The work performed at the Human Factors Center located at IBM's development facility in San Jose, California, is representative of human factors work being done by groups of human factors specialists throughout IBM. A few of the projects that the Center was involved with are described as examples to show how human factors concerns are studied in the development of products and systems. The examples were selected to indicate the broad nature of the problems studied and include hardware and software areas. The complete scientific techniques used in the projects are not discussed in this paper so that the focus of discussion will be on the nature, scope, and methodology of the human factors work. The computing and data collection systems used for human factors tests are briefly discussed.

Procedures of the Human Factors Center at San Jose

by R. S. Hirsch

Human factors is a technical discipline that basically studies what constitutes the best interfaces between man and machine. Human factors concerns may enter into the development of a product or system at any or all stages, from initial design to testing and use in the field. Studies often result in specific recommendations to design or change products or the procedures for their operation so that those who are going to use them do not find it to be difficult.

Groups of human factors specialists are located at most of IBM's development laboratories. One of the earliest of these groups was the Human Factors Center established at the IBM development laboratory in San Jose, California. Some of the work done at this Center is presented to show what procedures are followed in performing human factors studies. The Center is organizationally a part of the General Products Division, but approximately half its

resources and effort are expended on interdivisional, corporate-wide projects. The basic mission of the Center is to provide management with objective and comprehensive evaluations of products or systems to ensure that man-machine or man-system interfaces have been optimized and that any human factors risks have been assessed and minimized. Implementation of that mission has involved us in many projects, mostly hardware, some software, a few of which will be described as examples.

Computing and data collection support systems

Before discussing specific projects, it may be of interest to many readers of the *Journal* for us to describe our computing and data-collection systems and show how they have been developed specifically for human factors test purposes.

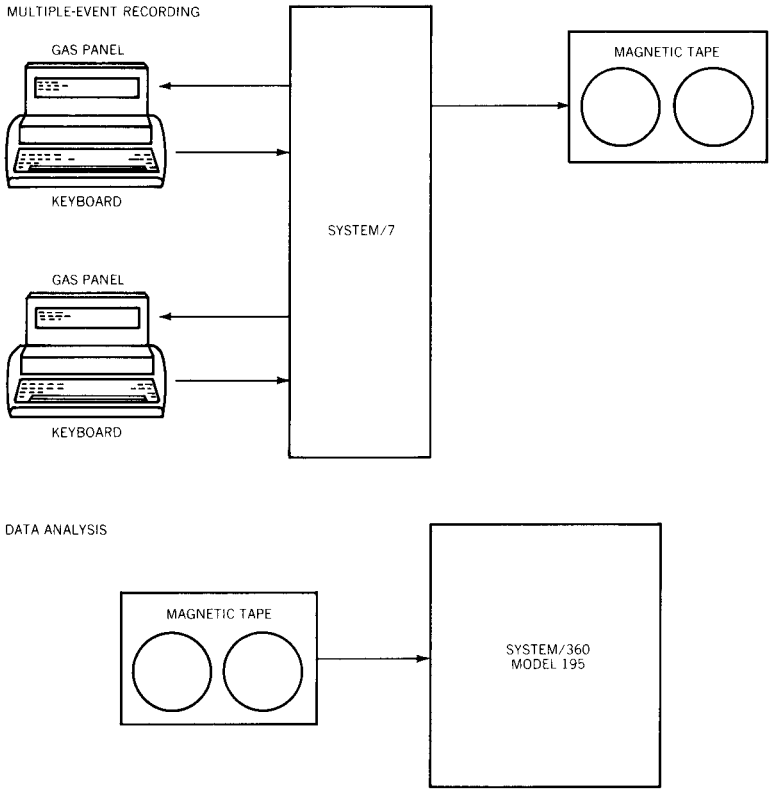
Computing systems

Our computing system requirements are dictated by the type of work we do: testing the human interfaces to other computing systems. The first requirement is, therefore, a way to connect other computing systems to ours. To do that we have used a succession of process control computers. Our first system was a modified IBM 1620; our current system is a standard System/7, but we are also presently testing a Series/1.

Since we attempt to be responsive to the demands of the development cycle, while at the same time maintaining our scientific rigor, we must be capable of testing many subjects simultaneously for a short time. This work requirement implies several system requirements. First, it means we must have not only a large amount of main storage (we have the maximum main storage in all our systems) but also a very flexible multiprogramming system, capable of running many different jobs at once as well as multiple copies of the same job. This includes several jobs sharing the same direct access files. On the 1620 system this requirement was met by *ad hoc* programming for each project. Our System/7 software had to be modified to attain both these requirements. This system requirement also includes being able to compile new programs at the same time subjects are being tested in order for us to prepare for our next test while running the current one. Our present operating system, developed by modifications to the standard MSP/7 software system supplied by IBM, allows us to run six jobs at once plus batch processing (utilities, compilers, etc.).

The second system requirement is related to these last comments; it is the need to be able to develop new programs quickly using a high-level language.

Figure 1 Data collection system of the Human Factors Center



A third system requirement implied by this work requirement is flexibility of attachment to our computing system. As new projects are developed, current projects must often be relocated. Our present system consists of permanent cables to each subject room which are assigned to the job by a JCL (job control language) that we developed. In this way a project can be moved from one location to another by simply unplugging the equipment in the one location, moving it to the new location and plugging it in there, and punching a few cards that are interpreted as the job is loaded.

The fourth and fifth system requirements are based on the fact that we need to gather data about our subjects' performances and store them for subsequent analyses. To store the data, we currently utilize magnetic tape and probably will continue to do so on the Series/1. Magnetic tape has three primary advantages: It can store a lot of data in a small space, it is easily transported, and it can run on either our system or a System/370 computer. In this way we can store data for years and reanalyze them later or loan them to someone else for reanalysis.

Finally, the data we store consist of subject responses together with an indication of the time they took to respond. Responses

may consist of readings from magnetic badges, pulsometers, pupilometers, touch-sensitive pads, etc., but in the great majority of cases they are keystrokes from visual display terminal keyboards. The time of response is read from the computer clock. For most experiments, only relative times are required. However, if real time is necessary, date and time of day can be entered under program control. Since we must be capable of adapting to a wide variety of devices with many different speeds of response, we need an interrupt and timing system that responds quickly. Consequently, we completely rewrote the timer and process interrupt routines. The latter now can turn around in less than 150 microseconds.

Data collection system

Our computer is used to collect data, to simulate the system under investigation, and to control the devices being evaluated. Figure 1 is a schematic of the data collection system. As can be seen, the computer receives signals from a terminal (in the example, a keyboard). Depending on the system logic being simulated, the received signal causes a particular response from the computer. The response may be an auditory tone, the actuation of a transport, the display of a message, or simply the printing (or display) of the character keyed by an operator.

event-recording and timing

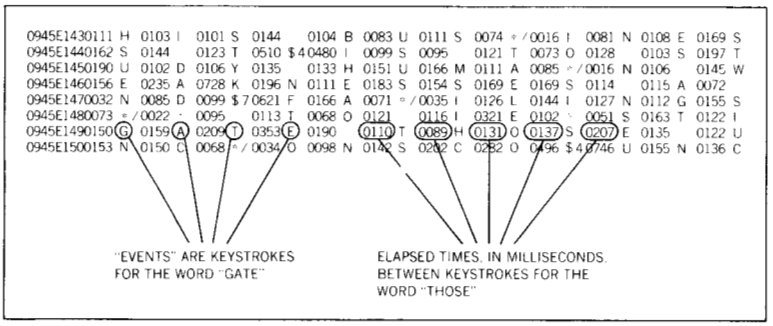
During the above process, the computer is event-recording and time-keeping. An "event" is any identifiable signal received or generated by the computer. An event, on the one hand, may be signals from the depression of a key on a terminal, from the placement of a document in a transport, from the insertion of a card in a reader, or from the passage of a person past a sensor. On the other hand, an event may also be the computer generation of a signal to print (or display) the character of a key depressed, the movement of a document placed in a transport, a tone "accepting" the card inserted in the reader, or the opening of a gate to allow a person to enter. In every case, the event is uniquely identified and recorded together with the elapsed time between events.

Figure 2 is an example of a multiple-event recording during a session in which an operator was typing meaningful sentences. What is shown is that each letter typed was identified and recorded. The numbers preceding each letter are the number of milliseconds that elapsed since the last letter typed (or space, etc.). As all these data are on tape (or disk), we are able to arrange and analyze the information in any way appropriate to yield answers to the questions that led to the study being conducted in the first place.

error identification

Whenever possible in the course of our tests, operators are given material to enter that is appropriate to the system being simu-

Figure 2 Data stream of the data collection "event" recording and timing



lated. That is usually accomplished by visiting IBM customers' locations and extracting a sample from the real-life application material. As a consequence, over the years the Center has accumulated a number of such application data bases. These are used, for example, in studies involving keyboards and displays, among others. The result is that we have identified a reasonably "typical" application that consists of some all-numeric, some all-alphabetic, and some alphanumeric material. The length and percentage of each subset can be varied to reflect a given simulated system.

A further characteristic is that "noise" has been removed from the input material. The kinds of noise referred to here are such things as illegible handwriting or other input features that interfere with the evaluation of the particular device under study. For example, having an operator try to decipher an illegible document may be "realistic" but does not contribute to the comparison of, say, two keyboards of differing technology. On the contrary, what is introduced is a delay in keying due to perceptual problems. The result is an inability to attribute the particular delay to the appropriate cause, namely, to the keyboard technology or to the illegible source document. Accordingly, our application input material has been rendered "clean" of as much extraneous noise as possible and stored in the computer.

As a consequence, when an operator makes a mistake, our software is instantly aware of it because it is doing event-by-event compare operations. If the operator or the system detects and corrects the error, our computer software records the error-making and error-correcting data. Thus, we are able to analyze the frequency of detected errors, the time required to correct them, and the errors that may be made during error correction.

Our software is able, furthermore, to identify errors that escape detection by either the operator or the simulated system. It is, in fact, only by a program such as ours that these errors can be identified and analyzed.

Figure 3 Data collection data stream during which an error was made and went undetected by either the operator or the simulated system

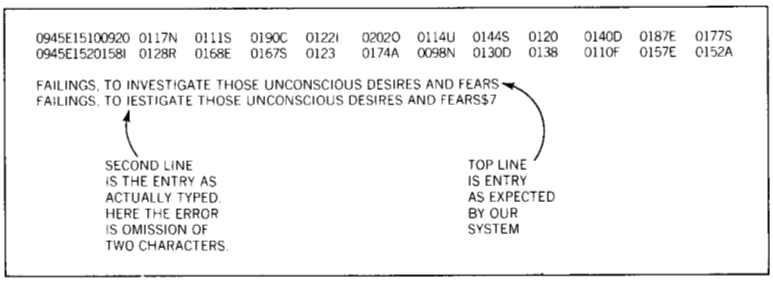


Figure 4 Operator feedback with throughput performance and percent errors

```
S  SES K TOTTME  BTWN EFFTME WSTTME  TKS  EFFKS  CLKS   DSP   ERC   D+F   TER   SUB   OM
54  95 N 603496 569228 515965 053263 2513  2498   015    12    06    00   006    00    02

 4.38 EFFECTIVE KEY STROKES PER SECOND
 4.84 NET KEYSTROKES PER SECOND (NO CORRECTION TIME)
 0.24 PER CENT UNDETECTED ERRORS
 0.24 PER CENT CORRECTED ERRORS
```

Figure 3 shows an example of an undetected error. The first line of text is the form in which the words were expected to be entered into the computer; the bottom line is how the operator actually typed the line. As can be seen in the example, the typist omitted two characters in the word "investigated." Neither the operator nor the simulated system was aware of the error, but our software detected it. In this way, we are able to analyze not only detected, but also undetected, errors. Such undetected errors are more important because they tend to contaminate data all through the system.

operator feedback

Periodically during a study, we stop a session to provide operators with feedback about their performance. Figure 4 shows the information typically given the operators in a keying performance study. Our procedure also includes discussing and reviewing with an operator both the errors detected and those undetected by the operator. In that way operators help us to identify the cause of each error. A summary is made of an operator's performance that identifies errors.

Although not all of the identified items are of equal importance, we have found that it is easier to identify and record the data than to rerun an experiment in order to answer a question that may be asked after data collection is completed. It has also been our experience to have been able to answer a question asked years after a study was completed, by a reanalysis of our store of experimental data.

