CMU-CS-81-140

Working Papers in User-Computer Interface

April 1981

Omer Akin and D. Radha Rao

Laboratory for Design and Information Processing
1319 Doherty Hall
Carnegie-Mellon University
Pittsburgh, Pa.

Contents:

This research was supported by the National Science Foundation, under contract number MSC-7825824.

Table of Contents

List of Tables

Working paper: 1
Efficient Computer-User Interface in Electronic Mail Systems

Omer Akin

ABSTRACT

This research explores the question of improving user-computer interface. The approach is one of observing and codifying various parameters that influence the efficiency of interface in the context of electronic mail tasks. In the first paper we observe "expert" and "regular" users of a mail system and analyze the sources of efficiency. It is clear that experts use a different, more specialized, set of commands in performing standard mail tasks. While experts perform these tasks with fewer errors and more "completely", it is not clear that they achieve this any faster than regular users. Recommendations for system design are made.

In the second paper we codify and examine the errors made during the experiments. Major portions of all errors fall under syntactic and typographical categories. Implications for automatic error recovery are discussed.

1. Introduction

Traditionally, research in computer sciences has had its primary focus on developing models and systems to attain superior design and implementation capabilities. However, with new software becoming available at an ever increasing rate, there is a parallel increase in our need to better understand the mechanics of computer-user interfaces, whether they are of a motor, cognitive, or perceptual kind.

A wide range of user issues are of importance to the computer scientist and many of them have already been studied in different contexts. Some note-worthy examples have dealt with:

- How is behavior organized during user-computer interfaces? (Card, 1978; Penniman, 1975; Hayes and Reddy, 1979)
- What performance differences exist between different computer systems due to user interface? (Roberts, 1979; Card, English and Burr, 1978; Penniman and Perry, 1976) How do we evaluate these differences? (Pew, Baron, Feehrer and Miller, 1977)
- Can we predict user behavior based on system characteristics? (Card, Moran and Newell, 1979)
- What modes of interface are appropriate? (Negroponte,1977; Card, 1978)

These studies deal either with describing a scheme for classification or measurement of interface, or they draw comparisons between alternative user performances. Such information is naturally important in evaluating systems during the implementation or the design stages. The question that underlies most of these issues is: How efficient is the interface in any particular circumstance and how do we measure and encode this level of efficiency?

In this paper we develop a method for encoding and measuring efficiency of use in an electronic mail system. We draw our observations from RdMail[1] , the Carnegie-Mellon University, Computer Science Department mail system. We also develop in a second paper (included in this volume) a taxonomy for codifying errors made while using the mail system.

Mail systems are of interest because they support a wider range of user behaviors than other tasks typically used in studying user interface. Furthermore, component tasks that constitute the electronic mail task have all been studied before, such as, text editing, retrieval and human-like interaction with machines. Consequently, in studying RdMail we are able to look at issues of user interface in the

---

[1] RdMail Message Management System User's Guide and Reference, Computer Science Department, Carnegie-Mellon University. January, 1981.

context of a fairly complex task domain for the first time and with substantial prior understanding of some of its primary sub-tasks. The conclusions we draw shed light on the design and implementation of electronic mail systems as well as composite user system environments that may have similar subtasks, such as interactive programming behaviors, text editing, and so on.

2. The study of behavior during an interface

No published study exists on the problem of user interface with electronic mail systems. Therefore the issues raised by interface studies available to us are only peripheral to our central question and are applicable to our general concerns and methods of investigation. Before we examine issues specific to our investigation, let us first review some of these general issues of user interface.

2.1. Operational Description of the Task Domain

A variety of tasks have been focused on in studying user-computer interfaces. In each case exploration of the work starts with an operational description of the task domain being studied.

Card *et.al.* (1976) studied the "anatomy" of text editing behaviors based on empirical observations of users. They found that cycles of standard editing tasks were sufficient to characterize subject's text editing behaviors. They defined these standard tasks at different levels, like typing and text assembly versus typing single characters. The prediction power of their model was not effected by the grain size of each task and a finite set of *goals*, *operators*, *methods* and *selection rules* accounted for user's behaviors adequately.

Penniman's (1975) study of user behavior with on-line retrieval systems identifies four general categories of operation, i.e., *index search*, *logic formation*, *document display*, and others, such as *begin session*, *print out*, *review search*, *exit*, etc. This paradigm enabled him to distinguish between performance in different retrieval systems. Penniman concluded that further development of these results is contingent on the development of higher order models of user operations.

Hayes and Reddy (1979) studied the components of "graceful interaction" between two communicating systems. While their results are intended to be applicable to the interaction of various human and non-human systems their observations are based on the former. They describe capacities and mechanisms that account for unambiguous communication in natural language, i,e., *flexible parsing*, *domain knowledge*, *explanation facility*, *focus mechanism*, *identification from descriptions*, *generation of descriptions*. This provides a foundation for the formalization of graceful user interface.

All of these studies aim at understanding the nature of user interface within given task domains by describing the parameters of operation in these domains.

2.2. Differences in user performance in different system environments

The most common method of documenting performance differences between alternative systems seems to be through the measurement of user behavior in the context of a standard task.

Roberts (1979) defined standard editing functions and examined user performances for different editors in terms of these functions. She measured performance through time and effort expended in standard tasks (defined through a fatigue factor). Similarly Penniman and Perry (1976) measured session durations and number of interactions in data retrieval systems. They found private data files to be more efficient than general data files as a function of the number of interactions per unit time. This suggests, as expected, that same level of personal file system facilitates use.

Card *et.al.* (1979) developed a method of predicting expert use time as a function of idealized system operation sequences corresponding to standard tasks. They broke down each operation into key-strokes to estimate total times. They point out that display time, a function of the efficiency of the total system environment, was also an important factor. It is important to note that all of these measures are aggregate measures. For example, average time required to perform a task includes a proportion attributable to user-system interface as well as one that is not. In other words, there is a component in all measurements that is a function of the overall system environment, such as terminal speed and operating environment, rather than the specific user system.

2.3. Prediction of user performance

A good measure of our understanding of how user-system interface works is our ability to predict user performance under different system contexts.

Card *et.al.'s* (1979) key-stroke method demonstrates that a significant portion of user time spent in text editing can be predicted accurately. However, the method applies only to expert users with perfect editing strategies and virtually no slack time, e.g., for examining stimuli for the next task, resting, chating with experimenter, etc. Even with these drawbacks Robert's (1979) comparison of the keystroke model's predictions against empirical data stands up well.

2.4. Modes of Interface

Another significant aspect of interface has to do with the devices that assist interface.

More specifically these are the tactile and perceptual tools available to the user; i.e., terminal, CRT, mouse, tablet. The central question is to access the appropriateness and efficiency of such means. Foley *et. al.* (1974) and Card *et.al.* (1978) studied the efficiency and accuracy of various pointing devices. Card found that the mouse was superior to joysticks, step-keys and text keys in speed as well as in accuracy.

2.5. Parameters of Studying User Interface

The above studies illustrate many of the important research questions for studies of user interface.

