T3 by speech, and half worked on T3 and T4 in the same order but exchanging speech and keyboard. At the end of this session a second structured interview was given.

SUBJECTS

Of the 20 subjects, 10 were qualified typists who were (or had been) employed within the previous months as secretaries for a minimum of two years. The remaining 10 were non-typists of a similar age and social background, with minimal keyboard experience. No subjects had ever used an interactive computer terminal.

REFERENCES

Adam, A. and Laurent, J.P. (1979). A debugger to teach programming to students. Paper presented to 8th World Computer Congoress.

Al-Jarrah, M.M. and Torsun, I.S. (1979). An empirical analysis of COBOL programs. Software-Practice and Experience, 9, 341-359.

Allen, J.R. (1974). The development of computer courses for humanists. Computers and the Humanities, 8, 291-295.

Austin, H. (1976). Teaching teachers LOGO: The Lesley Experiments A.I. Memo No. 336, Artificial Intelligence Laboratory, Massachusetts. Institute of Technology, Cambridge, Massachusetts.

Austing, R.H., Barnes, B.H. and Engel, B.L. (1977). A survey of the literature in computer science education since Curriculum '68. Communications of the ACM, 20, 13-21.

Baxford, T.H., Burkhardt, D., Dodd, W.P., Laflin, S., Parkyn, D. and Ramsay, P. (1979). ATOL: A simple language with powerful data structuring facilities. SIGPLAN Notices, 14, 5-15.

Baeker, R. (1975). Two systems which produce animated representations of the execution of computer programs. SIGCSE Bull., 7, 1-20.

Barron, D.W. (1977). An Introduction to the study of programming languages. Cambridge Computer Science Texts 7. Cambridge. Cambridge University Press.

(1) Alter to Congress.

(2) Insert Alcock, D. (1979). Illustrating Basic. Cambridge. into the slot indicated.

(3) Move the whole reference to the slot indicated.

(4) Get rid of the line indicatod.

(5) Move the whole reference to the slot indicated.

(6) Baeker should read Baker.

(7) Insert Gazzelli, P. A. The fruits of semantic trees into the slot indicated.

(8) Get rid of the last line.

FIG. 1. An editing task. Subjects received a marked-up copy of the document and the corrections required, as shown here.

DEPENDENT VARIABLES

Owing to limitations of the hardware it was impossible to log times for individual keystrokes. Only the times at which a command was completed could be recorded. The dependent variables recorded were threfore time per command, the command itself, whether the command was syntactically correct, and whether the command was logically possible. (A command to raise the cursor past the top of the document would count as logically impossible.)

During the structured interviews, subjects were asked for both overall and specific comparisons of the speech and keyboard editors. First, they rated the two versions overall on a 10-point scale in terms of general preference; then they rated the ease of performing each of four specific functions (NEW, RUBBER, MODIFY and TRANSFER), again on 10-point scales; and finally they were asked to elucidate the reasons for their ratings and to state their preferences for the speech or keyboard versions in terms of pleasantness, efficiency of use, and ease of learning.

Results

TIMES

It was not possible to record times for individual keystrokes, nor to extract "think times" between commands, but the total time to complete a set of editing tasks and the average time per command were both analysed. Logarithms of times were used throughout, to reduce the strong positive skew.

No particular editor or input medium was faster to use overall. Analysis of variance, comparing the two editor styles and the two input media, revealed no significant main effects or interactions, either early in practice (first two sessions) or late in practice (second two sessions). Naturally, one must beware of treating this as a demonstration that no genuine population differences exist.

Average times per command were analysed separately for the FC and MC editors, since the amount of keyboard input per command was much greater for the FC editor

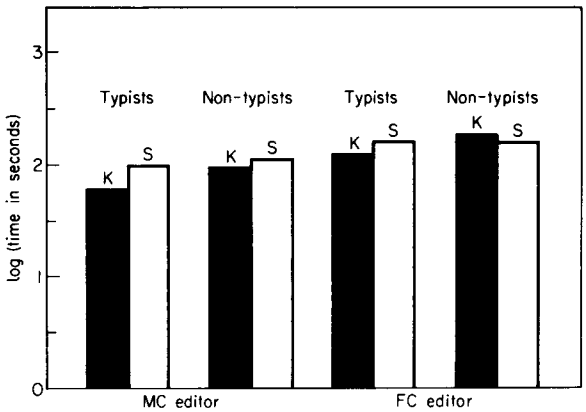

FIG. 2. Times per individual command. Hardward limitations made it impossible to obtain separate "thinking" and "typing" or "speaking" times; these times therefore encompass one entire transaction. K = keyboard, S = speech-plus-keyboard.