Table 1 Errors by type

Error type	*Key-board*	*Field*							
		Numeric		*Part number*		*Description*		*All*	
		#	%	#	%	#	%	#	%
Substitution	A	16	0.020	60	0.080	13	0.029	89	0.044
	B	8	0.011	35	0.051	10	0.024	53	0.029
Omission									
Extra keystroke									
Transposition									
Shift error	A	0	0	56	0.075	0	0	56	0.028
	B	0	0	11	0.016	0	0	11	0.006
Field omitted									
Other									
Totals	A	33	0.041	178	0.239	44	0.097	255	0.127
	B	21	0.028	71	0.104	51	0.122	143	0.077
Total keystrokes	A	83,978		78,906		46,752		209,636	
	B	77,753		71,721		44,487		193,961	

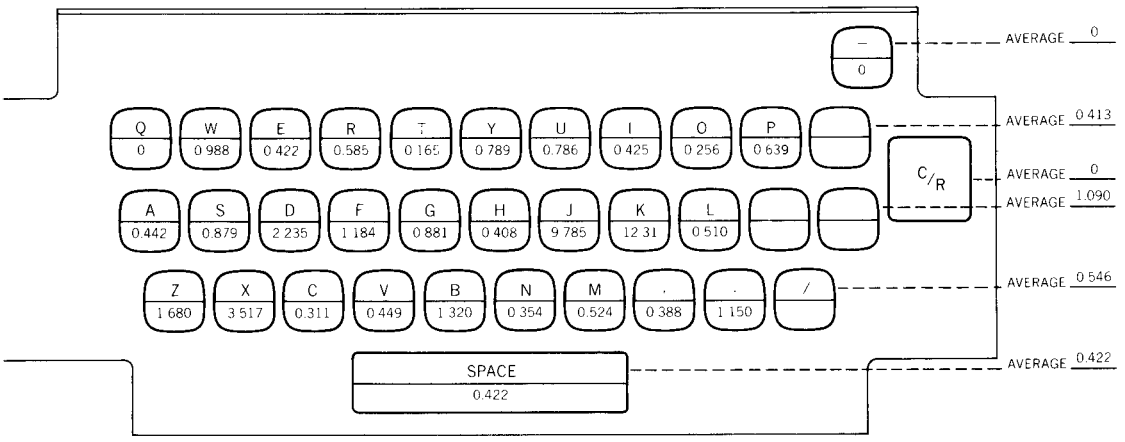
Portion of a table, highlighting the fact that, of the errors made during a particular study, a significant percentage were "shift errors."

Table 2 Shift key errors

Type	*Keyboards*	
	A	*B*
Early alpha shift: In numeric shift, down-shift early so last numeric character is sent alpha.	37	4
Late alpha shift: In numeric shift, down-shift late so next character, which is alpha, is sent numeric	2	0
Late numeric shift: In alpha shift, up-shift late so next character, which is numeric, is sent alpha.	6	1
Early numeric shift: In alpha shift, up-shift early so last alphabetic character is sent numeric.	0	1
Numeric shift omitted: In numeric shift, never down-shifted so numeric characters sent alpha.	7	3
Alpha shift omitted: In numeric shift, never down-shifted so alpha characters sent numeric.	2	1
Confused shifting: Numeric shift for alpha characters; alpha shift for numeric characters.	2	1

Example of an in-depth analysis of all errors related to the "shift" key (referred to in Table 1).

Figure 5 Percent errors during a study by key and by row



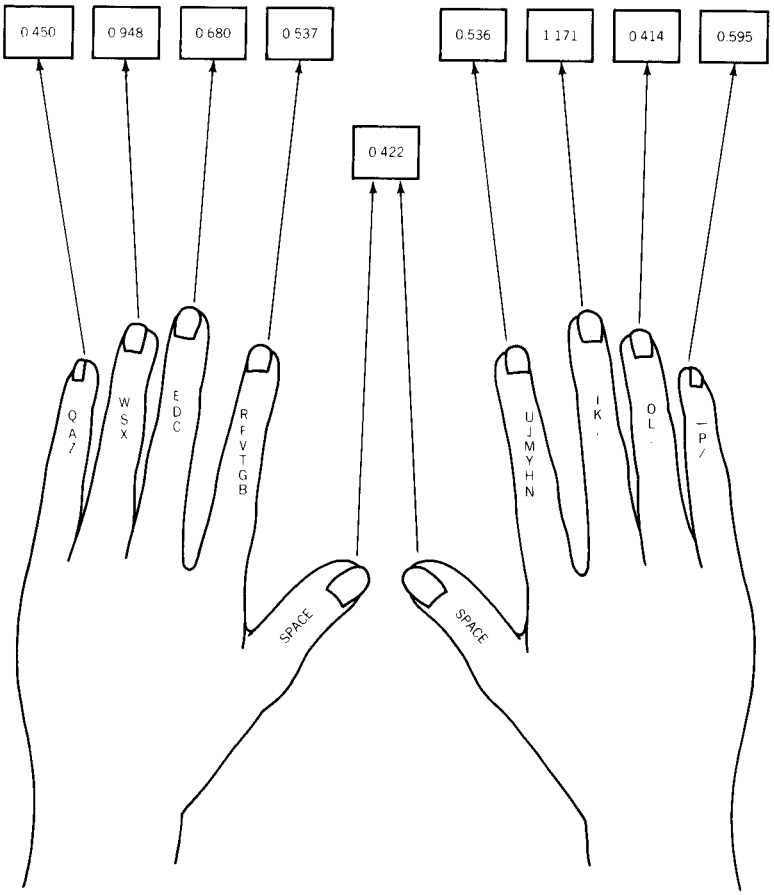
Data analysis

There is hardly much point to the collection of such detailed data unless something worthwhile is done with the information. Table 1 is taken from one of the reports that dealt with a comparison of two keyboards with different technologies. Extraneous information has been removed from the table in order to highlight the fact that, of all the errors made by the operators on each of the keyboards, a somewhat greater percentage of errors than one would expect by chance was attributable to the shift key on one of the keyboards. When we realized that fact, our next analysis was to look at every shift key error in detail.

Table 2 shows that the result of that analysis was the determination that almost all the errors were related to the make-break contact points designed into the shift key. That information was immediately transmitted to the appropriate engineering groups, the design was changed, and the study was resumed with the problem removed.

Another way to look at error data is shown in Figure 5. This method is particularly useful if one has an opportunity to influence the format of a keyboard. The figure shows the percentage of errors associated with each key during a particular study. Figure 6 presents still another look at errors, in this case as a function of the particular fingers involved. Incidentally, an effort is now being made to determine whether or not there is something systematic about error-making by key-entry operators. Figure 7 shows a table from another of our reports that tallies keys ac-

Figure 6 Percent errors by fingers; errors as a percentage of all characters struck by each finger; based on 515 996 effective keystrokes



tually struck against the keys intended to be struck. Obviously, what we are looking for here are unusually large numbers that suggest the locus of a keying confusion.

Some work of the Human Factors Center

During the approximately 20 years that the Human Factors Center has been conducting studies on IBM products, the point at which the work is performed varies from the time when the device is only a concept in an engineer's mind to a time after the product has been delivered and is operating in the field. As a consequence, our involvement comes at various stages during the product development process. Equally varied is the nature of the products, although all the products with which we are concerned have one thing in common: They are intended for use by so-called "end users," that is, people who range from the relatively unskilled (as, for example, the general public using a cash-issuing terminal) to the semi-skilled (a travel agent, for example, skilled

Figure 7 Confusion matrix from a study, showing the keys actually struck versus the key intended to be struck

KEY ACTUALLY STRUCK

INTENDED KEY	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	b	–	·	C/R	TOTAL
A					2										5				10	1	1		1	1		1					22
B	1		1	1		2	1								1							6									13
C				1															2					4							7
D			3		1		1				1	1				12			8												27
E	3			2			2		1						1				6				6	1	5						27
F				2														1							1						4
G		5	4	3		3																									15
H							3			2																					5
I					3										9					2	3										17
J								1			1																				2
K												1																			1
L										1				1	1				1												4
M									1		1			6																1	9
N		14		5				1				1	2						2	2	1						1				29
O	5								3	2					2	3			1			1									17
P																															0
Q																															0
R					4	2													2	3											11
S	1			2	1	1					1			2							1		2	1	1						13
T	1			1			4	1				1						19	1												28
U	1				1				6		1														1						10
V		2	1	1									1	1							2										8
W					2														1	1											4
X																										1					1
Y	1				2			10													1		1								15
Z																															0
b															1								4								5
–																															0
·												1																1	1		2	5
·											1																		18			19
C/R																															0
TOTAL	13	19	9	18	16	8	11	13	11	4	7	5	3	10	20	15	0	23	33	8	9	6	14	7	8	2	2	1	18	3	0	318

in travel arrangements but only incidentally as a typist) to the highly skilled (a full-time word-processing operator, for example). The products, then, are both hardware and software, and in some instances the product may be a system in which the interface to the end user includes both hardware and software.

Also, as a consequence of studying specific products, we sometimes identify problems of a general nature, which we frequently refer to as "basic" or "advanced technology" problems. Sometimes, too, we undertake a study as a service to our marketing force to help them in discussions with potential customers about the performance of a product.

Several examples will be described to illustrate the work of the Center. The examples have been selected to highlight the broad nature of problems studied. In presenting the examples we will not discuss the studies in detail. What we will present is a discussion of the problem that was the impetus for each study. Then, we will describe only the important features of the procedures followed in the evaluation. Moreover, for the sake of clarity we are deliberately not describing scientific techniques; only the most important of the experimental results will be reported and discussed.

The examples selected are intended to describe the scope of problems undertaken, to illustrate something about the experimental facilities and methodologies employed, and, lastly, to indicate something about the various stages during product development in which the Human Factors Center becomes a participant in that development.

One point that we would like to emphasize is that the *raison d'etre* of the Center is to conduct tests under controlled conditions and to derive objective test results, the data from which, together with other data, product management can use in making decisions about product design. Fulfillment of such a mission raises at least three issues and implications. One issue concerns what population to sample; the implication here is that the market for a product must be well-defined by the development group. When the Center understands the target population, we attempt at the beginning to visit customer installations to observe the work and characteristics of the workers. Whenever possible, we attempt to obtain sample application material with which we develop as realistic a set of test materials as possible.

A second issue is the experimental controls needed to derive reliable and detailed data. The implication here is that a laboratory must exist for studies of that nature to be conducted. In the following examples, we will describe the data collection system and procedure. A related implication concerns the training material that may be required for users to learn to operate the product productively. That means that written reference material must also exist; frequently, it is the responsibility of the Center to prepare the required training material, which, whenever possible, we implement in a computer-assisted-instruction mode. We do this in the interests of experimental control; it eliminates the instructor and the subsequent variability in the instructions. That material, incidentally, also becomes an evaluated variable in the course of our testing.

The third issue concerns the question, "How close to reality is the laboratory test to the field environment?" Part of the answer depends on how closely the tested population approximates the real population. Also, how like the "real" application is the test application? How good a representation is the prototyped or simulated hardware and/or software of the end product? These are ongoing concerns of the Center in devising our test procedures and analyzing the resulting data. The implication of this issue is that reasonable tests can only be conducted when good prototypes or simulations can be developed, either by the development groups or, if necessary, by the Center itself.

These matters will be raised again at various times in this paper when we present the examples and describe how we attempt to confront the issues and deal with their implications.

Among the facilities and resources of the Center is a literature file of current and historical reports which presently contains approximately four thousand documents accumulated over the past twenty-five years; accordingly, it is our practice to review the literature for reports relevant to our investigation, but, as our studies are usually product-specific, we find that few of the reports in the literature are related to our studies and that even those few have to be bent to fit our concerns. Accordingly, to focus the reader's attention on the nature, scope, and methodology of the Center's work, we will lighten the bulk of this paper by not including, except in a limited way, the extensive literature reviews, bibliographical references, and research discussions customarily included in papers of the psychological literature.

Hardware studies

The earliest evaluations conducted by the Human Factors Center concerned various keyboard-related issues. One example describes a study designed to determine the effect of keyboard format on the perfomance of nontypists. The second keyboard example deals with the evaluation of a Japanese language data-entry device. And, to illustrate further the variety of keyboard studies, an example is described concerning typewriter feedback for blind operators.

The development of display units as substitutes for hard-copy printed output in text and data entry (and retrieval) generated a number of questions related to the capabilities of cathode-ray tubes, among other display devices. The fourth hardware example concerns the effect of the size of lettering on the display.

Effect of keyboard formats on typing performance

Reviewing the history of the so-called standard typewriter keyboard, Dvorak[1] noted that the original keyboard design was based on the assumption "that typists would 'hunt and peck' with the first finger of each hand," and that in 1873, to avoid mechanical problems, the designer, C. L. Sholes, was therefore forced to locate "in different quadrants of the typebar circle the letters most frequently used in words."[2] The result was, Dvorak asserted, "one of the worst arrangements possible" for the modern touch typist. Declaring that "there is a better typewriter keyboard,"[3] he then proposed what has since been generally known as the Dvorak-Dealey keyboard. Although convincing evidence was submitted to support his claim,[4,5] Dvorak's design did not (for reasons that need not be discussed here) displace the original Sholes arrangement, and keyboard configurations continue to be a subject of investigation.

Keyboards, however, present investigators with relatively peculiar problems, such as the difficulty that arises when an experi-

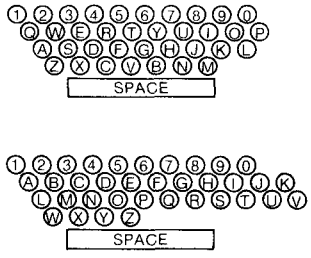
ment is intended to yield results applicable to data-entry keysets in general. The difficulty is associated with application-sensitive questions: If the keyboards are on card punches and bank proof machines, what production and error rates can be expected?[6] What effect is caused by the type of material keyed and the amount exposed?[7] Does redundancy in the data interact with age in the operator?[8] How are keying speed and accuracy affected when the application involves different alphabetic sizes and sequence lengths?[9]

the problem

This study,[10] which has been reported elsewhere in greater detail,[11,12] attempts to answer questions about keyboard arrangements when the application data and the operator characteristics are sufficiently well-defined to make questionable the applicability of findings from earlier studies involving differently described data and operators. Here the design was for a particular computer-based customer service system, and it was assumed that some kind of typewriter-like keyboard would be used for input to a central computer. The employees who were to operate the input device would not ordinarily be skilled typists, and it was further assumed undesirable to add typing skill to the qualifications for their positions. For this reason, the system designers asked whether or not the performance of unskilled typists might be improved by choice of a keyboard format different from the standard typewriter format.