First, they all attempted to codify the operational aspects of the particular task domains they have studied. In order to structure the questions related to user interface this seems to be an inevitable step. *Second,* a good number of studies have developed experimental controls from this initial codification of the task domain. They have identified typical standard tasks of interest for the system to be studied. This is also a necessary step to observe the effects of system and user properties on the nature of the interface. *Third,* a few of the studies have developed models capable of predicting user performance. This route, being the most robust approach to the problem, is also the most involved one and does not necessarily lead to generalized findings (Roberts, 1979). *Fourth,* studies of devices supplement our understanding of the appropriateness of total user environments. These findings, however, can be easily generalized and replication of findings seems to be an unfruitful path to take while many fresh questions remain available for study.

In this study we shall address some of these new questions in lieu of trying to replicate any of the earlier findings. We shall also develop an operational formulation towards more efficient interface with systems. None of the above studies yield explicit information about how to develop systems with greater interface efficiency or increase the efficiency in given systems. The major purpose of this study is to do just that. We shall examine how a given system adapts to *expert* versus *regular user*[2] behaviors; and how higher levels of efficiency can be attained. We shall do this by evaluating the behaviors exhibited by users of different skill levels in the context of standard tasks defined within the operational description of electronic mail sending. Before we define this task environment let us define what is meant by "efficiency" of interface and how we propose to study it.

---

[2]By expert user we mean users intimately familiar with the system, such as system programers. By regular users, we mean users that use the system daily but are familiar only with a few standard mail tasks.

3. Efficient User-computer Interface

What is efficiency of use and how do we identify efficiency?

An underlying aspect of most studies dealing with user interface is this notion of system efficiency. Roberts (1980) in evaluating text editing systems compares functionalities of different systems based on power of system operations and time required per operation. Card (1978) compares text editor performance in terms of key-stroke time per task and as a function of system displays. Penniman (1975) bases his comparisons on session-duration and number of interactions per task. Although the particulars of each comparison technique is somewhat different the overall similarity underlying them is the notion of performing a *standard task* with finite or, better yet, *minimum resources.*

Efficiency, then, *is accomplishing a standard task, using minimum resources. Time and effort of the users of the system and operating environments of both are examples of such resources.*

How do system environments influence efficiency?

Most of the studies we reviewed first codify characteristic operations and goals of the task environments they examine. Hayes and Reddy (1979) for example provide a detailed description of system properties necessary for graceful interaction. Card (1978) and Penniman (1975) also describe the necessary steps in performing text editing and information retrieval tasks, respectively. These represent what the user has to do in performing a given task as a function of his goals.

Each computer environment designed to support these tasks provides operations that correspond to the user's goals, such as searching, editing and formatting of information. To avoid redundancy and achieve some level of standardization, these operations can be decomposed into smaller operations common to many different forms of mail tasks. Consequently, each computer environment provides sets of low-level, standard operations, such as logging-on, typing and deleting characters, etc., that when packaged appropriately, correspond directly to the goals useful in the general task environment. This implies a methodology that is analogous to means-ends analysis where the operation sequences form the "means" and the task goals constitute the "ends." Consequently, the easier it is to find a set of operations to fulfill a goal, the more efficient is the interface. By "ease" we refer to means that can serve the user's resources best.

An efficient interface system, then, is one in which *maximum conservation of user resources is possible in matching the goals of the user and the system operations available to him while achieving these goals.*

How can we measure the efficiency of interface in a given system? How does this determine an agenda for research?

The first step in measuring user-computer interface efficiency as outlined above requires that the operations and goals of the general task environment and the system to be studied are codified. The system operations have to be specified in a way that relates to the format of the operations available in the system to the goals of the task environment. This is necessary to find the possible matchings between goals and operations. Furthermore, it allows user behavior of different skill levels to be mapped into a standard thus comparable format.

The second step is to show how various users and task environments support different interface patterns as a function of these operations and goals. Such comparisons would indicate how different users and task environments result in superior resource conservation. This requires that resource allocation such as *time*, *effort*, *accuracy*, so on, must be empirically measured. By *time* we mean the time necessary to perform a standard task at the terminal, measured by the real time necessary to perform the tasks. This time includes time spent towards problem understanding, reading, waiting for the system prompts, problem-solving, typing and verbalizing of thoughts. By *effort* we mean the amount of knowledge brought to bear on the standard task at the terminal, measured by the average number of system commands used to perform the task. Here we view knowledge of system commands as a resource, due to the total overhead it implies in the learning as well as the use of the commands. Hence, an efficient system we assume should try to minimize the number of commands necessary to perform a standard task. By *accuracy* we mean the average number of errors made during use. The fewer the errors, the more efficient the system is.

Then a four step research effort is necessary in our investigation,

1. Define task goals and system operations in a given system.
2. Measure user behavior in terms of time, effort and accuracy in performing standard tasks.
3. Compare conservation of these resources by users of different skill levels.
4. Postulate system features that can assist users to perform at desired levels of efficiency.

In the next sections we shall apply this method in the context of the mail system, RdMail, currently in use at the Computer Science department of Carnegie-Mellon University.

4. Use of Electronic Mail Systems.

4.1. Task Goals

Today, messages are sent in many different ways, e.g., US Mail system, singing telegrams, computer networks, phone messages, bottled messages, and so on. The diversity present in conventional mail systems are endless and the mechanisms with which information is packaged and transmitted are either transparent (paper and pencil) or irrelevant (Morse code) to the user.

In computerized mail systems this picture is radically different. The media available for use are finite, yet the actual mechanisms of information transfer, such as, word processing, file transmission, access etc, are less obvious, and more abstract. Consequently, conventional and computer mail systems, while similar in principle, are radically different task environments. The similarities can be found in very general terms. For example, both require selection of a mailing medium, composition, formatting, filing and forwarding of mail. However, when the nature of each of these components are examined closely, fundamental differences become apparent. The operations necessary for transmitting mail in computer systems are not known to the layman *a priori* like in conventional mail systems but have to be learned. Even experienced electronic mail system users do not and need not know the internal mechanisms of the system they use. Therefore, our analysis of mail systems here will be confined to our main area of interest: *electronic mail systems.*

Mail systems are primarily designed to allow transmission of messages between individuals. However the interaction between user and system is multi-faceted. A single homogeneous pattern of interaction will not account for the various sub-tasks generally used in sending mail. Five different categories of sub-tasks are commonly observed; a) Control of system functions, a) Search and display of old messages, c) Formatting and Sending mail d) Modifying message texts, e) Maintenance and creation of records.

4.1.1. Control functions

Two standard sub-tasks in this category are *starting* and *ending* the use of the mail system. Often these are simply equivalent to signing on and off of the mail system. Other control functions are; bypassing the main system using optional short format mail systems; modification of user and system options, accessing help and documentation files, etc. These activities constitute a very small portion of the data we have collected and hence we have very little to say about them.

4.1.2. Survey of information

Often the preparation of the message to be mailed requires examination of past mail. This examination is needed just to inform oneself about the contents of past correspondence. At times the new mail must be based verbatim on past correspondence. Consequently, seeking out relevant mail files and accessing on such files, *search,* and showing their contents, *display,* are standard sub-tasks used in surveying of message files.