and the average time was therefore greater. The mean log times are presented in Fig. 2. For the MC editor, analysis of variance comparing typing skill (typists/non-typists) and input medium showed that typists were "almost significantly" faster ($F = 4{\cdot}90$, $d.f. = 1, 8$, $P = 0{\cdot}058$) and that speech input was significantly slower ($F = 12{\cdot}8$, $d.f. = 1, 8$, $P = 0{\cdot}007$); there was no significant interaction, although visually Fig. 2(a) shows that the best times were certainly achieved by typists using keyboard input.

For the FC editor, the same analysis showed no significant effect of typing skill, but again speech input was significantly slower ($F = 10{\cdot}32$, $d.f. = 1, 8$, $P = 0{\cdot}0124$) and there was no significant interaction.

ERRORS

An error was scored when the subject gave a command that could not be obeyed, whether because it was syntactically ill-formed or because it was logically impossible or because it referred to a non-existent target string. An error was also scored when the subject gave a command that was obeyed, but that clearly did the wrong thing; typically, moving the cursor to the wrong line. The two types of error showed equivalent practice effects.

The raw error data were logarithmically transformed for statistical analysis, to reduce the usual positive skew. Analysis of variance (groups × input medium) revealed no main effects for groups ($F = 1{\cdot}58$, $d.f. = 3, 16$, $P = 0{\cdot}23$) or for input medium ($F = 1{\cdot}62$, $d.f. = 1, 3$, $P = 0{\cdot}22$); in other words, the overall error frequencies were not significantly different for typists and non-typists, or for speech and keyboard. The interaction was significant ($F = 3{\cdot}27$, $d.f. = 3, 16$, $P = 0{\cdot}049$).

Further analyses were made on the two editor styles taken separately, to explore the interaction.

(i) The MC editor [Fig. 3(a)]: analysis of variance (group × practice × input medium) of error frequencies revealed a highly significant practice effect, as one would expect, and a slight interaction effect for group × input medium ($F = 6{\cdot}52$, $d.f. = 1, 8$, $P = 0{\cdot}06$); separate analyses for typists and non-typists showed that the interaction was caused

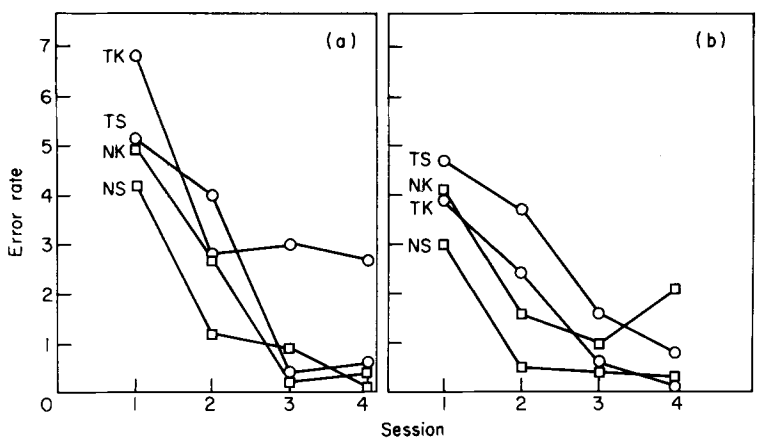

FIG. 3. Mean number of errors made by each subject during the four editing sessions. T = Typists, N = non-typists; K = keyboard, S = speech-plus-keyboard. (a) MC editor, (b) FC editor.

by typists making significantly fewer errors when using speech than when using keyboard ($F = 8{\cdot}82$, $d.f. = 1, 12$, $P = 0{\cdot}041$), an unexpected finding.