In selecting an alternative keyboard for comparison with the standard typewriter format, some consideration was given to the relatively minor modifications of the standard keyboard proposed by Dvorak[1,3] and others.[5] It seemed likely, however, that the advantages claimed for such modified keyboards would have little significance for typists not trained in touch typing. A more interesting alternative was a keyboard arranged in alphabetical sequence, so that an untrained typist would presumably have little difficulty in finding the keys and remembering their order. It seemed possible that the familiar order of an alphabetical keyboard would contribute to both the accuracy and the speed of typists using the hunt-and-peck method.

Figure 8 Standard keyboard at top and the alphabetical key arrangement at bottom



The two keyboards used in this experiment are shown in Figure 8. Adapting to the constraints of the basic typewriter architecture, we arranged the letters of the modified keyboard in alphabetical order, in two rows of 11 letters each, and placed the last four letters—W, X, Y, Z—on the bottom row.[13]

The problem to be investigated was this: On which of two keyboards, a standard typewriter keyboard or a sequential alphabetical one, will unskilled typists, with a given period of practice, type faster?

the subjects

Typists come with a wide range of skills. Further, many nontypists are to some degree familiar with the standard keyboard, despite a lack of training. In other words, typing skill must be regarded as falling on a continuous scale from some minimum to some maximum. The problem of selecting nontypists for this experiment was solved by first seeking subjects who identified themselves as nontypists, and then selecting for analysis only those whose initial typing rate, as determined from a pretest, was below a certain low level of skill, defined below.

Fifty-five college students who responded to a request for nontypists to participate in an experiment were employed in the study. Fifteen were excluded from the analysis of results because their pretest scores exceeded the level defining nontypist, and thus 40 actually furnished the results for analysis. They worked in pairs, each being assigned by the toss of a coin to either the alphabetical or the standard keyboard.

practice and testing sessions

It was assumed that, in the business system under consideration, if practice could be extensive, training might just as well be given on a standard keyboard. The problem was to decide on a practice period that might be acceptable to an employer who, for whatever reason, might not want to hire only trained typists as operators, or to conduct extended training programs in typing.

In view of these considerations, the study of each subject's performance was completed within approximately seven hours of practice interspersed with 10-minute sessions. The seven hours were divided into two three-and-one-half-hour periods on successive days.

Before beginning the practice session on the assigned keyboard, all subjects were given a pretest on *both* machines, after which the subjects practiced in pairs.

tests and practice materials

In selecting materials for practice and tests, we were guided by a desire to make the tested task primarily a manual one. In other words, an effort was made to minimize perceptual aspects of the task. Our reasoning was that we were primarily attempting to compare hunt-and-peck performance *on the two keyboards*. Hence, the tests should be such that subjects could concentrate on the keyboard by minimizing the perceptual effort required in copying from some text.

Accordingly, the material used for testing was selected to provide a subject primarily with cues that would leave him or her free to concentrate on the hunt-and-peck task, rather than on the material to be copied. The test material, therefore, required a subject to type items such as his name, address, telephone number, mother's name, father's name, etc. It was regarded as reasonably

Table 3 Pretest scores (strokes per sec) on both keyboards

Group (40 operators)	*Pretest score*	
	Standard keyboard	*Alphabetical keyboard*
On standard keyboard†		
$\bar{X}$	0.91	0.52
σ^2	0.206	0.038
On alphabetical keyboard†		
$\bar{X}$	0.88	0.53
σ^2	0.142	0.036
Results for all 40 *S*s		
$\bar{X}$	0.90	0.53
σ^2	0.170	0.036
$\bar{X}$ difference	0.37	
t	5.21*	

†$n = 20$.
*$p < 0.001$.

true that a subject used the cue sheet only to learn the order of items in the test, and knowing what to type next, would not look at the sheet again until ready to type the next item.

Another consideration in selecting this particular material was a desire to minimize the effects of learning the test material. The test was the same at every trial, and it is assumed that the test material was nearly as familiar to subjects at the beginning of the experiment as it was at the end.

As for practice material, a list of 450 names and telephone numbers was compiled from a random selection taken from the San Francisco telephone directory. To relieve the monotony somewhat, subjects were also given a selected short prose passage as practice copy. The major part of the practice was, however, restricted to the list of names and telephone numbers.

The analysis of results was confined to input rates for subjects whose pretest scores on the standard machine were less than two strokes per second. This speed, which translates to a speed of approximately 24 words per minute by the conventional method of calculation (but without the conventional penalty for errors) *may* be considered a rate approaching a level of skill justifying the label "typist." But, because of the difficulty of defining typists and nontypists, this level was chosen as a convenient cut-off point in this study.

results

The following discussion will be concerned with only three sets of scores: the pretest, Trial Test 1 (given early in the experiment), and Trial Test 2 (given after seven hours of practice).

Pretest scores on both keyboards. Table 3 presents the pretest input scores, in strokes per second, made by the 40 subjects, each tested on both keyboards. It is probably obvious from Table 3 that the two 20-subject groups do not differ significantly from each other on either machine. Comparing machines, we see that all 40 subjects achieved a mean input score of 0.90 strokes per second on the standard machine and 0.53 on the alphabetical keyboard. The difference of 0.37 between these means is associated with a t of 5.21, which is significant beyond the 0.001 level. In other words, the hypothesis of equality must be rejected because the average input rate on the standard keyboard is significantly higher than the average performance on the alphabetical keyboard.

Although the pretest results are interesting in themselves, the purpose of the study was to determine on which keyboard subjects could type faster after some practice. Accordingly, emphasis will now be shifted to the progress made, after practice, by subjects on their assigned machines.

Table 4 Test results

Test	*Group*	
	Standard	*Alphabetical*
Test 1	1.11	0.54
Pretest	0.91	0.88
Difference†	0.20	−0.34
t	1.79*	4.57***
Test 2	1.58	0.83
Pretest	0.91	0.88
Difference†	0.67	−0.05
t	3.51**	1.04*

Note: The pretest scores are those made by both groups on the standard machine, while Test 1 and Test 2 scores are those made by each group on the practice machine.
†The difference in each case is obtained by subtracting the pretest from the trial tests.
*ns, $p > 0.05$.
**$p < 0.01$.
***$p < 0.001$.

Results of Tests 1 and 2. Table 4 presents the scores on Trial Test 1 and Trial Test 2. The effect of practice has been assumed to account for the differences between the pretest and trial test scores. Thus, analyses to follow were made to determine to what extent each group was able, after practice, to achieve input scores (on the practice machine) equal to or greater than their own pretest scores on the standard keyboard. In other words, we shall not be concerned with each group's improvement, as such, on the machine used for practice.

Standard Machine Group. If we consider, first, the performance of the group on the standard machine, it will be observed that there was a nonsignificant (0.20 strokes per second) improvement, since the t of 1.79 indicates that the difference between the pretest mean and Trial Test 1 mean is not significant ($p > 0.05$).

Alphabetical Machine Group. When, next, the performance of the alphabetical group is examined, it can first be seen that their Test 1 scores were lower than their own pretest scores on the standard machine. Moreover, the difference between the two test scores is statistically significant ($p > 0.001$). In other words, the practice was insufficient to raise the scores of this group on the alphabetical keyboard to a point equal to their own pretest scores on the standard machine.

Table 4 also summarizes the scores used to compare Test 2 and the pretest scores. After approximately seven hours of practice, the standard machine group achieved an input rate that was a significant improvement over their pretest rates. Subjects in the alphabetical group, however, even after approximately seven

hours of practice, were still not able to type on the alphabetical machine as fast as they were able to type, *without practice*, on a standard machine. However, the magnitude of the difference between the scores in Test 2 and the pretest is, for this group, statistically nonsignificant. Accordingly, it may be inferred that, while the standard group improved significantly after seven hours of practice, it took this much practice for the alphabetical group to make their scores on the alphabetical machine equal to their pretest scores on the standard machine.

conclusion

The conclusion drawn from these results is that the alphabetical keyboard is certainly not better than, and may not be as good as, the standard keyboard for relatively low-skilled typists.

Some time after this study was completed, a replication of sorts was conducted at Bell Telephone Laboratories in which a somewhat broader population was tested, using subjects with various typing skills, ages, and backgrounds. The results obtained by Michaels[14] confirmed the findings reported here, not only for the equivalently unskilled operators, but also for the other skill levels as well. The general conclusion was "An alphabetically ordered keyboard showed no advantage over the standard typewriter arrangement in output rate, error rate, and the speed of learning. Operators with little or no typing skill . . . were as fast or faster with the standard keyboard."

A probable explanation for the superiority of the standard keyboard may be that, although not a perfect arrangement, the key array of the standard typewriter is also not a random one. Whatever its limitations, it was "human-engineered" even as early as 1873, and many of the most frequently used letters are, generally speaking, clustered in the center of the keyboard. Hence, hunting for a letter can usually be confined to a relatively small visual area. Another possible explanation may be that the alphabetical keyboard probably requires, first, a memory search to locate the letter in its approximate or relative alphabetical position, and then a visual search to find the key on the board (where it is situated without regard to the frequency of its use). Accordingly, the combination of the memory plus visual searches may be less efficient than a purely visual search where the probability is high that the visual area first scanned will contain the sought-for letter.

One caveat: The input material used in this study was highly meaningful. That is, the words typed included letters and letter combinations that approximated the frequency of use considered in the layout of the standard machine. It is not, however, obvious from the study that material less meaningful (such as stock symbols, random letter sequences, etc.) would necessarily yield the same results.

Japanese language typewriter

Many centuries ago, when the Japanese established a written language, they adopted the pictographs that the Chinese had developed. Each "word" was a picture—"tree" was a picture of a tree; "sun" was a circle with a few lines radiating out from it; "fish" was a drawing of one—thus, there were as many pictographs as there were things to be written about. The number rose to the tens of thousands (the largest dictionary contains approximately fifty thousand). Over the centuries, the pictures were gradually stylized to simplify writing them; the result is a lexicon of ideographs that are not as pictorial as the originals they represent but are somewhat easier to write.

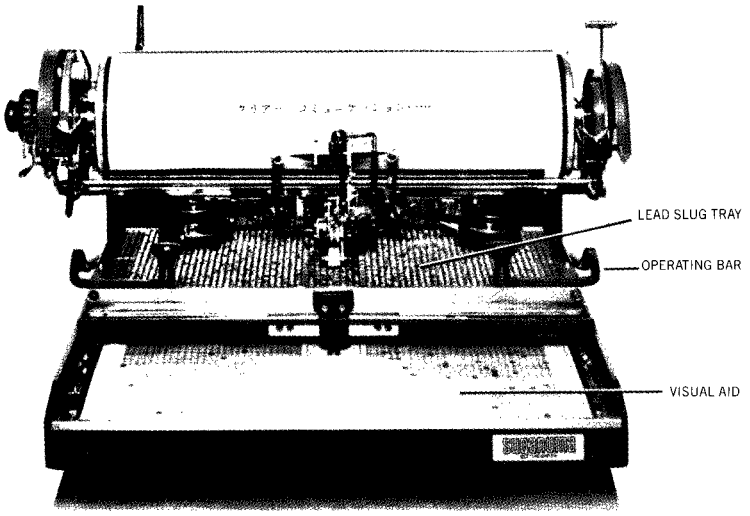
Despite the large number of Chinese characters (called *Kanji* by the Japanese), the Japanese were unable to find an appropriate pictograph for a number of spoken words in their language. Accordingly, in order to write those words also, the Japanese developed a syllabary in which each of the approximately 80 sounds is represented by a unique symbol. Those symbols, called *Katakana* characters, are used to write the sounds of "foreign" words or even the sounds of the Kanji characters.

At this point the reader may wonder why the relatively limited number of Katakana characters is not universally employed to substitute for the very large number of Kanjis. The answer lies in the homophone problem—many Kanjis, each a different symbol and meaning, are identically pronounced. English is not immune to homophones (for example, to, two, and too), but the distinctions of spelling help to distinguish meaning. In contrast, the Katakana symbols for the sounds "koka" can refer to 24 different Kanji ideographs, each having widely different meaning. So, abandoning Kanjis in favor of Katakanas is not the answer to the large character set. In fact, written Japanese is *enlarged* by the addition of Katakanas to Kanjis.

A further language refinement was the development of still another set of approximately 80 symbols, called Hiraganas, that, in combination with Kanjis and Katakanas, provide syntactical embellishment, such as formation of adjectives, adverbs, etc. Added to the Kanji, Katakana, and Hiragana characters are the frequently used alphanumeric characters. Thus, a typical Japanese sentence consists of all four language components, with approximately 35 percent coming from the large Kanji set and the remaining 65 percent represented by mixtures of Katakana, Hiragana, and alphanumeric.