4.1.3. Composing and Sending Mail.

Composition is usually done with the aid of predetermined system templates. Most systems prompt the user about the destination, subject, and body of a mail during composition. In special cases, if the user so desires, even these templates can be filled with contents of existing files, and automatic composition takes place. The actual mail *sending* operation is usually a straightforward operation. For example, in RdMail a single "action" called *Mail* is sufficient to forward the message currently composed. The composition portion is the more critical and complex part of this process.

4.1.4. Modifying Mail Texts

Composition can be done piecemeal, composing separately the different parts of a predetermined format, or including text(s) which are extracted from old files using the automatic-composition modes. In either case, like in other word processing tasks, newly composed mail files may have inconsistencies within them. This is why mail systems are equipped with *text editing* facilities, allowing users to revise message files before sending them.

4.1.5. Filing of Mail

Incoming message files are saved in mail systems as a matter of course until they are reviewed by the user. After review they are either saved permanently or deleted. This is often a matter of modifying the status of a file. When file systems get to be large there is a need to classify and organize them for ease of access. This requires that users be able to perform standard *file maintenance* tasks, such as modify status of, classify, copy, delete, as well as create new message entries.

4.2. Codification of User Behavior in RdMail

Above we defined the domain of operations available to the users. From here on in order to explore the initial question we set fourth, about the efficiency of electronic mail systems, we shall document how a user of RdMail moves inside this domain using the resources of the system to fulfill his goals.

During the course of this research we reviewed several mail systems including RdMail and Laurel of the Computer Science Department, Carnegie-Mellon University. Since we had access to the RdMail

system as an experimental medium, our observations from now on will apply only to it.

4.2.1. Standard Sub-tasks.

To create an experimental setting, we built a RdMail file consisting of 68 messages from past correspondence of one of the researchers. We also identified several standard "mailing" tasks to be performed on these messages. This is in accordance with the precedence set by other studies, as reviewed earlier.

1. **Composition task:**
   Compose a message of 2000 words or less about a research agenda and forward to a designated recipient.

2. **Surveying task:**
   a) Find the latest message from a particular individual and forward a copy to a designated recipient.
   b) Find information about a particular subject matter in communication with a particular individual during given dates and forward a copy to a designated recipient.

3. **Filing task:**
   Delete all messages from a designated mail sender.

4. **Surveying and Modifying task:**
   a) Find a message from a particular individual inquiring about a given subject matter. Edit and send this message as an announcement to all users.
   b) Find the latest communication on a given subject matter and write a synopsis and transmit to a designated recipient.

5. **Surveying, Filing and Sending task:**
   Find all communications to and from a particular individual; classify and transmit to a designated recipient.

Six subjects took part in this experiment. Three of our subjects were considered "experts". These were individuals either directly involved in the development and maintenance of RdMail at Carnegie-Mellon University or using the RdMail system extensively in their daily work; i.e., maintaining very large message files requiring a mastery of system capabilities. The other three subjects were individuals with regular knowledge of the system. Although these individuals used the system regularly, this did not involve complex system operations, and consisted of regular receiving and sending of mail. We shall refer to them as "regular" users from here on.

4.2.2. The Data

The subjects were stationed at a standard user terminal (Mini Bee) and interacted with the system through a key-board and CRT. The behavior of all subjects were recorded on video-tape in their entirety throughout the experiments. The subjects verbalizations, their facial expressions and their actions as displayed on the CRT were all recorded on video. Subsequently this data was transcribed into text form and correspondence to generic sub-tasks[3] used, actual system operations performed at the terminal, errors committed[4] and time it took to perform each task were indicated. Table 4-1 contains a sample transcription from the protocol of subject 1.

The behaviors of the subjects were classified into eight sub-task categories. These are slightly different from our a priori catagories because the actual behaviors of subjects varied somewhat from our expectations. The data was transcribed using the following criteria for identifying the beginning of each protocol segment falling in each category.

Subtask 1: *Composing new messages*: When a command is activated causing the user to enter contents of a new message, formatted after an existing message or a standard message format.

Subtask 2: *Displaying messages*: When a command is activated causing the appearance of any part of an existing message file on the CRT.

Subtask 3: *Editing existing messages*: When a command is activated causing the system to enter the edit mode.

Subtask 4: *Sending messages*: When a command is activated causing the system to forward contents of an existing message. The actual mechanics of this in RdMail is the queueing of a auto batch job for intersystem transfer of files.

Subtask 5: *Maintenance of messages*: When a command is activated causing the system to alter the status of a message, i.e., changing either its attribute or class.

Subtask 6: *Searching for a (set of) message(s)*: When a command is activated causing the system to flag a (set of) message(s) or their identification numbers for future operation.

---

[3]These are the sub-tasks defined earlier; composing new mail, displaying messages, editing test, sending mail, maintenance of files, searching the mail files, creating new files and miscellaneous control and system operations.

[4]A discussions of errors is included in the following paper by Rao and Akin.

Subtask 7: *Creating new messages*: When a command is activated causing the system to allocate new space for a new message entry.

Subtask 8: *Control functions*: These are a small set of commands necessary to begin use, modify options, get help and terminate use: such as *RdMail, Mail, Exit, Quit, Action (Abort), QRead, Read, Mail, Ckmail, Phone-msg, Version, Help, Topic, ?, News, Document, Profile, RebuildDirectory.*

These criteria were used to identify the exact point in the subject's protocols where a sub-task commences and by default the previous sub-task ends. In this way all actions of the subjects were categorized into one of the above eight sub-tasks, independent of the experimental tasks. Only a small amount of subject actions (< 1 %) did not belong in any of these categories.

Table 4-1: Sample Protocol

Task 1: Find latest message from Jamie carbonell and Transmit.

Line.No	Time	Sub-task	RdMail operation
1300	0.15	1.1 Find message from Jamie Carbonell (Search)	<- h from :carbonell:
1500	0.41	1.2 Format last message to Rao (Construct)	<- forw *of last/Rao/cc/ subj: message from carbonell
1800	0.12	1.3 Display message (Display)	Action ( )t
1900	0.6	1.4 Send mail (Send)	Action ( )m

Task 2: Delete all the messages from Rao

2100	0.18	2.1 Delete Rao's messages (File-maintenance)	<- del from :rao:
2200	0.24	2.2 Display deleted messages (Display)	<- h last
			(No response for the command)
22300	0.28	2.3 Find messages from Rao	<- h from :Rao:

(Subject moves to next task)

TASK: 3 Find messages from H Smith inquiring about the message system research. Edit and transmit the message so it can be used as an announcement to all CMU-CS users explaining his request.

2400	0.14	3.1 Find message from Smith	<- context from :Smith:
2600	0.17	3.2 Display message	<- t *of cont
2800	3.34	3.3 Edit message	<- Ed
3300	0.26		*p
3600			↑O
3700	0.8		*d600:1200

(Table 4-1: Sample Protocol continued).