(ii) The FC editor [Fig. 3(b)]: a similar analysis (group × practice × input medium) again revealed a highly significant practice effect. Separate analyses for typists and non-typists did not suggest that either group performed significantly better with one input medium rather than the other. Separate analyses for the two media, however, showed that non-typists using speech made significantly fewer errors than typists using speech ($F = 6{\cdot}16$, $d.f. = 1, 8$, $P = 0{\cdot}038$).

Overall, therefore, the FC editor produced the expected result, that the typists were better with keys whereas non-typists were better with speech, but on the MC editor typists performed particularly badly when using the keyboard.

TABLE 2

Percentage of errors made with each editor,† classified by type. Profiles for non-typists and typists were very similar and were therefore pooled

	(a) Command order		(b) No carriage return		(c) Wrong location		(d) Lack of context		(e) Inappropriate command	
	S	K	S	K	S	K	S	K	S	K
MC	13	21	13	18	5	3	3	3	7	15
FC	21	19	1	1	18	23	4	7	3	3

† S = Speech, K = keyboard.

When the errors were broken down by types (Table 2) it was clear that the different editors created different problems. Failure to end commands with the Return key was far more frequent in the MC editor, and using an inappropriate command was also more frequent—possibly because there were more commands available to be confused about. Sending the cursor to the wrong location was a more frequent source of errors in the FC editor, perhaps because subjects had to use target strings to locate the cursor and could not resort to simple UP and DOWN commands.

SUBJECTIVE MEASURES

The subjects gave overall ratings on a scale from 1 (very poor) to 10 (excellent) of a number of issues: overall impressions of each medium, ease of using four specific functions, pleasantness, efficiency, and ease of learning. The ratings were supplied during structured interviews at the end of the first and the fourth sessions. Mean overall ratings are displayed in Fig. 4.

Analysis of variance of overall ratings revealed a highly significant interaction between editor style and input medium, and separate analyses were made for the two styles of editor. For the MC editor, overall ratings increased significantly with practice, and at the first stage of practice non-typists gave the speech editor significantly higher ratings than the keyboard version, whereas typists rated the keyboard editor higher. The difference between input media for non-typists had disappeared by the final stage

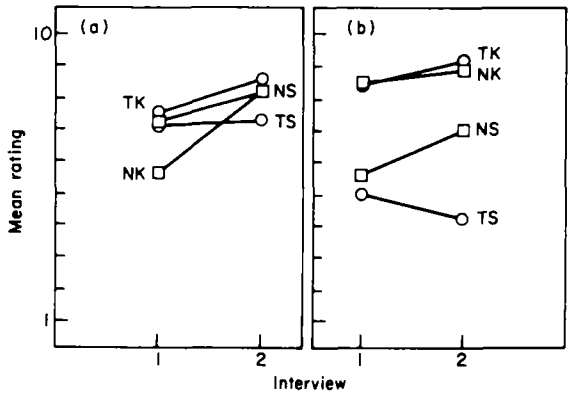

FIG. 4. Overall ratings (means of all rating scales) averaged over subjects. Interview 1 took place after the first editing session, interview 2 after the last. The highest ratings express most satisfaction. (a) MC editor, (b) FC editor. Key as Fig. 3.

of practice, but had increased for typists. This seems curious since typists made more mistakes with the keyboard.

The FC editor produced somewhat more dramatic results, as Fig. 4 shows. In the first stage of practice, keyboard input was given higher ratings than speech input by both groups. By the final stage of practice, typists' ratings of the speech version had dropped significantly, although all other ratings rose.

Ratings of four individual editing functions replicated the overall patterns. There was no indication that individual functions reversed the overall preferences.

Ratings of pleasantness during the first interview differed significantly between groups; non-typists preferred the speech input and typists preferred the keyboard input ($P<0{\cdot}05$, Fisher's exact test). Non-typists claimed that this was because they did not have to search for special editing function keys when they used speech. The typists, on the other hand, complained that having to stop typing and vocalize a command interrupted the flow of what they were doing.

The perceived efficiency ratings behaved similarly. Non-typists using the MC editor found speech easier to use, but non-typists using the FC editor were significantly more likely to prefer keyboard input for efficiency of use ($P<0{\cdot}05$, Fisher's exact test). They remarked that having to remember to switch modality from keyboard to speech for command input was less efficient because it was "just something else to remember".