In addition to handwriting or typesetting, sentences can be written by use of a "writing machine" (Wabun in Japanese), which is

Figure 9 The Wabun typewriter

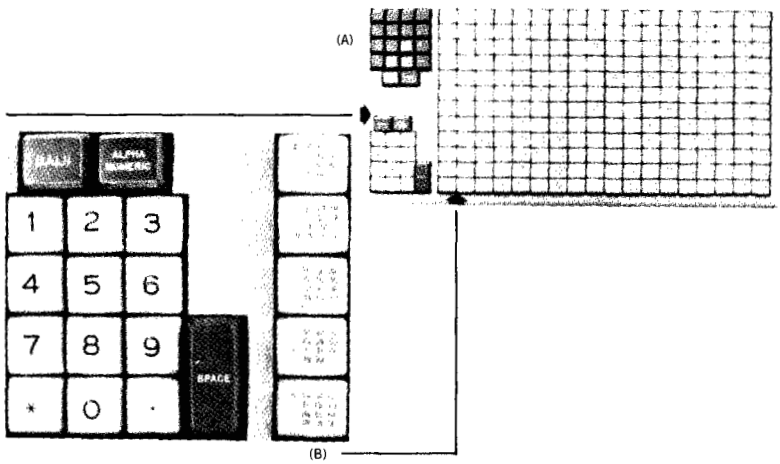


in reality a modification of setting type (Figure 9). Wabun consists of a large tray of lead slugs (about 2000 to 3000) and a "picker" mechanism to which the platen is attached. The operator searches for a character, and by movements of the tray and picker at right angles to each other, positions the picker over the character in the tray so that by pressing a lever the slug is made to strike the paper on the platen, after which the slug is reseated in the tray. It is obviously not a particularly speed-driven operation. To become skilled, operators train for six months to achieve "typing" rates of approximately 30 characters per minute.[15] Champion typists, using "tuned" Wabuns and well-practiced material, have achieved rates as high as 50 characters per minute.

To provide a means by which to increase the speed of typing and to make the output machine-readable, the IBM laboratory in Fujisawa, Japan, developed an electronic keyboard consisting of 216 keys, each containing 12 characters, each selected by 12 "shift" keys, thus providing a capacity of 2592 characters, approximately equivalent to Wabun capacity. Although the four language components (Kanji, Katakana, Hiragana, and alphanumeric) are all contained in the device, it will be referred to here simply and for convenience as the "Kanji Keyboard" (Figure 10).

The advantages of the Kanji Keyboard over Wabun are obvious: The operator "reads" the Wabun tray upside down, the Kanji Keyboard is read right side up. Wabun slugs are lead-on-lead; hence, contrast between the character and its background is minimal. On the Kanji Keyboard characters are printed in black on the white background of the key. And characters on the keys of

Figure 10 The multishift keyboard used in the study: (A) Top view of the entire keyboard; (B) Lower left-hand corner of the keyboard magnified to show character and shift keys



the Kanji Keyboard can be larger than "life size"—all of which contribute to improved readability. Finally, there are none of the mechanical movements of the Wabun in the operation of the Kanji Keyboard.

the problem

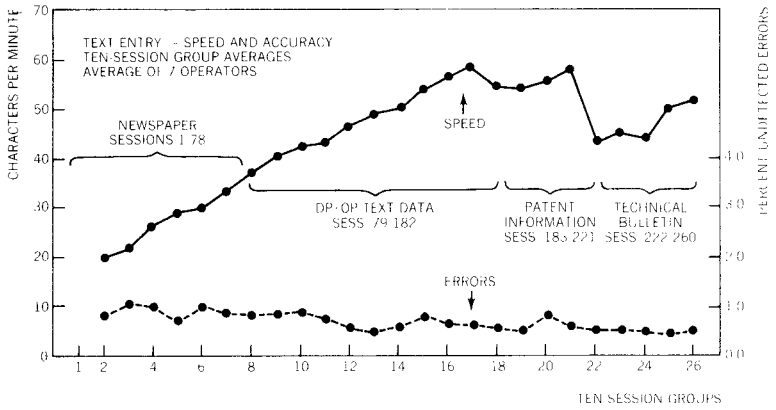
Despite the obvious advantages, the degree of superiority of the Kanji Keyboard over Wabun was not known. Accordingly, among the many human factors issues (size of keys, character size and placement, etc.) investigated[16] was the question, "How much better is the Kanji Keyboard than Wabun?"

Two issues arose at this point: What kind of test subjects would be adequately representative of the Japanese operators? What input material would be typical of the applications for which the keyboard would be used?

the end users

Discussions with planners and marketing personnel established that the end users would not possess any particular or unique skills or characteristics. Furthermore, even typing skills were not required because the number of skilled typists in Japan is limited. Accordingly, we advertised for Japanese women who were born in Japan, who had been educated in Japan, and who could pass a test proving that they could still read and write Japanese at least at high-school level. Ten operators were initially employed, but only seven remained until the end of the study. Three operators dropped out reportedly for personal reasons unrelated to the experiment. Each operator worked a four-hour day. They practiced on the keyboard almost from the beginning and continued doing so for the next three months.

Figure 11 Average speed and accuracy of all seven operators; data points are the averages of 10-session groups



the keyboard

Fabricated by the Fujisawa laboratory, the keyboards, as previously indicated, had a capacity of 2592 characters. The board had to contain 1968 basic (government-decreed) Kanji characters, and to that number were added another 300 frequently used Kanjis, the Katakana, Hiragana, alphanumeric, punctuation, and some special symbols—in all, 617 additional characters, making a total of 2585 characters.

input material

One set of material keyed by the operators consisted mainly of text taken from newspapers, magazines, company memos, patent documents, and technical bulletins. A second set consisted of names, addresses, birth dates, and related personnel data taken from the records of approximately 25 000 persons listed in the Japanese "Who's Who."

results

The primary result compares performance on the electronic keyboard with the lead-slug mechanical Wabun. The answer can be stated in brief: Whereas Wabun operators take six months to achieve an input rate of approximately 30 characters per minute, the operators on the electronic keyboard were able to achieve an average of 60 characters per minute after only three months, or in more general terms, half the time for an average of two times the input rate.

Figure 11 presents the average speed and accuracy of all seven operators. Each data point is the average of a 10-session group of scores. What can be seen is that there was a rapid rise in speed from almost 20 characters per minute at the start to something over 60 characters per minute toward the end of the study. It may be noted that the introduction of relatively unfamiliar terms (technical bulletins and patent information) produced a decrease in

speed, which should be expected. The figure also shows that the error rate was relatively stable (approximately 0.5 percent) almost from the beginning, also not surprising in what is essentially a hunt-and-peck operation.

Unfortunately, the figure, consisting as it does of the average rate for all seven operators, obscures what was achieved by individual operators. Some of the operators entered data in the range of 75 to 90 characters per minute—up to three times the Wabun rate in about half the learning time.

Typewriter feedback for blind operators

Sighted typists normally discover typing errors by proofreading the line or page of typewritten output. If the operator *thinks* an error has been made, a simple glance at the printed output establishes whether or not a mis-key has been made. Past pilot studies have indicated that operators look at their output for so-called detected errors about once every 30 seconds.[17] Accordingly, it is clear that *visual* feedback is significant to the production of error-free copy.

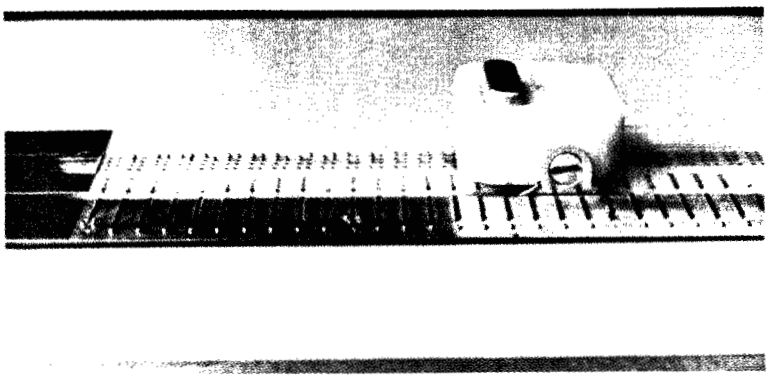
By definition, visual feedback is not available to blind typists. As a consequence, producing error-free copy is extremely difficult and time-consuming for them. There are some devices available to assist blind typists to "proofread," but none is easy, quick, or unobtrusive.

One approach developed at the IBM laboratory in Raleigh, North Carolina, consists of a single Braille cell "display" generated by a 3×3 array of pins driven by magnets to produce individual Braille characters corresponding to the characters stored in a card of an IBM Magnetic Card typewriter. The Braille display is located at the top of a plastic pyramid that sits on a platform which houses the magnets that drive the pins. The pyramid also serves as a hand rest for the blind reader. The pins display the last character typed when the magnetic card is played back in a single-character mode. It is also possible to go back to the beginning of a line or page and read out what was typed, character by character. In this way the typist can detect keystroke errors. Although slow and limited to Magnetic Card typewriters, it is used by several blind typists in IBM.

the problem

The purpose of this study[18] was to determine whether blind typists could learn to use two new devices and to determine which device the typists thought would be of greater help to them. One device, called TOUCHLINE, was developed in the IBM laboratory located in La Gaude, France, and is a modification of the single-cell Braille display. The other, called AUDIOLINE, is from the IBM laboratory in Los Gatos.

Figure 12 Closeup of TOUCHLINE Braille display, showing pins in Braille position and the erase shoe



Performance data were collected to isolate possible sources of difficulty (in the initial learning and after several days of practice) on each device, and to provide feedback to us as well as to the typists.

As described in the two preceding examples, the Human Factors Center normally puts emphasis and reliance on *performance* data when comparing alternative devices or procedures. In this case, however, important as performance may be, it was probably equaled or exceeded in importance by how the operators subjectively reacted to the devices. Our reasoning was that since the operators were already skilled typists, typing performance *per se* was unlikely to be affected by the devices. Rather, feedback from the devices was likely to affect only their "proofreading" for error correction. Although the measurement that might reveal device differences would be accuracy and time in error correction, we soon realized that there was more to the feedback than merely throughput assistance. Specifically, the feedback has a major psychological component, namely how the operator *perceives* being helped by the feedback. In short, we concluded that subjective reactions and responses from the typists had to be given as much, or even more, weight in the evaluation than was given to performance data.

the TOUCHLINE approach

TOUCHLINE can be considered a logical extension of the single Braille cell approach. As the typist types each character on the paper, a Braille character is simultaneously generated on the TOUCHLINE display. Any time the typist wishes to review what has been typed, or feels that an error may have been made, it is possible to read immediately in Braille the line that was typed.

Figure 12 is a closeup view of the tactile display of TOUCHLINE. The top surface of the box contains the line tactile display of 60 Braille characters with a 6-mm pitch. In front of the display is a

(tactile) column number scale. As typing progresses, a part of the display that combines the functions of a column position-indicator and an erase shoe (hence, called a position indicator erase shoe) moves along the display and remains over the last Braille character generated. When checking for an error in the last character, the typist can move the position indicator erase shoe out of the way by depressing an actuator bar on the top front surface of the display box. When the carriage return on the typewriter is pressed, a cam is engaged that lowers the erase shoe to automatically erase the Braille display during the return cycle.

Error correction for TOUCHLINE is as follows: First, the character in error is located by reading the Braille display. The position indicator erase shoe is positioned over the character by using the backspace correction key, which positions the lifting tape on the typewriter. The Braille character is erased by pushing on the top of the erase shoe, which pushes down the pins that form the Braille character. The key corresponding to the character in error is depressed, lifting the character from the paper. The resulting Braille character can then be checked, if desired, to make certain the proper correction was made. Next, the Braille character is again erased and the correct character struck on the typewriter. The new Braille character can again be checked if the typist wishes.

These operations have been described in some detail to help the reader understand that although not really complicated, TOUCHLINE involves a certain number of hand operations.

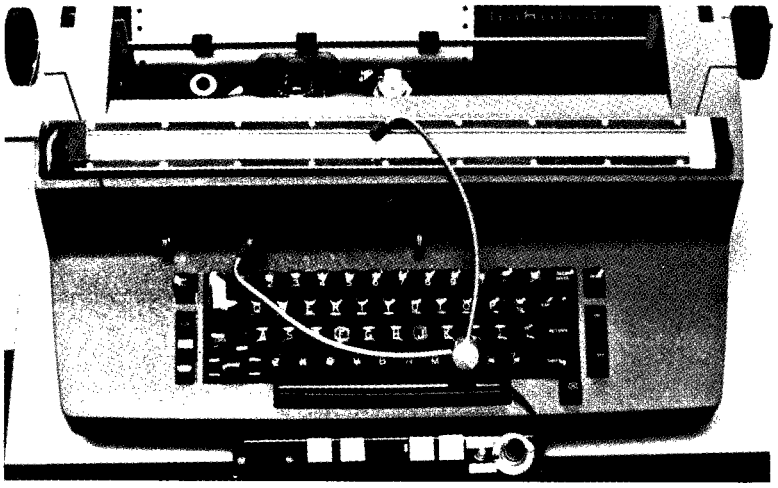
the AUDIOLINE approach

Basically, the AUDIOLINE device provides typists with audio feedback, allowing them to review the last character(s), the last word(s), or the whole line typed. (The product that was developed is called the Audio Typing Unit, but we knew the prototype by the name AUDIOLINE and use it here because it is a convenient contrast to its alternative, TOUCHLINE.) The engineering model employed in this test consisted of an IBM Correcting SELECTRIC® typewriter, a Votrax® voice synthesizer, and a microprocessor. The functions of AUDIOLINE were controlled either by normal keys on the typewriter or by a row of pushbuttons located immediately below the keyboard on the front surface of the typewriter cover.