Line.No	Time	Sub-task	RdMail operation
3800	0.33		*p/.
4100	0.7		d100:500
4200	1.25		l100
4700	0.16		*p/.
4900	0.47		l 400
5300	0.12		*w:foo.txt
5500	0.8		*e
6100	0.9	3.4 Delete mail	<- del 69
6200	0.2		<- pop
6300	0.4	3.5 Display	<- h 17
6500	0.3	3.6 Undelete mail	<- undelete 17
6900	0.42	3.7 Construct message	<- mail R Rao/cc/ subject: request from Hugh Smith/file:foo.txt
7000	0.18	Display	Action()t
7400	0.5	Send mail	Action()m

5. Experts versus Regular Users

The underlying hypothesis in having two groups of users take part in the experiments was that efficiency of use varied for users with varying backgrounds and skills. We encoded efficiency through three measures as defined earlier; errors, system knowledge and time.

5.1. Errors in Use

Each entry by the subjects consisted of one or more characters typed at the terminal followed by a carriage-return. Such an entry was considered an error if and only if the system responded with an error message. Other errors that were detected before the carriage return was hit were often corrected by the subjects immediately. Errors included commission error, as well as occasional omission errors. By far the most common omission error was the left-out characters and prerequisite commands; where a RdMail command, prerequisite to a command used at the time, was left out or the current one was entered incompletely. Typical commission errors were: mis-specifying the argument of a RdMail command, entering an illegal abbreviation, entering extraneous characters, and so on. A complete codification of all errors is included in the next paper.

Table 5-1: Errors at the Terminal

	Frequency of all RdMail commands	Total of all errors	% of Errors per # of RdMail commands
EXPERTS:			
Subj-1	54	0	0.0
Subj-4	55	1	0.018
Subj-5	31	1	0.032
REGULAR USERS:			
Subj-2	59	5	0.085
Subj-3	57	6	0.105
Subj-6	62	4	0.065
TOTALS:			
EXPERTS	140	2	0.014
REGULAR USERS	178	15	0.054
BOTH	317	17	0.0536

The frequency of errors clearly supported our hypothesis. Experts made far less errors than regular users, a total of 2 as opposed to their 15 (table 5-1). A comparison of the group means indicated that the difference is significant.

5.2. Knowledge as Resource of User

We also assumed that the knowledge users bring to bear on the experimental tasks is a resource like time or accuracy. In other words efficiency is also a function of the amount of knowledge applied to the task at hand. The more knowledge or resources are necessary in performing a given task, the less efficient the interface is. Conversely, efficient use would be characterized by the conservation of knowledge resources brought to bear on the task at hand.

In order to simplify our task we reduced our question about knowledge to "knowledge of system commands." Hence, we measured the knowledge used in terms of the RdMail commands applied to each task. The underlying assumption, here, is that knowledge necessary for applying a command is an invariant. Regardless of the efficiency with which this knowledge is acquired or the effectiveness with which it is applied, the knowledge itself represents a cost for the user which is manifested through the use of commands. The fewer commands are used to perform a standard task the more efficient the system. Consequently, a kind of efficient user behavior can be achieved through the conservation of the user's effort. A crude measure of this is the number of commands used to perform

a standard task.

Then the first hypothesis we tested was that experts used fewer commandsper task than regular users. Our data did not support this.The experts used an average of 6.38 commands per task while the regular users used an average of 7.57 commands (table 5-2). While the difference is in the direction of our hypothesis, its magnitude is not significant (t = .5862, d.f. = 12).

Most of the experimental tasks were rather simple and shared many standard subtasks, i.e., search, create, forward, etc. Hence in our next analysis we retabulated the data in terms of the sub-tasks. In this way we were able to examine each sub-task, how it is executed, what RdMail commands are used in executing it, independent of the particular experimental task it happened to be a part of.

Table 5-2: Number of RdMail commands used in each experimental task.

	TASKS*								
	1	2a	2b	3	4a	4b	5	TOTAL	# of Commands per task
EXPERTS:									
Subj-1	4	3	10	14	7	10	**	48***	8.0
Subj-4	4	2	19	10	1	15	3	54	7.71
Subj-5	3	2	4	6	3	10	4	32	4.57
REGULAR USERS:									
Subj-2	5	3	15	8	6	7	5	49	7.0
Subj-3	4	2	6	10	3	21	4	50	7.14
Subj-6	15	5	6	12	11	6	5	60	8.57
AVERAGES:									
Experts	3.67	2.33	11	10	3.67	11.67	2.33	44.67	6.38
Regular users	8.0	3.33	9	10	6.67	11.33	4.67	53.00	7.57

* See section 4.2.1 for definition of tasks. ** Task not done. *** Based on 6 tasks.[5]

Table 5-3 shows a tabulation of the commands for each sub-task. Commands not shared by experts and regular users are included separately. Analysis of variance indicates that a significant portion of

[5]Furthermore, the frequencies examined in our earlier tabulations are totals that include repeated uses of the same commands. Since we are really interested in the repertoire of commands used rather than the absolute numbers, in our retabulation we included only the frequencies of types of commands used and not all instances of use.

the variance in the data is accounted for by:

a) the different sub-tasks for all commands: $F(p < .01) = 28.71$; d.f. = 7,1 and

b) the different subject groups for commands not shared between subject group, $F(p < .01) = 19.34$; d.f. = 7, 1

This supports the hypothesis that while the total number of commands used by the experts is more or less the same as that of regular users, the pool of commands used exclusively by experts is significantly greater in number than that of the regular users. In other words, experts have a wider "vocabulary" of commands, but this does not necessarily cause them to do fewer number of things at the terminal. In fact, no significant difference exists between the number of commands used per task by the two user groups.

Table 5-3: Number of known RdMail commands used per sub-task category.

	SUB-TASKS								
	Compose	Display	Edit	Send	File	Search	Create	Control	Totals
EXPERTS									
Not shared* Commands	3	4	4	6	13	21	0	2	53
All** Commands	4	7	7	7	16	25	1	3	70
REGULAR USERS									
Not shared* Commands	2	0	2	2	9	6	0	1	22
All** Commands	3	3	5	10	14	19	1	2	57
TOTALS									
Not shared* Commands	5	4	6	8	22	27	0	3	75
All** Commands	7	10	12	17	30	44	22	5	127

* Commands not shared between Experts and Regular users. ** All commands.

5.3. Time Spent at the Terminal

The above counter-intuitive conclusion would be accounted for, if the experts spent less time at the terminal executing their commands. This would imply that although just as many are used the commands used by the experts execute in less time and are therefore more efficient. At first we found that experts spent significantly less time ($F (p < .05) = 11.55$, $df = 5,1$) at the terminal compared to regular users (table 5-4). This implies that while the number of operations were the same, the time each operation took and the particular combination of operations used by the experts was more efficient than regular users.

Table 5-4: Time Spent at the terminal

		Reading text from Screen	Typing	Waiting for display to appear on Screen	Waiting for system Response
EXPERTS	Subj: - 1	4.10	7.03	8.06	7.39
	Subj: - 4	4.43	28.02	5.15	6.38
	Subj: - 5	3.30	8.54	2.26	3.34
	TOTAL	11.83	43.59	15.47	17.11
REGULAR USERS	Subj: - 2	13.17	15.35	8.01	4.00
	Subj: - 3	16.44	40.59	7.38	6.28
	Subj: - 6	10.51	16.40	4.45	7.10
	TOTAL	40.12	72.34	19.84	17.38

We measured total time at the terminal in aggregate form which includes time of:

a) reading experiment instructions
b) reading long texts off of the screen
c) typing text and commands
d) waiting for the system to complete visual display of text
e) provide verbal explanations, and
f) waiting for system to execute commands.