Learnability ratings again followed the same pattern, with more (four out of five) of the non-typists using the MC editor claiming that speech input was easier to learn. Interestingly, a number of subjects (seven out of 20) felt that the two media were equally difficult to learn. The difficulties they had, it was asserted, were not due to the medium, but to the editor design.

Judgments on these issues had not changed to any statistically significant degree in the second interview at the end of the last session, although ratings for keyboard input tended to benefit most strongly.

Conclusions

The overall results can be briefly summarized. Comparing input systems, speech input was uniformly slower, probably because of our limited hardware. Error frequencies

were comparable, but typists were more accurate when using keyboard versions while non-typists were more accurate when using speech. Speech was initially the preferred input medium for non-typists, but their ratings fell with practice. The typists consistently preferred keyboard input.

Comparing the FC (long transaction) editor and the MC editor, we found that times were little different, that MC's popularity increased with practice, and the FC's popularity also rose *except* for the typists using it with speech: their dislike of it increased with practice. For some reason typists made more mistakes when using MC with the keyboard input, mainly through forgetting to use the RETURN key at the end of a command.

COMMAND LANGUAGE DESIGN

The frequency of errors caused by failing to use the RETURN key suggests that, where possible, command languages should avoid the need for an explicit terminator. Unfortunately, that is not always easy to achieve. Both the problem and the difficulty of its solution have been previously noted by Black & Sebrechts (1981).

The preference ratings and error scores produced some unexpected findings. It was expected that with short transactions (the MC editor) speech would be somewhat preferred; and with longer transactions, the preference for speech would be more marked. In fact, however, little interaction was observed for non-typists, while the typists reversed our expectations by giving very low ratings to FC with speech. Yet at the same time, they made fewer errors using FC with speech than with keyboard-only!

The results lend some degree of support to the widely-held belief that short transactions are best for novices, and seem to suggest that short transactions are particularly appropriate for speech interfaces.

SPEECH RECOGNITION, MICE AND ICONS

In the Introduction we remarked that views on the utility of speech recognition were very varied. The present results support the sceptics—at least within the limitations of our hardware. That typists preferred using keyboard input was to be expected, but our non-typists also preferred the keyboard by the end of the experiment; that is, after only four 1-h sessions.

One cannot guess how their preferences would have lain had our equipment been more sophisticated, nor does the experiment predict preferences in tasks where keyboard input can be entirely eliminated. However, as mentioned above, many other information-processing tasks require test strings as parameters, and our results would certainly seem likely to generalize so far.

At present the keyboard entry of commands has become unpopular. Besides the anticipated use of voice control, there is increasing interest in other alternatives. Smalltalk's mouse-controlled cursor has been followed by the Xerox Star, by the Apple Lisa, and by the MicroSoft mouse. Pointing to icons for commands has been investigated by Hemenway (1982) and even gesture and eye-gaze are being investigated (Bolt, 1982). The results reported here do *not* indicate that eliminating typed commands automatically improves performance or satisfaction, at least when the user's task still requires typed parameter strings.

Are the problems caused by a "switching effect"? That is, would they vanish if more sophisticated hardware were used, or is it inherently difficult to switch output modality within a command because it disrupts prior planning? Although one cannot decide

without further research, such comments as "just one more thing to remember" incline us to believe that there are problems in switching modality, and that the novel interfaces now coming into use will have to overcome such difficulties.

References

BARNARD, P. J., HAMMOND, N. V., MORTON, J., LONG, J. & CLARK, I. A. (1981). Consistency and compatibility in human–computer dialogue. *International Journal of Man–Machine Studies*, **15**, 87–134.

BLACK, J. B. & SEBRECHTS, M. M. (1981). Facilitating human–computer communication. *Applied Psycholinguistics*, **2**, 149–177.

BOLD, R. A. (1982). Eyes at the interface. *Proceedings, "Human Factors in Computer Systems" Conference*, Gaithersburg, Maryland. NBS/ACM.

CARD, S. K., ENGLISH, W. K. & BURR, B. J. (1978). Evaluation of mouse, rate-controlled isometric joystick, step keys and text keys for text selection on a CRT. *Ergonomics*, **21**, 601–613.