Figure 13 shows a closeup of the pushbutton controls on a typewriter equipped with AUDIOLINE. The functions available to the typist through the pushbutton controls are:

1. Last character(s) review
2. Last word(s) review—spelled out
3. Line review—spelled out
4. Last word review—phonetically spoken
5. Line review—phonetically spoken

Figure 13 Closeup of AUDIOLINE function keys



6. Expanded letter, i.e., "B" = "bravo," "P" = "Papa," etc.
7. Distance to move the type element back (or forward) to reach a detected error in the line
8. Position at which the type element is located
9. Line number announcement
10. Ability to start and stop the audio as a line is being reviewed
11. Option to hear each character as it is being typed (echo mode)

The space bar, backspace, and correction-backspace keys on the typewriter also provide audio review capabilities by vocally stating which character the type element of the SELECTRIC® typewriter is moving over as these keys are pressed.

Error corrections on AUDIOLINE are made by first locating the error either by automatic feedback or by the review functions. Automatic feedback is given in cases of overstrike or accidental hyphens. The backspace correction key is used to position the type element over the character in error and to position the lifting tape. AUDIOLINE repeats the character (or characters in the case of an overstrike) to be lifted. The key thus indicated is then pressed, lifting the character from the paper. If the proper correction is made, AUDIOLINE announces "OK." If not, AUDIOLINE says "correction error" and tells what partial character remains. After hearing "OK," the typist can enter the correct character.

results

The results of this particular study will be presented in two parts. Performance data will be presented first and will be followed by the operators' responses to a questionnaire.

Performance results. As the typists started the evaluation, each was assigned, at random, to either TOUCHLINE or AUDIOLINE for two days. Then each one was moved to the alternate machine for another two days. On the fifth day—called the *Test Day*—each typist used both machines, the order in which the machines were used again being alternated between typists.

The six typists whose performance data are discussed here were, generally speaking, trained typists. Two of the six were IBM secretaries and the other four were typists skilled enough to qualify them for such employment. Our observation was that, although getting used to each device was not a formidable task, it did nevertheless take a certain amount of time. In retrospect, it is reasonable to conclude that the typists certainly learned to use each device in two days, but that two days were not enough to become completely expert in their use. Despite the probability that we are looking at performance data that are still not at the asymptote of the learning curve, certain conclusions can nevertheless be inferred from the fact that there do not appear to be any significant performance differences between the two devices.

The observed differences were relatively small and the total error rates (combining both detected and undetected error rates) were not significantly different. In other words, there is nothing in the performance data to suggest significant performance differences between the two devices. The conclusion, then, is that blind typists at this early stage of learning are able to perform on both devices at a satisfactory level. The undetected error rates compare favorably to those of sighted typists.

As mentioned earlier, the subjective data obtained from the typists overshadow all other considerations that pertain to performance so much, especially in the absence of performance differences, that a discussion about performance becomes relatively unimportant.

Subjective results. The typists were questioned on their arrival at the Human Factors Center about their backgrounds and attitudes about typing aids. Although all the typists had substantial experience with Braille, it is interesting to note that proofreading, even with aids like the Braille Cell, was still a source of some frustration. At the outset of the study, an attempt was made, in an informal and unstructured manner, to elicit opinions about what their attitudes might be concerning the use of audio feedback versus Braille feedback. In every case the response was in favor of Braille feedback. It is not a surprising response, because they knew and used Braille, but were inexperienced with audio output. Apparently, they were not even able to conceptualize audio as a method of feedback.

Following the practice and test sessions, we conducted an interview session (questionnaire) with the typists during which their attitudes about the two devices were reviewed. The best way to describe the result is that it was a love affair with AUDIOLINE. This is not to say that they considered TOUCHLINE as no good or useless. On the contrary, they recognized the merits of the Braille approach, but they regarded AUDIOLINE as having significantly greater advantages.

The most important advantages include the following:

- AUDIOLINE is relatively unobtrusive. It appears that the blind are sensitive about their blindness and do not like it made obvious by the use of large noisy equipment.
- The typists do not have to take their hands from the home row of the typewriter to identify their typing errors and to verify that the corrections have eliminated all errors.
- They particularly appreciate the reassurance that AUDIOLINE gives when it says, "Correction Error Partial" for incompletely (or incorrectly) corrected errors, and "OK" when the correction is successfully completed. This type of feedback was of great importance to them.
- Finally, the greater flexibility and greater number of tested (and potential) functions were highly rated.

CRT display legibility with reduced character size

The most frequently encountered character size in common printing provides upper-case character heights of approximately 2.54 millimeters (0.100 inch). This is the nominal size of most typewriter typestyles and has also been a standard size for computer printing. Virtually all books, journals, and popular magazines employ type of that size. The reason this character size is so ubiquitous is that it is considered comfortable for general reading as well as being legible for people with normal vision.

But most CRT (cathode ray tube) display terminals employ larger characters than the usual printing size. The characters are generally formed by selecting elements from a fixed character matrix, such as 7×9 elements. Early CRT displays had rather low resolution and poor contrast; hence, larger characters were necessary for legibility. However, there is today a much improved display quality and often a desire to increase the character capacity of displays.

the problem

The number of characters displayed per line, the number of lines, and the interline spacing determine the size of CRT required for any given character size. Character size also has human factors implications related to the display size. The principal human factors effect of large CRTs is related to the increased elevation of

their uppermost line. This leads to increased glare from overhead illumination and increased head tilting, especially for users wearing multifocal eyeglasses. Another effect of large CRTs is that they require greater depth, which usually reduces the available viewing distance with normal table depths. This increases the visual angle subtended by the display, requiring increased amplitude of eye and head movements. Consequently, character size can be a very critical parameter for applications that require page-size displays. The purpose of this study[19,20] was to measure any differences in the readability (speed) or legibility (accuracy) of upper- and lower-case letters displayed at their normal size and at the smaller size common in ordinary printing.

subjects

The operators employed in the study were 12 skilled typists obtained for one day each from a temporary employment agency. "Acuity Chart for Near" (Good-Lite Company) was used to check their near-distance visual acuity. The test required reading the smallest-size Sloan letters under an illuminance of 160 lux. Distance was not controlled since an operator's distance from a display is not normally controlled. All operators tested had normal or better near visual acuity (with corrective lenses where applicable).

procedure

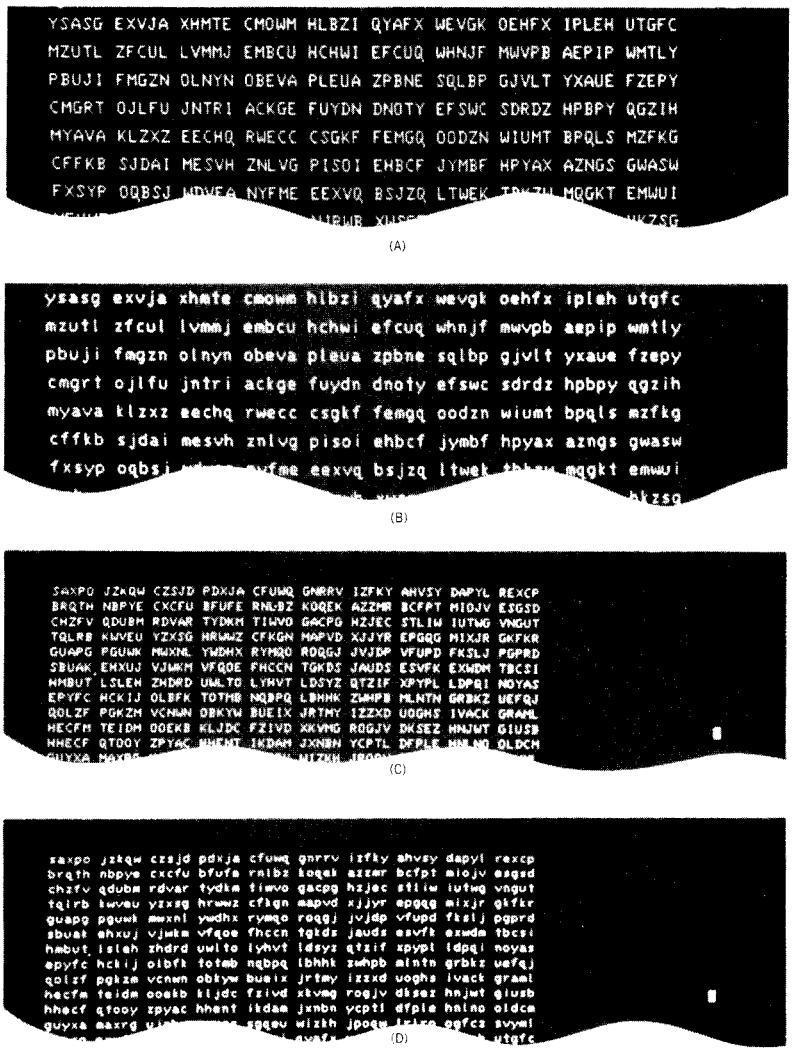
Two IBM 3277 (Model 2) Display Terminals with U.S. EBCDIC dual-case character sets were connected to the Human Factors Center's System/7 computer via special keyboard interface devices. Raster size adjustments made possible the display of upper-case characters 3.43 millimeters (0.135 inch) high on one display and 2.54 millimeters (0.100 inch) on the other. The larger size corresponds to the size of the standard upper-case characters normally used with this terminal. The display screens were in their normal vertical orientation. Both displays were adjusted to produce a full raster luminance of 50 cd/m^2 (candela per square meter), and the ceiling illumination was adjusted so that the illuminance, measured perpendicularly to each display, was 160 lux. Upper-case characters were seven elements high, and lower-case letters were no less than five elements high.

Test "pages" were read from a direct-access disk storage unit. Three unique pages were allocated to each session. Every page consisted of 20 lines each with 10 "words" of five random characters and a space. Figures 14A to 14D are photographs of the four treatments. These treatments will be designated by the shorthand notation of CASE-SIZE where CASE is either UC (upper case) or LC (lower case) and size is either N (normal) or R (reduced).

system operation

The program provides several phases of operation that will be described in their normal order. The operator performs the tests by keying each display line followed by the RETURN key. RETURN causes the uppermost line to be removed from the display. After

Figure 14 Portions of the four displays (treatments) used in the study: (A) upper case in normal size; (B) lower case in normal size; (C) upper case in reduced size; (D) lower case in reduced size



the last line is entered, the next "page" is presented. A session is terminated by the next RETURN key depression that occurs more than 10 minutes after the beginning of the session (first keystroke). Error-correction features will be described below. All key entries and entry times are also recorded on data tape.

The next phase was error classification. Each line containing one or more errors undetected by the operator was displayed with the original line above the entered line. Discrepancies were designated with asterisks displayed on a third line. The display also indicated an audit trail of all key entries and interkeystroke inter-

Table 5 Latin square experimental design

Treatment order	*Period of day*			
	1st	*2nd*	*3rd*	*4th*
A	LC-R	UC-R	LC-N	UC-N
B	LC-N	UC-N	LC-R	UC-R
C	UC-N	LC-N	UC-R	LC-R
D	UC-R	LC-R	UC-N	LC-N

Notes: Three operators were assigned to each of the four indicated orders of treatment by period sequences. LC = lower case; UC = upper case; R = Reduced (0.100″ height); N = Normal (0.135″ height).

vals to assist the reconstruction of the error history. Errors were then classified as legibility, typing, or others.

If the error was of any type other than a single substitution of one character for another, it was assigned to the "other" category. If the error was a substitution, it was then classified from a confusion matrix. The matrix indicated for each possible substitution one of the three categories based on our prior experience. The typing and legibility categories were only assigned when both were not plausible, and when both were plausible this category was also assigned to "other."

After the experimenter had classified the last error, statistical treatment of data was automatically initiated. The display presented a session summary of keying rate and error rate, classifying the errors according to error category. The keying rate and error rate values were then plotted on the operator's progress graph, and the session summary statistics were retained on the data tape to complete the session cycle.

The experimenter first demonstrated the terminal operation to the operators. They were particularly shown that keyed characters did not normally appear on the display, but that keying the "DISPLAY" key presented the line last keyed on the lowest line of the display. They were also informed that keying "BACKSPACE" moved the cursor one step to the left for each depression and removed the character above the cursor. Characters then entered were retained on the display until "SPACE" had been keyed. Simple error corrections without using the display were demonstrated. After these demonstrations, operators went through a practice session together with the experimenter, who provided remedial guidance as needed.

The operators performed in four experimental treatment periods, each treatment consisting of five sessions. The four treatments consisted of working with combinations of upper and lower case and the two display heights. The operators were assigned to the Latin Square design shown in Table 5. It can be seen that the

Table 6 Performance results of 12 operators

Performance measures	*0.100 inch height*		*0.135 inch height*	
	Lower case	*Upper case*	*Lower case*	*Upper case*
Total keystrokes	96,282	93,911	94,766	95,627
Effective keystrokes per second				
Mean	2.586	2.519	2.541	2.561
S.D.	0.425	0.396	0.423	0.383
Operator-detected errors (%)				
Mean	0.660	0.681	0.672	0.707
S.D.	0.366	0.361	0.379	0.362
Cleared keystrokes (%)				
Mean	2.959	3.019	3.158	3.176
S.D.	1.554	1.446	1.714	1.568
Clear time (%)				
Mean	6.59	7.19	7.49	7.34
S.D.	3.27	3.50	4.05	3.06
Legibility errors (%)				
Mean	0.1209	0.0841	0.0925	0.0722
S.D.	0.0714	0.0458	0.0539	0.0755
Typing errors (%)				
Mean	0.1973	0.2198	0.1943	0.2307
S.D.	0.1393	0.1463	0.1308	0.1396
Other errors (%)				
Mean	0.1102	0.0951	0.0753	0.1138
S.D.	0.0775	0.0693	0.0548	0.0583
Total undetected errors (%)				
Mean	0.4283	0.3990	0.3621	0.4167
S.D.	0.2179	0.1934	0.1895	0.2042

height change is always made in the middle of the treatments to balance order effects. After the final period, the operators were asked to state their preferred treatment by height and case.

results

From Table 6 it can be seen that the only measure that resulted in any statistically significant differences was Percent Legibility Errors. This is defined as 100 times the number of legibility errors divided by the effective keystrokes.