Obviously the factor we are directly interested in is the last factor: how do experts compare to regular users in terms of time spent while waiting for the system commands to execute? On the other hand, some other factors-items (b), (c) and (d) above-can also be argued to indicate efficiency of commands used. That is, if the command allows us to inspect contents of a message with minimum display on the screen, or if it takes less time to type (or to display on the screen), then the system is efficient.

Hence, we compared direct execution times between experts and users as well as the indirect times due to typing and displaying functions (table 5-5).

Table 5-5: Time spent on each Task.

	TASKS						
	1	2	3	4	5	6	TOTALS
EXPERTS	2.03	0.97	20.45	33.12	4.94	13.32	74.83
REGULAR USERS	9.93	4.44	21.20	47.78	10.03	20.64	114.02
TOTALS	11.96	5.41	41.65	80.90	14.97	33.96	188.85

There was no significant difference in command-response times between the two subject classes ($t = 0.176$). On the other hand for reading and typing times there were significant differences ($t = 16.19$ and 4.036, respectively d.f. = 2). This implies that while the specialized commands used by the experts take just as long to execute, they possibly are the cause of less visual search and display on the screen, consequently amounting to efficiency in the long run.

6. Implications for System Design

6.1. Efficient Use

The measures of efficiency we have used imply that expert users are more accurate and faster yet not particularly efficient in their use of knowledge. Experts select their commands from a larger pool of system commands. That is, in addition to a small set of core commands shared with regular users, the experts also use a large set of specialized commands (table 6-1).

This implies that, in a limited sense, the regular users are in fact more efficient than the experts. They perform all tasks with as much effort (measured in terms of number of commands per sub-task) using a significantly smaller set of commands. In terms of the effort necessary for acquisition and maintenance of knowledge about RdMail commands, the regular user's performance is clearly more efficient. They perform just as well with less resources. Consequently, they use each operation on the average more frequently than the experts (2.74 times as opposed to 2.0 times for the experts.)

On the other hand, experts make fewer errors and, due to the specialized commands, make better

use of the system's resources. They are also significantly faster in terms of command execution time spent at the terminal.

6.2. Designing for Efficient Interface

First, let us start with the unexpected finding, that regular users are as efficient as the experts in using their knowledge resources. The command structure of RdMail allows users to perform the same sub-task with different commands. For example both "mail" and "forward" are commands that communicate message(s) to other users, the main distinction being the way in which the message can be modified prior to sending. *Forward* allows to append a prefix to existing messages (s) and sends every thing to other users; while *mail* prompts the user about various parts of the message; e.g., its headers, text, etc. Hence, users can perform the same sub-task through generalized commands as well as through commands that can be modified through arguments to better suit the special circumstance at hand.

The data suggests that the larger portion of the commands shared by both user groups are of a generalized form. While the commands used only by experts are highly specialized ones. regular users apply very few such specialized commands Table 6-1 shows the basic commands used by subjects without the keywords and arguments that go with them. Even in this form the experts have a larger repertoir of commands. This result is consistent with one's intuition. What is inconsistent with our intuition is that this does not allow the experts to perform their tasks with any fewer number of commands than the regular users.

Second, it seems inevitable that experts and regular users (and we suspect novices, whom we did not study here, are also in the same boat) operate at different levels of accuracy. Errorful behavior is rarely a problem for experts. regular users, however, encounter errors more frequently and consult *help* and *documentation* files often. regular users spent an average of 10.42 minutes at the terminal in using an average of 7.67 help commands through out the protocols, as opposed to experts who rarely get help (in our case, none). For greater efficiency, user systems should be equipped with automatic documentation and error recovery capabilities. In the next paper we shall discuss the latter at greater detail.

In order to prevent these specialized help features from hindering other groups of users, such as experts, systems must also be adjustable to specific user profiles. RdMail has the option of being modified by the user based on his needs and preferences. However, this requires extensive knowledge of the system and very few users venture into it. A documentation system better

integrated with user commands and automatic error recovery capabilities triggered through incidence of errors and user habits can mean more graceful interaction with different user groups.

Table 6-1: Commands used by Experts and Regular users.

SUBTASK	EXPERTS	REGULAR USERS
1. Compose	<- MAIL[6]	<- MAIL
2. Display	<- TYPE <- EDIT <- ↑L	<- TYPE <- EDIT
3. Edit	<-ALIAS <- EDIT <- MAIL <- RETRY	<- RETRY <- EDIT <- MAIL <- RETRY
4. Send	<- MAIL <- FORWARD <- REMAIL <- ANSWER	<- FORWARD <- REMAIL
5. File	<- DELETE <- ALLOCATE <- DEALLOCATE <- CLASSIFY <- MOVE <- EDIT <- PUT	<- DELETE <- LIST <- SORT <- FORWARD <- MOVE
6. Search	<- HEADERS <- WHO <-↑Z <- CONTEXT <- NUMBER <- CURRENT <<- from[7] <<- subj <<- field to <<- from	<- HEADER <- WHO <- READ <- HEADERFIELD
7. Create	<- CREATE	<- CREATE

[6] Upper case words represent RdMail commands

[7] This functions as a command within the mode of the MULTIPLE command.

Table 6-1: continued

SUBTASK	EXPERTS	REGULAR USERS
8. Control	<- POP	<- EXIT
	<- QUIT	<- QUIT

Finally, in systems with many redundant commands which are suitable for different levels of user skills, and which represent different efficiencies in terms of system resources, i.e., CPU time, user awareness of these issues can lead to more efficient interface. If the users are aware of how efficient a command is, such as *forward*, compared to a generalized one, such as *mail,* they can: a) choose to learn or not learn the new command, and b) decide when to use one as opposed to the other.

Since the efficiency of a command is also a function of the arguments used, the size and the contents of the message files, and the particular context the user is in, it is difficult to measure efficiency of each command in normative terms. Consequently a system feature that supplies feedback to users on the actual efficiency of the commands they regularly use can ultimately achieve substantial efficiencies in use by influencing user habits.

6.3. User Recommendations for Improving RdMail

One of the tasks we used, the composition task,called for a short statement from the subjects about what they would like to improve about RdMail. In this way we obtained explicit feedback on the system and desired improvements on it. This is a useful source to verify against our independent observations.

The subjects made twelve explicit suggestions about RdMail (table 6-2).Two of these comments agreed with our observations about the documentation of help commands. Another four comments suggested improvements about the display capabilities of RdMail. Two recommendations were made about maintenance and search of RdMail files, each. And the remainder of the comments were on editing and sending message files.

The two comments which were encountered in multiple instances were about help messages as mentioned above and speed of the system. It seems that there is a need to quickly survey mail files without getting involved in extensive text handling. Other significant system improvements suggested were:

1. Ability to manage visual data on the screen through a window, to scan short headers, simultaneously with other text or graphical information, while

composing text.

2. A graceful editing capability built into the mail system, rather than moving into an edit mode identical to the standard text editing system.