EMBLEY, D. W. & NAGY, G. (1981). Behavioral aspects of text editors. *ACM Computing Surveys*, **13**, 33–70.

GOULD, J. D., CONTI, J. & HOVANYECZ, T. (1983). Composing letters with a simulated listening typewriter. *Communications of the ACM*, **26**, 295–308.

GREEN, T. R. G., PAYNE, S. J., MORRISON, D. L. & SHAW, A. C. (1982). Friendly interfacing to simple speech recognisers. *Behaviour Information and Technology*, **2**, 23–38.

HEMENWAY, K. (1982). Psychological issues in the use of icons in command menus. *Proceedings, "Human Factors in Computer Systems" Conference*, Gaithersburg, Maryland. NBS/ACM.

LEDGARD, H., WHITESIDE, J. A., SINGER, A. & SEYMOUR, W. (1980). The natural language of interactive systems. *Communications of the ACM*, **23**, 556–563.

MEYROWITZ, N. & VAN DAM, A. (1982). Interactive editing systems, Parts I and II. *ACM Computing Surveys*, **14**, 321–352; 353–415.

MORAN, T. P. (1981). The command language grammar: a representation for the user interface of interactive computer systems. *International Journal of Man–Machine Studies*, **15**, 3–50.

PAYNE, S. J., SIME, M. E. & GREEN, T. R. G. (1983). Perceptual structure cueing in a simple command language. *Memo No. 550*, MRC/SSRC Social and Applied Psychology Unit, Sheffield University.

ROBERTS, T. P. & MORAN, T. P. (1983). The evaluation of text editors: methodology and empirical results. *Communications of the ACM*, **26**, 265–283.

TESLER, L. (1981). The Smalltalk environment. *BYTE*, **6** (8), 90–147.

THIMBLBY, H. W. (1982). Character level ambiguity: consequences for user interface design. *International Journal of Man–Machine Studies*, **16**, 211–225.

TREU, S. (1975). Interactive command language design based on required mental work. *International Journal of Man–Machine Studies*, **7**, 135–149.

WELCH, J. R. (1980). Automatic speech recognition: putting it to work in industry. *Computer*, **14**, 65–73.

Appendix: Speech recognition devices

Recognizing connected speech is extremely difficult, because the computer must find ways to segment the speech into words and then identify the words. Since words sound different in different contexts, and in normal speech there are no acoustic signals for breaks between the words, the problem requires very sophisticated methods incorporating much linguistic knowledge.

Far easier is the recognition of isolated utterances. A limited vocabulary of messages is constructed by recording one acoustic signal for each message. No attempt need be made to analyse the signals, and although words are normally used the signals could

equally well be phrases, whistles or sneezes. The recorded signals, or "templates", are compared one by one to the unknown signal to be identified, and a measure of similarity is computed. The message associated with the best matching template is deemed to be its identification. Some small degree of linguistic "knowledge" is introduced by designing a measure of acoustic similarity which is insensitive to some extra-linguistic features, such as the fundamental frequency of the utterance (which has no linguistic significance in English), but there are great technical difficulties in coping with words pronounced slowly sometimes and quickly at other times. In consequence, isolated-utterance recognizers cannot mimic human perception. They will unexpectedly confuse two utterances that sound quite distinct to us humans, or distinguish between utterances of the same phrase that sound identical to us.

To ameliorate the technical problems it is usual to "train" the recognizer before each use, by asking the speaker to read aloud all the utterances to be recognized while the sounds are recorded in digitized form to be used as templates. Also, the speaker must be cool and disciplined, so that each utterance sounds much the same each time.

Preliminary investigations with our equipment and our subject population gave little ground for believing that our subjects would be sufficiently consistent. It is extremely hard to read a list of words in the same way that they will later be uttered when used as commands, or to speak each command in the same way whether one is relaxed or agitated. An adaptive recognition algorithm was therefore developed in which the training samples were selectively updated after a "confident" recognition; to do this several samples of each utterance were stored, and if one sample gave a sufficiently close match to an input utterance, a *different* sample would be replaced by that utterance. Thus, if the manner of speaking slowly shifted the templates would shift too. The algorithm is fully described by Green *et al.* (1982).