The usual way to analyze a two-factor design (case and size) is by considering each factor and its interaction with the other. However, it was decided *a priori* that the size effect was of primary interest for both cases. Orthogonal contrasts were defined for these two sizes by case effects, leaving a third orthogonal contrast of case difference averaged over both sizes. Designating these three *a priori* orthogonal contrasts permits testing the significance of each of these comparisons at any desired level of confidence.

Table 7 Analysis of variance table for percent legibility errors

Source of variation	*df*	*Mean square*	*F*	*Probability*
Between operators	11			
Order	3	0.0170	1.67	
Operators within order	8	0.010		
Within operators	36			
Periods	3	0.0010	0.78	
Displays	3	0.0051	3.97	$p < 0.05$†
U.C. size	(1)	0.0008	0.65	
L.C. size	(1)	0.0048	3.70	$p < 0.05$*
Case	(1)	0.0098	7.52	$p < 0.01$**
Pooled error	30	0.0013		
Latin square error	(6)	0.0013	1.00	
Within error	(24)	0.0013		

†$F_{0.95}(3.30) = 2.92$.
*$[t_{0.95}(30)]^2 = 2.88$.
**$F_{0.99}(1.30) = 4.17$.

Table 7 presents the analysis of variance of the Percent Legibility Errors variable. The Order factor is tested against the Operators-Within-Order and is not significant. The other factors are tested against a Pooled Error since the two error variances were homogeneous. The Periods factor was obviously not significant, suggesting that there were no appreciable learning or fatigue effects. The Displays factor was significant at the $p < 0.05$ level, although the orthogonal contrasts should be examined regardless of this significance.

The use of a complete set of orthogonal contrasts permits the partitioning of the entire Sum of Squares for the Displays factor into the three contrasts shown next. The upper-case size factor is obviously not significant. The lower-case size factor is tested against a one-sided hypothesis, since it is assumed that the size reduction cannot improve legibility. This is accomplished by taking the square of the critical t value at the 0.05 level, from which it can be seen that the null hypothesis must be rejected. Finally, the case factor over both sizes resulted in significance at the 0.01 level of confidence.

There are a variety of ways to compare the four treatment effects *a posteriori*. The Duncan Multiple Range Test[21] indicates that only the LC-R treatment is significantly inferior (at the 0.05 level) to the UC-R and UC-L treatments, which is expected since the case contrast was significant. Dunnett's test for multiple comparison against a control[21] may be used by regarding the UC-N treatment as a control. This test reveals that only LC-R is significantly inferior (at the 0.05 level). This result, like the Duncan test, indicates that there was no significant case difference for the larger-size characters.

The operator preferences tallied as follows: UC-N: 3, LC-N: 3, UC-R: 5, and LC-R: 1. There is little correlation between performance and preferences, and most operators had difficulty in deciding on their choices.

conclusion

The results may be summarized by stating that, when the VDT user was allowed to self-select viewing distance and when the task was the transcription of nonmeaningful character strings, the legibility of LC-R characters was found to be inferior to the other three conditions, and lower-case was inferior to upper-case pooling over the two sizes. While these were the only differences found to be statistically significant, we should examine their practical significance. It can be seen in Table 6 that the total error level of operator-undetected errors of all kinds was about 0.4 percent for all combinations. Against this value, the lower-case size difference of only 0.03 appears to be within the noise level. (It may be noted that the total error rate for the larger characters favored the lower case by a difference of more than 0.05 percent units.) The result of a statistically significant increase in legibility error rates for lower-case characters displayed at the smaller size is considered to be of trivial practical significance. Undoubtedly, contextual clues in normal text would remove such differences. However, it should not be concluded from this study that display terminals need not present characters any larger than the size generally encountered in printing. The key factor is the visual acuity of the user. There are undoubtedly a very significant percentage of potential terminal users whose near acuity is subnormal or inadequately corrected. There is the further problem of accommodation loss which is corrected with bifocal or trifocal eyeglasses. Decreasing character size may help reduce head tilt necessary to view the top of the display, but this may be a compromise due to reduced acuity at the natural display distance. It may be that glasses corrected for the most comfortable eye-to-display distance is the best solution for all users suffering from loss of accommodation.

Software studies

The earliest users of computer systems were people who were engineers, mathematicians, and scientists. Computer programs were, in many cases, developed by them; hence, their sophistication in the use of computers was reflected in the complexity of the programs and programming languages that they produced.

Despite the fact that the majority of those earliest users are still around (and working), an entirely new class of users has now emerged whose educational backgrounds and work experience require drastically simpler programming languages if they are to use computer systems effectively. As a consequence, the Human Factors Center has been applying and expanding the techniques

of hardware evaluation to the study of software designed for use by end users, that is, those whose work, profession, and background have not provided much, if any, exposure to the use of computers.

Some brief historical observations may be helpful at this point. One of our earliest efforts in this field was done over 10 years ago and involved an attempt to understand the fundamental characteristics of a programming language.[22]

Language definition was believed to be a good starting point for experimentation with a variety of language features. Several exploratory steps were taken. For example, an attack on the problem was made that attempted to implement and test a very simple string processing language, similar to LISP in that it used conditional expressions and recursive function definitions.

Some six months later the language had been defined and an interpreter implemented using the IBM 2760 Optical Image Unit as the primary terminal. This system was used to test the ability of untrained high-school students to program effectively using these concepts. An experiment consisted of a student attempting to program a fairly standard set of string and arithmetic functions starting from primitives. Changes to the system were made freely, depending on operating experience.

Emphasis was on language simplicity and generality rather than on execution efficiency. Linear character strings were the only data type, and no provision was made for explicit attribute declarations. Commands and functions were denoted by words that could be entered via the optical unit by simply pointing a light pen at the desired word on the screen. There was an equivalent syntax using special characters for built-in functions and commands that allow reasonably fast entry on a typewriter, but with a bigger burden on the programmer to remember the symbols and their meaning.

The basic syntax was parenthesis-free, and reserving of characters or words was minimized. "Blank" was the basic delimiter between words. A command roughly corresponded to an executable statement in PL/I. An attempt was made to bring in so-called system commands (JCL) at the same syntactic level. A program was a character string that contained a sequence of commands along with appropriate operands that could be function expressions.

Primitive functions were concatenation, taking the first character of a string, taking all but the first character, testing equality, and a simple conditional expression. Elementary functions were programmed using these primitives (logical true was denoted by the

character T, false by F, and undefined functions returned a null string). All functions were recursive. In addition to the functions, a small set of commands was provided, including assignment, function definition, command definition, and branching. A reasonable scheme was defined for the scope of variables, function, and command names in well-nested programs, but a similar scheme was not completely satisfactory for labels used as branch targets.

This effort is described in detail above to underscore how considerable and complex such an effort becomes before a single test subject can be employed to evaluate the concepts that motivated the work. It took nearly a year of program development to reach the point where evaluation could begin, even on an elementary level. Although the interpreter appeared fast enough for the simple experiments planned, it was frustratingly slow when compared to what we have all come to expect of computer operations. Also, although the original design was used in a few exploratory experiments with young students programming basic arithmetic and string handling functions, it was clear that many major extensions, modifications, and variations of the design were necessary before any definitive studies could be conducted. The big disappointment in all of this was that so little of applicable value emerged from so much work.

A few additional observations from that experience are worthy of brief mention. In order to evaluate the human factors of a programming language, the language had to exist or development of a programming language was the necessary first and laboriously long effort. Also, emphasis shifted during the course of the work because it was, after all, an exploratory effort. As a consequence, in order to address the multitude of variables characteristic of programming languages, experimental effort followed a frustratingly zigzag course. The result was a decision to terminate the effort in favor of an approach that, while less ambitious, might nevertheless yield more near-term positive results. As a consequence, we do not consider all that effort wasted; on the contrary, it served to focus our attention on more realistic goals.

Another observation is that work with this objective may be best aimed at the nonprofessional programmer, who needs a lot of help. The nonprofessional, however, usually performs work in a highly application-dependent environment. As a consequence, if one concentrates on the nonprofessional, it may be hard to generalize the results.

There have been, and are, attempts to monitor by objective means the work of professional and nonprofessional programmers. The aim of the monitoring efforts is to learn something fundamental about programming that will affect performance. These

efforts are usually hampered by the fact that it is difficult to affect the structure of a language independently of the system in which it is embedded. The result of such efforts, has been, at best, the development of some training procedures that help the user to learn to live with the system. The efforts rarely result in any fundamental change to the system—or indeed, in even getting to know anything fundamental *about* the system.

The first software example reported here deals with what seemed, at first glance, to be a trivial question: In viewing successive "pages" of data on a display, should you scroll (that is, move) the data or move the window in which the data appear? As will be seen, when even so simple a question is investigated, relatively complex issues emerge.

The second example describes an attempt to compare the user efficiency of two text-editing systems.

Moving data: windowing or scrolling

To study a different star, the astronomer moves his telescope. To study a different bacterium, the biologist moves his microscope slide. The viewing instrument is being moved for the astronomer, whereas the viewed object is being moved for the biologist. These scientists have no choice; the nature of their equipment requires that they operate in a predefined way. The user of a video display terminal (VDT), however, can be given a choice. The VDT user views a representation of an area of computer memory. In most cases the portion of memory the user wishes to see is much larger than what will fit on the screen at one time. For this reason almost all VDTs are equipped with some sort of *scroll function*. The scroll function allows the user to display data that are located beyond the limits of the screen.

One way of conceptualizing the use of the scroll function is to visualize the display of data as if it moves (scrolls) behind the stationary VDT. This would be analogous to the biologist moving a slide beneath his stationary microscope. If a user were operating under this mode, and wished to display data currently beyond the upper limit of the screen, he would use the *scroll down* command. This command has the effect of moving the *data* down to bring the desired information into view. Similarly, to view data that are currently beyond the left border of the screen the user would use the *scroll right* command, which would move the data to the right.

The alternate way of conceptualizing the scroll function on a VDT involves visualizing the display screen as if it were a movable window through which the stationary data could be viewed. This, of course, is analogous to the astronomer moving his telescope

over a (relatively) stationary star field. In this case, for the VDT user to display data located beyond the upper border of the display screen, he would use the *window up* command. This would have the effect of moving the *window* up, which would again bring the desired information into view. Similarly, to view data beyond the left border of the display screen, the user would use the *window left* command.

Thus, giving the command to move up, down, left, or right will have entirely different results depending on whether your command is interpreted to mean to move the data or to move the window. For the purposes of this study, these two modes will be referred to as scrolling and windowing, respectively.

the problem

The use of the windowing or scrolling mode varies among VDT systems. Examples of each can be found even within the product line of any one manufacturer. The ideal would be to enable the user to select the preferred mode, as some systems do, but even then the question is which mode to make the default condition if one or the other mode is not preselected. Although the benefit of standardizing all VDT systems to one of the modes may be obvious, which should be chosen is not. While each has its own intuitive advantage, no empirical evidence could be found to suggest superiority of one mode over the other. In fact, it seems quite possible that neither is actually superior. If this is indeed the case, all that may be needed is an arbitrary standardization. The purpose of this study[23] was to answer the following questions:

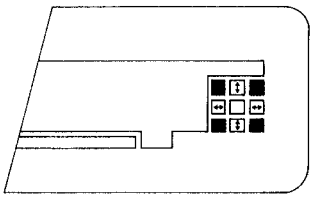
- Do novice users have a natural "bent" in their performance toward windowing or scrolling?
- Is performance in one or the other mode more efficient?
- Does training in the conceptualization of either mode improve performance?

procedure

A total of 188 eleventh and twelfth grade high-school students were subjects in this experiment. None of the subjects had experience moving data on a terminal screen, although some had used computer games of various sorts, and some had used hard-copy terminals.

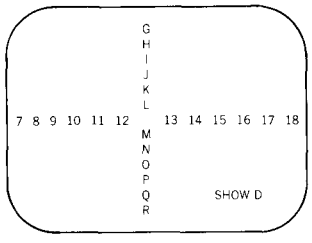
An IBM 3277 keyboard and a Mini-Tec® data screen terminal (made by Tec Incorporated) were used as the input and output devices, respectively. They were connected to our System/7 computer, which controlled stimulus presentation to the subjects and gathered data on speed and accuracy of performance. Subjects communicated with the computer using *arrow keys* (the 5, 7, 9, and 11 PF keys of the 3277 keyboard), and the space bar. All other keys were covered with an overlay. A diagram of the keyboard is shown in Figure 15.

Figure 15 IBM 3277 keyboard layout (with overlay) used in Experiment 1



The keys were arranged to approximate a joystick with double-headed arrows to indicate the axes of key control (vertical or horizontal). Direction of display movement (up, down, left, or right) was inferred from the position of keys relative to each other. If the subject pressed the left key to display data on the left, then the result was windowing. Pressing the right key to display data on the left was scrolling. That the subjects understood the relationships between key position and display movement is inferred from the results obtained.