3. Greater facility to survey message contents without displaying all contents of message files.

Table 6-2: Subjects Suggestions about RdMail

Subjects	Desired Features	Undesired Features
subj-3	1. Ability to send hard copy mail (Sending) 2. Ability to bring old mail to ones attention in future. (Maintenance)	
Subj-5		1. Pointer marks old message after editing is complete. (Search) 2. "New mail" message does not display headers. (Display)
Subj-4	1. Real window management ability for: - Short headers - Text to assist composition - Graphic information. (Display) 2. Graceful editor interaction. (Edit)	1. Too slow. (General-Display) 2. Inability to run multiple programs. (Control)
Subj-2	1. More basic-form content in help files. (Control)	1. Help files assume greater than elementary knowledge of the system. (Control)
Subj-6	1. Faster start-up and initial access into files. 2. Ability to use a second "subject" field in order to identify file contents Better: Esp. through the header command. (Maintenance-Search) 3. Cleaner help command interaction. (Control)	1. Too slow. (General-Display)

Summary

This study represents a preliminary examination of some of the issues related to user efficiency in electronic mail systems. We have neither exhausted all issues nor sufficiently examined all questions raised. However, we have provided some answers to the questions posed, as well as uncovered a counter-intuitive aspect of user efficiency in RdMail. The specialized commands that constitute the "vocabulary" of experts did not reduce the number of "things" done at the keyboard such that regular users were at times as efficient as experts in terms of the number of commands used in executing a standard task.

7. Bibliography

Card, Stuart K. (1978) Studies in the Psychology of Computer Text Editing Systems. Xerox Palo Alto Research center. Palo Alto, California.

Card, S. K., Moran, T.P., and Newell, A. (1976) The manuscript editing task (p76-00082). Xerox Palo Alto Research Center, Palo Alto, California.

Card, S. K. English, W. K., and Burr, B. J. (1978) Evaluation of mouse, rate-controlled isometric joystick, step keys, and text selection on a CRT. Ergonomics 21 (1978), 601-613.

Card, Stuart K., Moran, Thomas P., and Newell, Allen. (1979) The keystroke-Level Model for user Performance time with Interactive systems (SSL-79-1). Xerox Palo Alto Research Center. Palo Alto, California.

James D. Foley., Victor L. Wallace. (1974) The Art of Natural Graphic Man-Machine Conversation. Proceedings of the IEEE, VOL. 62, NO. 4, April 1974.

Hayes, P. J., and Reddy, R. (1979) Graceful interaction in Man-Machine Communication. Tech. Report, Computer Science Department, Carnegie-Mellon University.

Martin, J. (1973) Design of Man-Computer Dialogues. Englewood Cliffs, N.J.: Prentice-Hall, Inc.

Moran, Thomas P. (1978) Introduction to the Command Language Grammar. Xerox Palo Alto research center. Palo Alto, California.

Negroponte, N. (1977) Computer mediated inter and intra personal communications. MIT, Architecture Machine Group.

Nickerson, R. S. (1977) On conversational interaction with computers. In S. TRUE, Ed., User-oriented Design of Interactive Graphics Systems, pp. 101-113. New York: Association of Computing Machinery, Inc.

Penniman, D. W. (1975) A stochastic process analysis of on-line user behavior. Battelle's Columbus Laboratories.

Penniman, D. W. and Perry, J. C. (1976) Tempo of on-line user interaction. Battelle Columbus laboratories.

Pew, Richard W.; Baron, Sheldon; Feehrer, Carl E.; and Miller, Duncan C. (1977) Critical review and analysis of performance models applicable to man-machine system evaluation. BBN Report 3446. Cambridge, Massachusetts: Bolt Beranek and Newman Inc.

Roberts, T. L. (1979) Evaluation of Computer Text Editors. Palo Alto Research Center. Palo Alto, California.

Working paper: 2
Summary of user errors at the Terminal[1]

D. Radha Rao and Omer Akin.

[1]This research was supported by the National Science Foundation, under contract number MSC - 7825824. For more information, contact O. Akin at the Laboratory for Design and Information Processing, 1319 Doherty Hall, Carnegie-Mellon University, Pittsburgh, Pa. 15213.

Table of Contents

List of Tables

1. Introduction

This paper is part of an effort to develop a model to measure the level of efficiency in interfacing electronic mail systems with users at the terminal. In this paper, we define the guidelines for codifying errors made during the interaction. These findings will suggest ways of developing new systems with greater efficiency and improving the performance of existing systems.

The experimental task was made up of a set of standard tasks necessary for managing electronic mail.[2] The standard operations of Rdmail, the system we studied, includes manipulating large message files. The set of the standard tasks taken into consideration are *composing, editing, searching, reviewing, sending, filing and creating new messages.*

1. Composing: Assembling and formatting the contents of messages.
2. Editing: Editing an already existing message text, changing headers.
3. Reviewing: Reviewing newly constructed or edited messages, drafts and blind copies.
4. Sending: Forwarding messages to other users of the system.
5. Searching: Searching a given message file to find particular messages by inspecting their contents, headers, source and dates.
6. File maintenance: Classifying messages according to date, content, subject.

All six subjects volunteered to participate in the study were familiar with Rdmail, the electronic mail system widely used in Computer Science Department, Carnegie-Mellon University. Three subjects we considered "experts" are those who are either directly involved in the design and maintenance of the Rdmail system or use the system extensively in their daily work. The other three subjects we considered "routine users" are familiar with the system and use it on a regular basis. All the subjects were asked to perform the tasks outlined above in Rdmail.

2. Summary of Error Categories

There are two basic categories discerned from errors made at the terminal by our subjects: the *symptom* and the *source* of the errors. By *symptom* we mean appearance or form of the error. By *source* we mean the probable cause for the errors to occur. We based our categorization on the *symptom* and further elaborated each category based on possible *sources*.

---

[2]The findings reported in this paper are based on an empirical study observing errors by users of an electronic mail system called Rdmail, which is currently in use at the computing facilities of the Department of Computer Science, Carnegie-Mellon University. The experiment was the same as the one described in the previous paper.

Mistyped character errors are simple typographical errors most probably caused through the mechanics of typing. There seems to be a multitude of forms these errors come in. We categorized these errors into *transposition, substitution, omitted character* and *extra character* errors.

Grammatical, natural language errors are defined as all errors made while typing text which is not in direct response to system command prompts. Text entered into files, user notes and comments are some examples of such text. The grammatical errors made in this context, then, correspond to natural language syntax errors.

Grammatical, command language errors are all errors made while responding to system command prompts. Some examples are: *mis-typing key-words or arguments* in a system defined command, and *using an inappropriate command.*

2.1. Mistyped character errors

2.1.1. Transposition

Transposition error is the reversing of character positions in a word. For example;

suing :: = [3] using;

Struab :: = Straub;
seplling :: = spelling;

2.1.2. Substitution

The substitution error is manifested where one or more character(s) are typed while other character(s) were intended. This error seems to generate from many different sources.

Multiple key character error, a form of substitution, is caused by typing the wrong special character key, i.e., "control", "tab", "escape", "shift", "back-space", due to the unfamiliar position of that key on the current key-board. For example;

↑i :: = I

Another common substitution error is *inserting an unintended character* in place of the intended one. For example;

h from :f :: = h from :Rao;
+ 73 :: = :73

[3] In this notation the right hand side of :: = indicates the correct entry while the left-hand side indicates the erroneous entry, by the user.