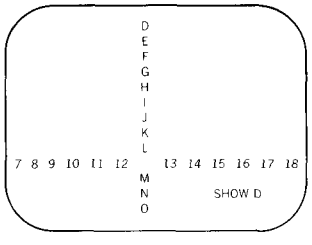
Figure 16 The display in its initial centered position



Subjects were randomly assigned to five different treatment groups. The five groups were:

1. Self-defined. Subjects were allowed to self-define the system in whichever mode they wished to operate. No explanation of windowing or scrolling concepts was given.
2. Window/concept. Subjects were constrained to operate in the window mode. Prior to testing, subjects were given an explanation and a demonstration of the window concept.
3. Window/no concept. Subjects were constrained to operate in the window mode, but were given no explanation or demonstration of the windowing concept.
4. Scroll/concept. Subjects were constrained to operate under the scroll mode. Prior to testing, subjects were given an explanation and demonstration of the scroll concept.
5. Scroll/no concept. Subjects were constrained to operate under the scroll mode, without an explanation or demonstration of the scroll concept.

Figure 17 The display after pressing any one of the arrow keys



The scenario for the groups was as follows:

Self-defined group. After reading a set of instructions, the subjects in the self-defined group were presented with the display illustrated in Figure 16. Subjects were required to display letters and numbers that were currently beyond the limits of the screen, in this case, the letter D. The subject pressed whichever arrow key that he felt would accomplish this task. The computer responded with the display change as shown in Figure 17. No matter which key the subject pressed, the letter D was displayed (i.e., no choice was wrong). The decision of which key to press depended entirely on how the subject conceptualized the task. After displaying the desired letter during this instructional phase, the screen was automatically reset to its initial "centered" position. Each of the four directions of movement was defined in this manner. For each direction, the subjects were required to display a character (D, T, 5, or 20) by pressing what they felt to be the appropriate arrow key. Each subject was able, by this method, to define what function each of the arrow keys would have for *him* or *her*.

Concept groups. Subjects in the window/concept and scroll/concept groups were given written instructions and demonstrations of windowing or scrolling.

No-concept groups. These groups were given the written instructions of what to do but no explanation or demonstration of the windowing or scrolling concept.

After receiving appropriate instruction (and in the case of the self-defined group, going through the self-definition phase), all subjects performed 80 display problems. For each problem the subject was asked to display a letter *and* number simultaneously (i.e., SHOW X and 20). A sample problem is shown in Figure 18. This particular problem required three keystrokes to perform. Note that each keystroke displayed three new characters on one end while removing three characters from the other. Once the subject had displayed the desired letter and number, during this problem phase, he depressed the space bar, and the next problem was displayed. The computer ignored the space bar depression if the correct letter and number were not displayed; that is, each problem had to be completed correctly before going on to the next problem. During the problem phase of the study, the display was not centered at the beginning of each new problem (as it was in the self-definition phase for the self-defined group); rather, the subject started out each problem with the display remaining at the conclusion of the previous problem. Each problem required between one and six keystrokes to perform. The time required to perform the problems, as well as the number of keystrokes made, were recorded automatically by the System/7 computer.

The 80 display problems were divided into two sessions of 40 problems each. There was a break after the first session, during which time each subject was given feedback regarding his speed and accuracy on the first 40 problems.

Of the 34 subjects in the self-defined group, 30 defined the system in the windowing mode, while only four defined the system in the scrolling mode. This difference was analyzed with the Chi-square statistic, utilizing the Yates correction for 1 *df* tests.[24] The result showed a significant difference: chi-square (1) = 18.38, pF 0.001.

Four separate analyses were performed on the performance data:

1. Mean total time it took subjects to complete the first 40 problems
2. Mean total time for the second 40 problems
3. Mean total moves for the first 40 problems
4. Mean total moves for the second 40 problems

A graphical presentation of the data for these four analyses is shown in Figures 19 through 22. The figures show that in each

Figure 18 Sample problem requiring three keystrokes; (A) centered position, (B) after first keystroke, (C) after second keystroke, (D) after third keystroke

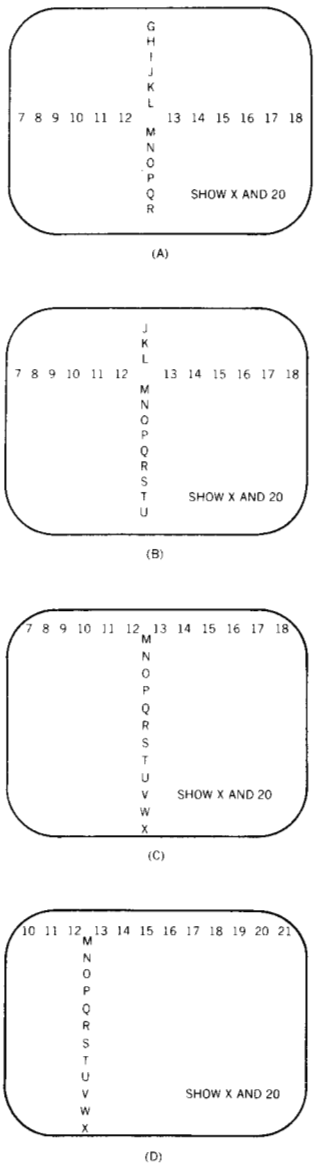


Figure 19 Mean total times, first 40 problems, Experiment 1

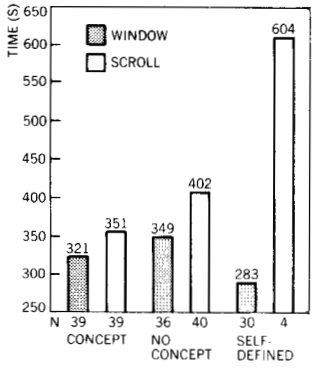


Figure 20 Mean total times, second 40 problems, Experiment 1

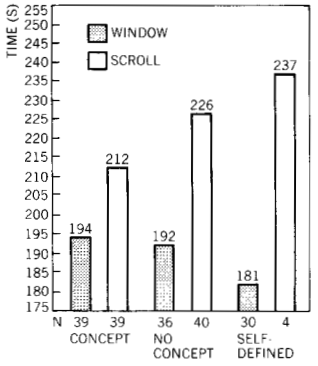


Figure 21 Mean total moves, first 40 problems, Experiment 1

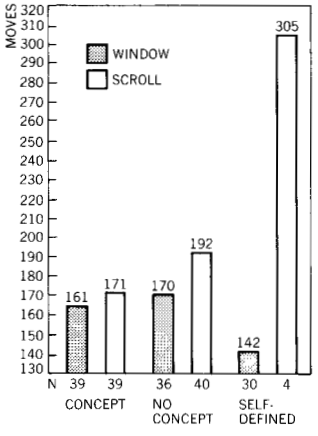


Table 8 Analysis of variance: Mean total times (in msec), first 40 problems—Experiment 1

Source	*SS*	*df*	*MS*	*F*
(Treatment)	(5.349×10^{11})	(5)		
Window/concept vs window/no concept	1.482×10^{10}	1	1.482×10^{10}	<1
Scroll/concept vs scroll/no concept	5.235×10^{10}	1	5.235×10^{10}	3.00
Window self-defined vs window/concept and window/no concept	5.705×10^{10}	1	5.705×10^{10}	3.28
Scroll self-defined vs scroll/concept and scroll/no concept	1.978×10^{11}	1	1.978×10^{11}	11.36†
All window groups vs all scroll groups	2.129×10^{11}	1	2.129×10^{11}	12.23†
Error	3.168×10^{12}	182	1.741×10^{10}	

†$p < 0.001$.

case the windowing groups performed faster and with fewer moves than did the scrolling groups.

For each of the four performance analyses, the between-group variances were divided into five different orthogonal contrasts in order to make the following statistical comparisons:

1. Window/concept versus window/no concept
2. Scroll/concept versus scroll/no concept
3. Window/self-defined versus window/concept and window/no concept
4. Scroll/self-defined versus scroll/concept and scroll/no concept
5. All window groups versus all scroll groups

The results of these analyses are given in Tables 8 through 11. As may be seen, the windowing groups performed significantly faster and with fewer moves than did the scrolling groups. No significant differences in performance were found between the concept and no concept groups. The four subjects who defined the system to scroll (scroll/self-defined group) took significantly more time and a significantly greater number of moves to complete the first 40 problems than did the subjects who were constrained to scroll (scroll/concept and scroll/no concept groups). This effect was not present during the second 40 problems. Finally, subjects in the window/self-defined group made significantly fewer moves to complete the first 40 problems than did subjects in the window/concept and window/no concept groups.

It may be said that under the conditions specified in this study, operation in the windowing mode demonstrated its "superiority" on every measure. It should be pointed out, however, that while

Table 9 Analysis of variance: Mean total times (in msec), second 40 problems—Experiment 1

Source	*SS*	*df*	*MS*	*F*
(Treatment)	(5.127×10^{10})	(5)		
Window/concept vs window/no concept	4.951×10^{7}	1	4.951×10^{7}	<1
Scroll/concept vs scroll/no concept	3.645×10^{9}	1	3.645×10^{9}	<1
Window self-defined vs window/concept and window/no concept	3.067×10^{9}	1	3.067×10^{9}	<1
Scroll self-defined vs scroll/concept and scroll/no concept	1.245×10^{9}	1	1.245×10^{9}	<1
All window groups vs all scroll groups	4.299×10^{10}	1	4.299×10^{10}	9.99†
Error	7.836×10^{11}	182	4.305×10^{9}	

†$p < 0.01$.

Figure 22 Mean total moves, second 40 problems, Experiment 1

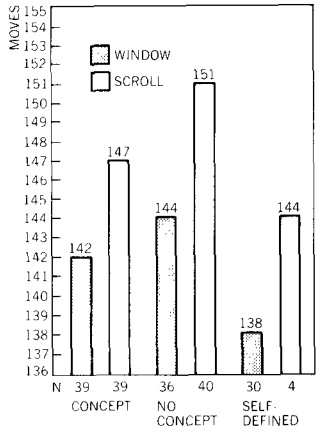


Table 10 Analysis of variance: mean total moves, first 40 problems—Experiment 1

Source	*SS*	*df*	*MS*	*F*
(Treatment)	(1.174×10^{5})	(5)		
Window/concept vs window/no concept	7.159×10^{2}	1	7.159×10^{2}	<1
Scroll/concept vs scroll/no concept	8.469×10^{3}	1	8.469×10^{3}	3.16
Window self-defined vs window/concept and window/no concept	1.326×10^{4}	1	1.326×10^{4}	4.95†
Scroll self-defined vs scroll/concept and scroll/no concept	5.800×10^{4}	1	5.800×10^{4}	21.63*
All window groups vs all scroll groups	3.698×10^{4}	1	3.698×10^{4}	13.79*
Error	4.880×10^{5}	182	2.681×10^{3}	

†$p < 0.05$.
*$p < 0.01$.

the majority of the novice users indicated in a questionnaire a preference for the windowing mode, a fair percentage (12 percent) of users did prefer scrolling. For this reason, whenever possible, users should be given an option of which mode to use, with the default being the windowing mode. In cases where giving that option is not practical, we recommended that the windowing mode be used.

VM/370 NED-SCRIPT and TSM: a comparison of text-editing efficiency

the problem

The following is a description of a study that is not, strictly speaking, a psychological experiment. Rather, it illustrates how the fa-

Table 11 Analysis of variance: mean total moves, second 40 problems—Experiment 1

Source	*SS*	*df*	*MS*	*F*
(Treatment)	(3.419×10^3)	(5)		
Window/concept vs window/no concept	1.070×10^2	1	1.070×10^2	<1
Scroll/concept vs scroll/no concept	3.200×10^2	1	3.200×10^2	1.12
Window self-defined vs window/concept and window/no concept	5.371×10^2	1	5.371×10^2	1.87
Scroll self-defined vs scroll/concept and scroll/no concept	9.651×10^1	1	9.651×10^1	<1
All window groups vs all scroll groups	2.358×10^3	1	2.358×10^3	8.24†
Error	5.214×10^4	182	2.865×10^2	

†$p < 0.01$.

cilities of the Human Factors Center were employed to conduct a simulation study to compare how a system operates, given real data entered by humans. The objective of this study[25] was to determine whether a specific on-line text-editing task could be performed more efficiently using a facility called NED (new editor) and SCRIPT/370 than by using the TSM (Terminal System Moderator) editing and formatting facility. The criterion for determining which of the two text tools was more efficient was the time required to complete a text-editing task. The editing task was identical for both systems, and a set of commands necessary to perform the editing efficiently was written for each system.

procedure

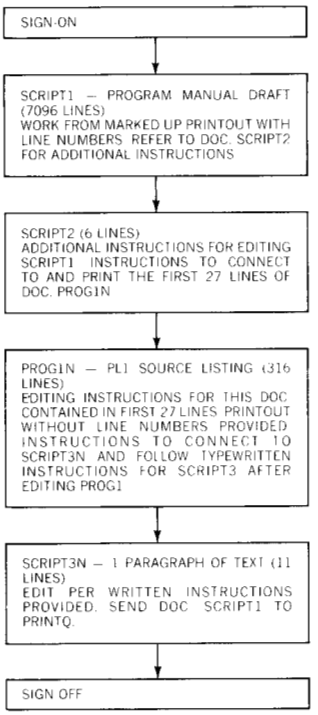
The editing task consisted of additions and/or modifications to a set of text files. A scenario was defined for performing the necessary editing, and then for each scenario, a series of editing activity blocks, 15 in all, was defined. An editing activity block consisted of a set of editing commands required to access and modify a text file. A flow diagram of the editing scenario is shown in Figure 23.

To eliminate the problem of skill variability among individuals, the command set for each editor was transcribed onto a magnetic card and then read into each system using the IBM Communicating Magnetic Card Terminal. The user's input for each system was thus entered at a constant rate and free of errors. Terminal sessions were monitored using the Human Factors Center's MERANDA (Multiple Event Recording AND Analysis) program that records and time stamps data coming to and from a selected ter-

Table 12 Input and output character counts

Editing activity block	*Total input characters*		*Total output characters*	
	TSM	*NED-SCRIPT*	*TSM*	*NED-SCRIPT*
1	29	36	91	62
2	28	19	25	0
3	37	57	131	0
4	6	21	22	0
5	35	44	86	0
6	14	38	0	22
7	28	26	69	0
8	18	29	9	20
9	14	17	45	0
10	22	25	34	1
11	26	30	65	4
12	23	29	66	0
13	44	69	122	54
14	60	63	52	5
15	57	94	127	147
Totals	441	597	944	315

Figure 23 Test script description



minal. Since MERANDA is run on an independent computer system in the Center, no additional load was placed on the system being monitored.

results

In order to evaluate text-editing efficiency, measurements were made on the following aspects of each of the two editors:

- Input character counts
- Output character counts
- Number of commands required to perform editing task
- Number of signed-on users at time script was being run
- Total elapsed time per session
- Total activity block time per session
- Times for individual activity blocks

Characters. Table 12 presents input and output character counts. The large number of input characters required to perform the editing task with NED-SCRIPT/370 is due primarily to the need to specify file name and file type whenever the edit command is issued or whenever other file-related commands (e.g., Print, Get) are executed. Conversely, the output character counts for NED-SCRIPT/370 are shorter than they are in TSM because the CMS (Conversational Monitor System) messages are shorter, and the editing verification messages can be turned off if the user so wishes. The VM/370 (NED) concept of very short messages and elimination of confirmation messages at the user's request are highly desirable features for the expert user.

Table 13 Input command counts per subject

Block	*Total TSM commands*	*Total NED-SCRIPT commands*
1	2	3
2	1	1
3	1	2
4	1	2
5	1	1
6	1	3
7	2	1
8	2	4
9	1	1
10	1	1
11	1	2
12	2	1
13	5	8
14	3	5
15	5	9
Totals	29	44

Commands. Table 13 lists the command count for input. The number of commands required to edit the test script is 35 percent greater on NED-SCRIPT/370 than on TSM. Here the difference is attributed to the need to enter a "Verify Off" command each time editing of a different file is started. Another reason is that the line pointer must be moved to either the top or bottom of the file at various points in the script. Since the TSM does not employ the line pointer concept, these additional commands are unnecessary.

Timings. In evaluating the results, a number of terms will be used that require some definition. First, there were eight sessions during which one of the two systems was run. A *session* is the time from sign-on to sign-off for one run of the test script. The *block total* is the sum of block times per session for each system. Another term used is *adjusted time*, which applies only to NED-SCRIPT/370 and is meant to cover the sum of the block times per session less the time to file large documents (Script1). The *total elapsed time* includes time from sign-on to sign-off plus some session time not in blocks.

Data will also be presented in terms of *mean elapsed session time*, which is defined as the sum of the total elapsed time divided by eight. *Mean block time* is the sum of the block totals divided by eight. Finally, the *mean time for Block n* is the sum of Block *n* divided by eight.

Table 14 is a summary of the timing data. From the data for both the elapsed session times and total activity block times, we can

Table 14 Timing data summary

Editing activity block	*TSM*		*NED-SCRIPT/370*	
	Mean time (sec)	*Standard deviation*	*Mean time* (sec)	*Standard deviation*
1	14.9	3.1	3.1	0.1
2	7.7	1.2	9.2	0.1
3	16.0	0.2	5.1	0.1
4	5.4	1.4	3.1	0.1
5	17.6	1.9	5.8	0.1
6†	17.8	1.0	118.5	26.5
7	13.6	0.3	1.9	0.0
8†	8.6	0.2	102.6	25.9
9	8.1	0.3	1.7	0.2
10	12.1	0.9	2.3	0.1
11	16.5	3.0	9.1	6.7
12	15.8	3.0	3.5	0.1
13†	46.7	6.7	125.9	32.0
14	17.6	1.6	10.8	0.1
15†	30.6	4.3	204.2	42.1
Mean block time		248.9		606.8
Mean adjusted block time*		—		241.4
Mean elapsed session time		367.7		727.3
File "SCRIPT1" time				
Per session		—		365.4
Per command		—		91.4
				(S.D. = 26.1)

†These blocks contained a command under NED-SCRIPT/370.
*For NED-SCRIPT/370, mean adjusted block time = block time less time required to file "SCRIPT1."

see that the editing task was performed approximately twice as fast on TSM as on NED-SCRIPT/370. The increased time to perform the editing on NED-SCRIPT/370 is due mainly to the delay associated with filing document Script1 after each phase of editing. During the course of editing with NED-SCRIPT/370, document "Script1" was filed four times. Since TSM does not make use of the Working Storage concept, file storage times for document "Script1" were significantly shorter (3 to 5 seconds for TSM as compared to 56 to 124 seconds for NED-SCRIPT/370). For all activity blocks in which it was unnecessary to file document "Script1," the times required to perform the task(s) in those blocks were comparable on both systems.

A file time test was run on VM/370 (NED) to determine the time required to file various document sizes ranging from 500 to 10 000 lines, with an average line width of approximately 70 characters. Each document size was filed five times consecutively during three separate terminal sessions. The time required to respond to the "file" command was calculated from MERANDA data. It was

seen that the file time bears a linear relationship to the file size, with response times becoming increasingly long for documents in excess of 1000 lines.

conclusion

Because character counts and command counts are lower on TSM than on NED-SCRIPT/370, the fewer keystrokes result in a higher degree of typing efficiency for TSM. More terminal printing is done on TSM than on NED-SCRIPT/370 (3:1 ratio), which results in increased session time (approximately 42 seconds). This time could be reduced by providing a TSM equivalent to the NED "Verify Off" command.

In addition, the timing data collected during this study indicate that when document sizes are small (500 lines or less) there is relatively little difference in the editing efficiency of the two systems from the standpoint of the user. However, as the size of the document file increases, text editing becomes significantly more efficient on TSM compared to NED-SCRIPT/370.

Summary

By a discussion of certain hardware and software projects, we have illustrated how the Human Factors Center in San Jose implements its mission to evaluate products to determine that the man-machine interfaces have been optimized and assessed. To summarize, we list the steps taken to realize that mission, most of which were described in the projects just discussed.

- *Evaluate the task.* This involves understanding the user's application(s) and hardware, and the characteristics of the operators. An attempt is made, from experience, to help the planners and engineers to assign to the operator and to the machine (or system) those tasks that are most appropriate to each. In the course of clarifying the application, samples of its data base are extracted for use in subsequent tests and simulations.
- *Simulate the application environment.* This means setting up in the laboratory those characteristics of the application environment that are likely to interact with the operators during task performance. An attempt is also made to control extraneous application "noise" that would contaminate the data collection and obscure interpretation of the results.
- *Compute-control the data collection.* The essentials of this step are: (a) identify and record all possible events (such as a keystroke depression, the transport of a document, etc.), (b) measure all times between and associated with the events, and (c) identify and record all errors, both detected and undetected by the operator or system.

- *Analyze the data.* Determine the operator and system performance by calculating throughput and error rates, and, in the case of errors, determine the cause for each.
- *Communicate.* At this point an attempt is made to inform engineering about the results of our tests, and, where appropriate and possible, to recommend design changes that may lead to performance improvements.
- *Iterate.* If time permits and the changes warrant it, we frequently rerun the study to determine that the changes have indeed resulted in real improvements.
- *Define training sequences.* Very often, our simulations suggest ways by which customers may use our test sequences and procedures to train operators.
- *Validate the simulation results.* This involves post-installation visits to review field performance to determine that the results achieved in the laboratory are reflected in the field. Obviously, feedback from such reviews helps to refine subsequent simulations and test procedures.

ACKNOWLEDGMENT

The computing and data collection support system of the Center was originally developed by J. A. McLaughlin (currently with the IBM Tucson laboratory), and we are deeply grateful to him for that work.

The Human Factors Center in San Jose is an outstanding example of team effort. Although individual members may take responsibility as principal investigators for specific studies, they are backed up and supported in their efforts by other professionals in the Center, and a principal investigator of one study may become the professional support on a subsequent investigation. The author of this paper is fortunate to be manager of a technically competent and professionally skilled group of people who have made significant contributions over a period close to 25 years. Contributions of various members of the Center played no small part in the writing of this paper. Current members of the Center, in alphabetical order, include J. M. Boyle, K. F. Bury, C. K. Clauer, R. N. Cotton, W. H. Emmons, R. J. Evey, J. P. Hares, B. S. Isa, D. A. Krysiak, B. W. McVey, A. S. Neal, V. M. Ramos, B. A. Rupp, R. M. Simons, and J. W. Wylie. While it is their work that is described in this paper, any errors of interpretation or exposition are solely the responsibility of the author.

Furthermore, the continuing operation of a facility such as this requires special management attention. Starting with R. B. Johnson, who established this activity initially, the Human Factors Center has had the interest, understanding, and support during the past 25 years of M. M. Astrahan, B. O. Evans, L. V. Kline, C. R. Nichols, L. D. Stevens, and E. F. Wheeler.

CITED REFERENCES AND NOTES

1. A. Dvorak, "There is a better typewriter keyboard," *National Business Education Quarterly* **11,** 51-58, 66 (1943).
2. *Ibid.* p. 51.
3. A. Dvorak, N. I. Merrick, W. L. Dealey, and G. C. Ford, *Typewriting Behavior*, American Book, New York (1936).
4. D. D. W. Davis, "An evaluation of the simplified typewriter keyboard," *Journal of Business Education* 10-11, 11-12 (May), 10, 29 (June), 21-22, 38 (September), 19-21 (October) (1935).
5. R. T. Griffith, "The minimotion typewriter keyboard," *Journal of Franklin Institute* **248,** 399-436 (1949).
6. E. T. Klemmer and G. R. Lockhead, "Productivity and errors in two keying tasks. A field study," *Journal of Applied Psychology* **46,** 401-408 (1962).
7. R. L. Hershman and W. A. Hillix, "Data processing in typing: Typing rate as a function of kind of material and amount exposed," *Human Factors* **76,** 483-492 (1965).
8. P. Rabbitt and J. E. Birren, "Age and responses to sequences of repetitive and interruptive signals," *Journal of Gerontology* **22,** 143-150 (1967).
9. R. L. Deininger, M. J. Billington, and R. R. Riesz, "The display mode and the combination of sequence length and alphabet size as factors in keying speed and accuracy," *IEEE Transactions on Human Factors in Electronics* **7,** 110-115 (1966).
10. R. S. Hirsch was the principal investigator.
11. R. S. Hirsch, *A Human Factors Comparison of Two Keyboards Proposed for Special Purpose and Limited Application*, Research Report RJ151, IBM Corporation, Research Division, San Jose, CA 95193.
12. R. S. Hirsch, "Effects of keyboard formats on typing performance," *Journal of Applied Psychology* **54**, No. 6, 484-490 (1970). Much of this article is reprinted here. Copyright 1970 by the American Psychological Association. Reprinted by permission.
13. Many variations of the arrangement selected were possible. For example, instead of 11 keys each on the top and "home" row of alphabetics, one could have considered nine and 10, respectively; that would have placed seven on the bottom row instead of the four used in the study. Pilot studies that employed some of the alternatives did not appear to show any differences among them. Furthermore, replications (both published and informally reported to us) that used somewhat different numbers of alphabetic keys on the various rows have all confirmed our findings.
14. S. E. Michaels, "Qwerty versus alphabetic keyboards as function of typing skill," *Human Factors* **13**, 419-426 (1971).
15. Visits were made to several Wabun typing schools in Japan, and extensive discussions with school directors, teachers, and students established that 30 characters per minute, after six months of training, was the goal to be achieved, as that level of skill usually assured the student of subsequent employment as a Wabun typist.
16. The complexity and magnitude of this study dictated that it be a team effort in the Human Factors Center in cooperation with the IBM Fujisawa Development Laboratory. Principal investigators were S. Hirose for the laboratory in Japan and W. H. Emmons for the Center.
17. Among the several exploratory efforts that led to that frequency was one that involved blind typing. The only time the typists saw their output was when they pressed a "display" key and were shown the last line of keyed input. The frequency with which the 20 operators called for "display" was an average of once every 30 seconds.
18. Principal investigators were J. W. Boyle for the Human Factors Center, E. Nassimbene representing Los Gatos (AUDIOLINE), and C. Pape representing La Gaude (TOUCHLINE).
19. C. K. Clauer was the principal investigator.
20. C. K. Clauer, *CRT Display Legibility with Reduced Character Size*, Human

Factors Center Report HFC-25, IBM Corporation, San Jose, CA 95193 (December 1977).
21. B. J. Winer, *Statistical Principles in Experimental Design*, 2nd Edition, McGraw-Hill Book Co., Inc., New York (1971).
22. This work was conducted primarily by Dr. John A. McLaughlin, now at the IBM Tucson laboratory.
23. K. F. Bury was the principal investigator.
24. E. W. Minium, *Statistical Reasoning in Psychology and Education*, John Wiley & Sons, Inc., New York (1970).
25. J. M. Boyle was the principal investigator.

The author is located at the IBM General Products Division headquarters, 5600 Cottle Road, San Jose, CA 95193.

Reprint Order No. G321-5144.