Yet another substitution error occurs when a *special character key is not released* in time to correctly type the next character. For example;

AFter :: = After;
THis :: = This.

Shifting key position error is caused by placing the hands in a position which is one or more keys removed in either direction (up-down or side ways) from the usual hand position. This results in the shifting of each key character on the key-board, equal to the shift of the hand(s), respectively. For example;

svibr :: = above
skdi :: = also[4]

2.1.3. Omitted character

These are errors that occur where one or more characters intended to be typed are completely omitted. For example;

*d600 :: = *d6000

Key-position error another possible form of this error category occurs when certain keys typed in a given sequence result in the users inability to produce the adequate motion in making the appropriate key-stroke; often resulting in an insufficiently depressed key and a missing character in the text. In the examples below the subject was unable to hit the "o" key when it occured after "m," a number of times.

mre :: = more;
mve :: = move.

2.1.4. Extra character

This error consists of typing one or more unintended characters. A popular instance of this is *the double character error* caused by hitting two keys of the terminal simultaneously which is mis-read as a duplicated electrical signal, causing a duplicated character to appear on the screen. For example;

subbj :: = subj;
boguus :: = bogus;
souurce :: = source;

Another common instance of this error is simply *typing an extra character* which is not intended. For example;

facests :: = facets;

---

[4]These examples do not come from the protocols, but was fabricated to illustrate the point.

efforst :: = effort.

2.2. Grammatical Errors of Natural Language

In all of the above examples simple typographical errors are produced and they are quickly recognized by the users and corrected on the spot. Other errors that are of some syntactic significance involve the re-assessment of larger portions of the text to detect and recover from. Usually knowledge of syntactic rules of natural language are necessary to accomplish this. For example in the following cases the subjects alter the grammatical and semantic contents of their sentences:

thus :: = the;
Here is the :: = Here are the messages.

2.3. Grammatical Errors of System Language

These are errors caused by the misuse of the Rdmail system commands. There were four kinds of such errors encountered in the protocols: *sequence of command, improperly specified argument, improperly specified abbreviations,* and *meta-variable errors.*

2.3.1. Sequence of command

Often a command is used in a context where other commands are necessary to perform operations on the data to enable the initial command to work. That is certain logical dependencies exist between commands due to the special circumstances of the data the system is operating on. When such pre-requisite commands are left-out an error occurs. For example, one subject realizes that he has to *create* before *moving* a file, after first attempting to move the file.

2.3.2. Improperly specified argument or key-word

Often a keyword or argument, or both are improperly specified in a command. This is a common error. For example, the *last* command is erroneously specified at first, and then corrected to the *context* command:

forw la :: = forw cont.

2.3.3. Abbreviations of keywords

Improper abbreviations of system-reserved words is also a common error, especially with inexperienced users. For example, below the "Wholeheaders" command is erroneously abbreviated as "wh".

<-wh :: = <-w

2.3.4. Failure to substitute for meta-variable

Sometimes the user takes a meta-variable used in the help message illustrating the use of a regular variable as the variable itself and uses it in constructing his commands. This will cause the erroneous use of the meta-variable as an argument. For example;

<- h from <name> :: = <- h from "Jones"[5]

3. Frequency of Errors

The above error categories were sufficient to codify all errors made by the six subjects. Appendix A contains a complete list of all codified errors. The examples in each error category imply that certain error recovery strategies would be effective in reducing the time of recovery from errors and/or occurrence of errors. For example, most typographical errors can be identified through a spelling corrector by comparing with a pre-compiled dictionary of words. Subsequently, a "pattern matcher" of sorts can classify each error into a sub-category. If the mistyped words map into a known word by means of transformation of the hand position on the key-board, then, this error category would be "shifting of key-position." If words with mixed cases are encountered, i.e., "AFter", then the system can identify the error as "unreleased special character key." Once error sub-categories are identified, then the system can assist in automatic recovery by substituting the correct spelling of the word, after consulting the user.

Needless to say such automated recovery features are costly to implement. For one thing there is the initial overhead of system development. But more importantly, the overhead in terms of on line response times may be the most critical deterrent to the implementation of such automated error recovery systems. In fact significant delays will result from this and the purpose of automatic recovery may be defeated, that is less efficient user interface may result.

Consequently, two parallel questions must be asked: Can automated recovery or better yet error avoidance be realized with little or no overhead? And what frequently encountered error categories should such efforts be focused on?

The answer to the former question is likely to be affirmative in those cases where autopometric adjustments to key-boards, can remedy the problem, i.e., for "multiple-key character" and "key-position" error. This does not apply to other types of errors as easily and the control of these errors is beyond the scope of this discussion.

---

[5] This example does not come from protocols, but was fabricated to illustrate the point.

However, we can answer the later question here. The implication of the later question is that recovery from error should focus on the most frequently encountered error categorie(s). We can guide the system designer's efforts by examining the frequencies of occurrence and identifying most common errors. In this way error recovery strategies can take into account the frequency with which these errors are expected to occur.

Table 3-1: Error Categories

Subject type	Typographic	Natural language	System syntax	Total
Expert	64	4	6	74
Routine User	62	5	16	83
Total	126	9	22	157

The first question we asked was, if the frequency of errors varied as a function of subject categories and/or error categories as defined above (table 3-1). Analysis of variance indicated that while the subject types did not account for variance in these frequencies ($F(2,1) = 0.69$), error categories did ($F(2,1) = 105.44$). In other words, the frequencies observed for different error categories varied significantly. The frequencies of occurrence for sub-categories of errors also fluctuated considerably:

Transposition	14
Multiple key character	1
Simple substitution	35
Extraneous special-character key	2
Shifting key position	0
Omission	23
Omission due to key-position	11
Duplication	33
Extra character	6
Natural language grammar	9
Command sequence	2
Improper key-word	19
Improper abbreviation of key-word	1

The categories of some significance (those with frequencies higher than 10) are: *transposition, simple substitution, omission, duplication* and *improper keyword.* The only statistically significant difference between the number of errors made by subjects is in the "system syntax" category. Cumulatively the routine users made much more syntactic errors (16) than the experts (6) ($t = 3,057$;

44.40	<- h headersearch from "cohen"	12-improper command Illegal message sequence or "headersearch"- missing quoted string. Headersearch from "cohen"
	<- hheadersearch "cohe"	8-double char.
	<- h to "cohen"	12-improper command
	<- h field	12-improper command Illegal message sequence at "FIELD" - unexpected and of line FIELD
	<- h to "cohen" -	12-improper command illegal message sequence at --" "unknown symbol TO "COHEN"
50.06	Imrp :: = Improvements	1-transposition
51.12	abour :: = about	3-inserting unintended char.
52.48	help :: = I think	10-grammatical natu.language
53.32	b: :: = written	??

SUBJ: 4

Location on Transcription	Error	Error category
....	<- htu :: = run from carbonell	12-improper command
2.32	belon :: = below	3-inserting unintended char.
3.15	cold :: = could	6-complete omission
4.10	pru :: = print	3-inserting unintended char.
8.15	te :: = the	6-complete omission
8.55	syystems :: = systems	8-double char
9.10	fi :: = ?	????
9.45	Thankyou :: = Thankyou!	10-grammatical natu.language
17.00	synopsus :: = synopsis	3-inserting unintended char.
17.25	Ga :: = Grateful	6-complete omission
17.40	notnion :: = notion	9-extra char. not intended
18.40	tryying :: = trying	8-double char.
18.50	thus :: = the	10-grammatical natu.language
19.00	facests :: = facets	9-extra char. not intended
19.20	uuse :: = use	8-double char.
20.00	se :: = specification	6-complete omission
20.50	specifible :: = specifiable	7-key position
21.30	had :: = has	Other: 10OR 3-grammatical natu.language OR inserting unintended char.
21.35	bbeen :: = been	8-double char.
21.40	approah :: = approach	7-key position
22.05	efforst :: = effort	9-extra char. not intended
22.55	oor :: = for	Other: 3- OR 8 inserting unintended char. OR double char
24.35	usr :: = user	7-key position
26.15	sppelling :: = spelling	8-double char.
	<- put field totu :: = <- h	12-improper command

	with cohen <- put foo/donc	11-command sequence "donc lookup error. No existing file. Put aborted"
28.40	cohel :: = cohen	3-inserting unintended char.
31.40	<-s :: =<-deall	12-improper command
33.00	<ddea :: = <-da :: = deall	8and6-double char.andcompplete omission
35.15	oof :: = of	double char.
35.25	ad :: = and	Other: 6 OR 7-complete omission OR key position
35.40	of coure :: = ofcourse	10-grammatical natu.language
36.00	usr's :: = user's	7-key position
36.45	thig :: = thing	7- key position
36.50	hed:: = headers	7- key position
37.50	tes :: = text	3-insering unintended char.
38.30	syys :: = system	8-double char
39.00	sourlce :: = source	9-extra char. not intended
39.15	ppic :: = picture	8-double char.
39.30	ec :: = excetp :: = except	6and1-complete omission and transposition
39.40	mak :: = manage	????
	ht :: = think	1-transposition
40.00	workking :: = working	8-double char
40.05	suppii :: = -support	3and8-inserting unintended char and double char.(unint.ch = next key)f22.03
40.20	user :: = used	3-inserting unintended char.
41.00	sum :: = substantive	3-inserting unintended char.
42.10	moe :: = more	7-key position
42.10	keystroked :: = keystrokes	3-inserting unintended char.
42.50	bl :: = plesant	3- " "
43.00	add :: = and	3and8-inseerting unintended char and double char
43.10	suus :: = system	3and8-inserting unintended char and double chaar.
43.40	reaally <--really	8-doubkle char.
43.45	ht :: = them	1-transposition
43.55	invvoca :: = invocation	8-double char.
44.00	exaamination :: = examination	8-double char.
44.10	craig :: = Craig	10-grammatical natu.language

SUBJ:-5

Location on Transcription	Error	Error category
8.00	subbj :: = subj: "graceful interaction"	8-double char.
12.25	mre :: = more	7-key position
12.50	seplling :: = spelling	1-transposition
14.00	su :: = using	1-transposition

18.58	Strua :: = Straub	1-transposition
	<<-to "struab"	12-improper command
		No such command
20.09	souuce :: = source	8-double char
20.15	editted :: = edited	8-double char
20.30	THis :: = This	4-not releasing sp.char.key
20.50	displlay ::= display	8-double char.

SUBJ: 3

Location on Transcription	Error	Error category
1.30	form :: = fron :: = from	1and3-transposition and inserting unintended char.
	<- send	12-improper command. No such command
3.00	mss :: = message	6-complete omission
7.35	subju :: = subject	3-inserting unintended char.
23.30	Hays :: = Hayes	6-complete omission
23.39	suggestiions :: = suggestions	8-double char.
24.00	oo :: = of	Other: 3 OR 8-inserting unintended char. OR double char.
24.53	*s :: = *d2300	12-improper command
25.00	*d3300+ :: = *d3300:3600	3-inserting unintended char.
31.00	tri <--trying	3-inserting unintended char.
32.00	oo :: = of	Other: 3 OR 8-inserting unintended char. OR double char
33.30	CMUD :: = CMUA	3-inserting unintended char.
34.00	<-h112 :: = <-h12	8-double char.
36.00	Messagge :: = Messags	8- double char.
37.00	essagees :: = Messages	8- double char.
38.00	during :: = during	10-grammatical natu.language
39.40	ch :: = cohen	6-compplete omission
	<- hall	Other: 12 OR 6-improper command OR complete omission
41.52	dcon :: = dcohen	6-complete omission
42.00	"ch :: = "cohen"	6- complete omission
	<- mov from "cohe", to "cohen"/dcohen	12-improper command
		Illegal message sequence at " " - unknown symbol FROM "COHE", TO "COHE"
	<- mov to "cohen"/dcohen	12-improper command
		Illegal message sequence at " " - unknown symbol. TO "COHEN"
44.45	<-ehp :: = <-help	1-transposition
	<- hetu :: = helptu	1and6-transposition and complete omission

	<- helptu :: = h to "cohen"	12-improper command Illegal message sequence at " " - unknown symbol TO "COHEN"
50.10	"ch :: = "cohen"	6-completre omission
	<- h to "cohen"	12-improper command
56.40	/de :: = dcohen	9-extra char. not intended
58.30	"ch :: = "cohen"	6-complete omission
58.40	imprive :: = improvements	3-inserting unintended chaar.

(SUBJ: 3 continued on next tape)

1.2	sentt :: = sent	8-double char
5.10	main :: = mail	3-inserting unintended char.
8.46	taks :: = takes	6-complete omission
9.49	redeliverd :: = redelivered	6-complete omission

SUBJ: 6

Location on Transcription	Error	Error category
12.00	R.- Rao :: = R.Rao	10-grammatical natu.language
	<- headers from carbonell	12-command error.Illegal message sequence at "FROM"- missing quoted string FROM carbonell
15.00	Macdonals :: = Macdonalds	6-complete omission
18.00	*ii :: = *i	8-double char.
22.03	carbonnell :: = carbonell	8-double char.
22.25	ch :: = cohen	7-key position
23.50	Rah :: = Radha	7- " "
	E :: = Rdmail	????
24.54	a :: = I am	7-key position
25.20	, :: = ;	Other: 10 OR 6-garmmatical natu.language OR complete omission
25.40	doesnt :: = donesn:t	3-inserting uninteended char.
	donesn:t :: = doesn't	3and9-inserting unintended char and extra char. not intended
26.00	fles} :: = files"	9-extra char.[)]not intended
	<- mail 48,51,62	12-impproper command "48" not found in addresse file "51" not found in addresse file "62" not found in addresse file
26.20	Alo :: = Also	6-complete omission
26.40	filed :: = field	1-transposition
26.55	wuu :: = would	8-double char
27.50	meaningu :: = meaningful	6-complete omission
28.15	Alos :: = Also	1-transposition
29.00	"FRAZZEL" :: = "FRAZZLE"	1-transposition